Partially Unified Multiple Property Recursive Partitioning (PUMP-RP): A New Method for Predicting and Understanding Drug Selectivity

Thomas P. Stockfisch*

Accelrys Inc., 9685 Scranton Road, San Diego, California 92121

Received August 10, 2002

The decision tree method for classification problems has been extended to accommodate multiple dependent properties. When applied to drug discovery efforts this means a separate activity class can be predicted for each of several targets with a single tree model. A new tree representation and growth procedure, PUMP-RP, has been developed. The final architecture of the tree allows for easy interpretation as to which independent variables and split values are important for all targets and which are specific to a given target. It should thus be usefully applied to studies of drug specificity. A side benefit of the new method is that it can make use of data with missing (or even sparse) dependent property values. This has the potential to leverage copious data from an older, well-studied target while beginning to study a newer target for which only a small amount of data are available.

1. INTRODUCTION

In recent years recursive partitioning has emerged as an important analytical tool in the field of high-throughput screening. The resulting decision trees not only have reasonable predictive power but also are easy to interpret, leading to increased understanding of how to design active compounds. The method is well suited to the large quantity of categorical data (e.g., molecules classified as either "active" or "inactive") that is currently being generated.[1−3] However, optimizing the binding only to a single target (or optimization of a single physical property) without regard to others is usually insufficient. In many biological systems the side effects of a drug are due to its indiscriminate binding to multiple targets. For example, two isoforms of cyclooxygenase, COX-1 (constitutive) and COX-2 (triggered by inflammatory insults), are known. COX-2 inhibitors are anti-inflammatory agents, but COX-1 inhibitors damage the gastrointestinal tract.[4] It is, therefore, desirable to understand what features can allow a molecule to inhibit COX-2 while simultaneously not inhibiting COX-1. In principle, separate decision trees could be developed which would allow for the prediction of each target; however, this approach has two main drawbacks. First, enough data must be collected to determine two models. Second, given the large number of molecular properties a decision tree might utilize, it is unlikely that the two trees can be compared to discover exactly what distinguishes the two targets. Therefore, an extension to recursive partitioning has been developed that creates a single tree that predicts more than one dependent variable. It is herein termed Partially Unified Multiple Property Recursive Partitioning ("PUMP-RP") and is the subject of a patent application.[5] The method arranges for nodes that apply to all targets to be separated from those that apply only to particular targets. It thus constitutes a partial unification of the representation of the multiple properties in tree form. The separate parts of the tree make clear which features the targets have in common as well as how to increase the likelihood that a molecule will be selective for just one target. Another application for PUMP-RP is to model additional physical properties. For instance, we might be interested in simultaneously predicting not only activity but also toxicity and absorption. An additional benefit of the method is that it can be used even if all the properties have not been measured for each molecule in the training set. The goal of the current article is a clear explication of the algorithm as well as a demonstration of some of its statistical capabilities. Validation of the method on a real system is deferred to a separate companion article[6] which discusses an application of the technique to the aforementioned COX-1/COX-2 selectivity problem. The remainder of the current article is organized as follows. Section 2 reviews the single-Y tree-building procedure which is used as a starting point for the new multi-Y method. In this article dependent variables are referred to as "Y" variables or "properties", and independent variables are referred to as "X" variables or "descriptors". The new multi-Y method is presented in section 3. Testing on an artificial data set is described in section 4, followed by a discussion of advantages of the method in section 5.

2. BACKGROUND—REVIEW OF SINGLE PROPERTY RP

The new method is developed from a particular variant of recursive partitioning promulgated by Breiman et al.,[7] implemented in the Cerius[2] CSAR software module,[8] and called single-Y RP as reviewed here. Single-Y RP creates a set of yes/no questions which are organized into a hierarchical tree form with one question per node. The answer to each question determines the remaining questions that are asked for a given molecule, i.e., the remaining nodes on the tree that are traversed. The process ends at a terminal ("leaf") node in which a class prediction is made. Each node (leaf or otherwise) has an associated class which is determined by plurality rule based on the training set data. The tree is constructed in a deterministic fashion from available training

* Corresponding author phone: (858)799-5000; fax: (858)799-5100; e-mail: tps@accelrys.com.

10.1021/ci0203794 CCC: $25.00 © 2003 American Chemical Society
Published on Web 07/24/2003

data in three phases: question-asking, growth, and pruning. First all possible questions are asked involving single descriptors that could potentially partition members of the training set. Questions are in the form of an inequality for numeric descriptors, or equality for categorical descriptors, respectively. During the growth phase, each question is scored at each node according to a growth phase metric. Various metrics are available with one of the most successful being the Gini Impurity. In this case the impurity *I* of a node is given as

$$I = \sum_{i \neq j} p_i p_j$$

where p_i is the fraction of the members of a node that belong to class value *i* and similarly for p_j. The Gini metric maximizes the decrease in impurity (ΔI) from a potential node question

$$\Delta I = I - p_L I_L - p_R I_R$$

where p_L and p_R are the fraction of node members that partition each way for a given question, and I_L and I_R are the impurities after branching. The process of asking a question at a node and creating two new descendent nodes from the answer is called "splitting". We begin at the root node that contains all the training examples and keep splitting using the best question until no split offers an improvement in the growth phase metric. At this point the tree is usually too large (overtrained). In the final phase the tree is pruned back to a more reasonable size, representing a balance between accuracy on the training set and simplicity. Simpler trees tend to give more reliable predictions on new molecules not in the training set. A branch is a node and all its descendent nodes down to the leaf nodes. A "subtree" of a tree is any tree that includes the root node and has zero or more branches removed. The result of the pruning phase is the subtree of the original overgrown tree that minimizes the pruning phase metric

$$R_\alpha = R_0 + \alpha N_{\text{leaf}}$$

where R_0 is the (optionally weighted) number of training set examples that are incorrectly classified by the tree, N_{leaf} is the number of leaf nodes, and α is a tunable pruning parameter that controls the size of the tree. The misclassifications can be weighted differently for different classes, for different Y variables, or for different types of molecules. This is commonly necessary when there are a small number of actives and a large number of inactives. In this case a reasonable weight factor for a given class is the total number of observations divided by the number of examples of that class. The method can be extended to continuous Y variables by predicting the average of the training examples in a leaf node. R_0 is then the (optionally weighted) sum of absolute deviations of training set examples.

There are only a finite number of values of α that result in a different optimally pruned tree, because unless α is increased enough to remove at least one node the change has no effect on the final tree. The choice of value for α can be done by intuition and visual inspection of the resulting trees, by cross-validation, by quality of prediction on a test set, or by some other criteria. Variations on this process include specifying a minimum number of molecules per node, limiting the number of questions for each X variable, and specifying a maximum allowable tree depth.

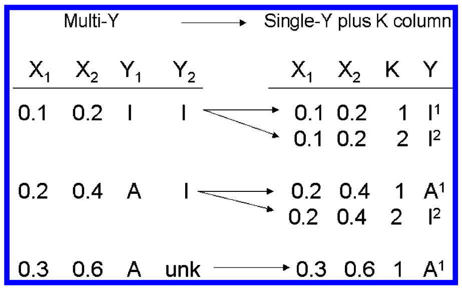

Figure 1. Mapping to single Y. The example data set shown has two descriptors (X_1 and X_2), two properties (Y_1 and Y_2), and three rows (molecules). Each property can take on the class values "I" and "A", with "unk" representing an unknown value. After mapping there are three "descriptors" (X1, X2, and K), just one property (Y) and five rows. The single property takes on the class values I1, A1, I2, and A2. Note that the "unk" value disappears.

3. PUMP-RP ALGORITHM

A. Overview. First the multi-Y data is mapped to a single-Y form. The algorithm described in section 2 can then be utilized. We first create a tree that models the different Y variables independently and then gradually modify it into one which treats them all the same. Branches are saved during this process enabling us to optimize the tradeoff between the specificity and generality of the model. In the final model, splits which are important for all Y's lie near the root, while those specific to a given Y end up near the leaves.

B. Mapping to Single-Y Representation. We start with a training data set consisting of D descriptor values $\{X_1, X_2, ..., X_D\}$ and P property values $\{Y_1, Y_2, ... Y_P\}$ for each molecule in the set. There is thus one "row" of data for each molecule. Figure 1 illustrates this representation. We make P copies of each row, keeping all of the descriptor values but only one of the property values in each copy. In the first copy we keep the property value for Y_1, in the second copy we keep Y_2, etc. The *k*th copy retains the property value for Y_k. For bookkeeping purposes, should the same class name be used for different properties, we add the value of *k* as a superscript to the class name. To keep track of which property is retained in a row we add a new descriptor with its value set to *k*, and refer to it as descriptor "K". We now have the $(D + 1)$ descriptors $\{K, X_1, ..., X_D\}$ and only a single property, Y.

This representation can accommodate missing Y values: we simply do not make a copy of the row for that property. Further consideration of missing Y values is deferred to Subsection H. A simple example is given in Figure 1. The single-Y tree architecture can now be used for prediction provided we are always careful to ensure that there is a split on the K descriptor before reaching any leaf node. The root node contains samples of every property type (in equal numbers if there is no missing data), and any leaf node contains samples of only a single property type. To predict the *k*th property of a new test molecule, we send the associated $\{K,X_1,...,X_D\}$ descriptors through the tree with K set equal to *k*. However, it turns out that the new K descriptor must be treated in a special way, or else it will almost always win out when scored with almost any growth phase metric.

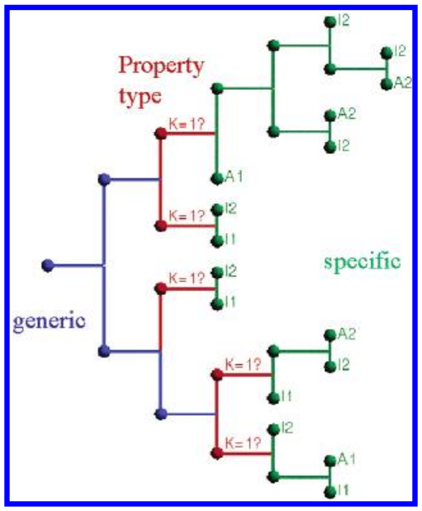

Figure 2. PUMP-RP tree. This hypothetical PUMP-RP tree for two properties consists of a generic portion (shown in blue) near the root that applies to both properties; a specific portion (shown in green) near the leaves that specialize for a single property; separated by a layer of splits on property type (shown in red) somewhere in between. In this and all tree depictions “true” branches downward and “false” upward.

Instead of making a binary split on K, we make them multiway. That is, if the number of properties P is greater than 2, then we do a P-way split when we split on K. The rest of the tree is always binary. The consequence of this is that the nodes that split on property type (called “K nodes”) collectively constitute a single layer that separates a “generic” portion of the tree from a “specific” part. Every node in the generic part applies to every property, and every node in the specific part applies to only one particular property (see Figure 2).

C. Growing the Pure Specific Tree. To decide where K nodes should best occur, we first grow a tree with a split on K at the root. Each node below this one contains data from only a single property. We construct the rest of the tree using the single-Y algorithm in section 2. That is, we split nodes repeatedly using the growth phase metric and then prune the overgrown branches back using the pruning phase metric. This result is called the “pure specific tree,” since we have a completely independent model for each property (see Figure 3). In fact, all the predictions are the same as if we had run single-Y RP separately on each of the properties and created P different trees.

D. K Node Regrowth. We save the entire pure specific tree as the alternate specific branch S_0. Then we resplit the root node using some descriptor other than K and using a modified splitting rule. The growth phase metric for each of the properties is computed separately, and the worst one is used as the score. For instance, for Gini Impurity we use

$$\min_k \Delta I_k$$

as the score, where ΔI_k is the drop in impurity for the classes of Y_k resulting from the split. This rule guarantees that a split appropriate for each property is chosen. The resplit node is now termed a “generic” node since its split is useful for every Y_k. Each branch is now immediately split on K again and regrown using the single-Y procedure, including pruning. However, it is not allowed to prune K nodes. This ensures that a leaf node consists entirely of a single property. Instead of discarding S_0, it is associated with the new root node for possible later use. At this point we are beginning to grow the generic portion of the tree and move the layer of K nodes leaf-ward.

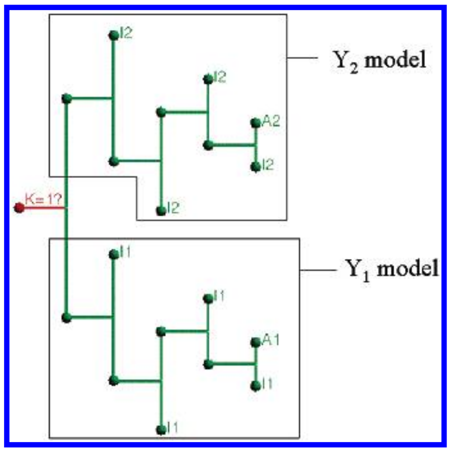

Figure 3. Pure specific tree. The pure specific tree has a split on property type (K) at the root, followed by two completely independent trees. The two branches below the root are what would be obtained by running single-Y RP separately on each property.

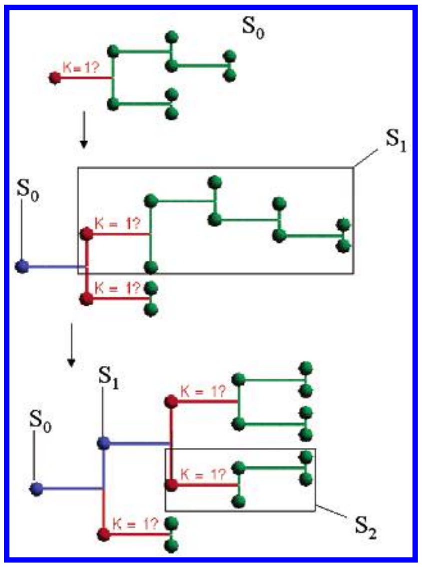

Figure 4. Regrowing specific branches. Each regrown K node creates one generic node and one saved alternate specific branch (shown connected with a black line). Note that generic nodes (blue) are separated from specific nodes (green) by K nodes (red). Also note that the size of the generic portion of the tree is increasing.

E. Continued Regrowth. Each K node is regrown, recursively, as outlined in subsection D, creating additional generic nodes and saving alternate specific branches S_1, S_2, etc. (see Figure 4). This procedure is continued until one of two stopping criteria is met: (i) No split considered in place of the K-split has a positive score for each Y_k. For Gini impurity this means that no split has a positive drop in impurity for each Y_k (i.e., $\min_k \Delta I_k$ is less than or equal to zero). (ii) No splits below a K node survive pruning.

The first criterion ensures that every node in the generic subtree is truly generic. The second criterion keeps statistically unimportant splits from appearing in the generic part of the tree.

F. “Specializing” the Maximally Generic Tree. When all regrowth ceases we have a tree which is “overgeneralized” and needs to be “specialized”, usually containing nothing

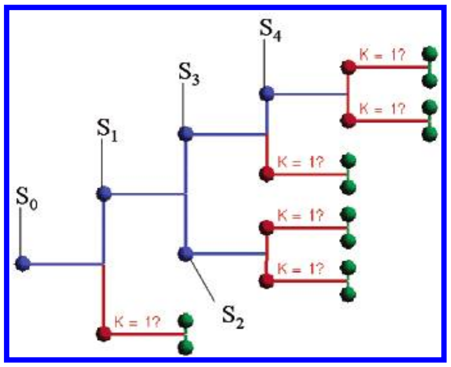

Figure 5. Maximally generic tree. No more generic nodes can be created. Alternate specific branches (S_0, S_1, S_2, and S_3) are saved so that they can be restored through a later procedure that optimizes the tradeoff between an overly general and overly specific model.

but generic and K nodes with leaf nodes immediately following (no specific nodes). This is somewhat analogous to the point in single-Y RP at which we have an overgrown tree that needs to be pruned. Associated with each generic node is the alternate specific branch which it replaced. We call this model the "maximally generic tree" (see Figure 5). Many different trees can be constructed from this one by following either the main generic branches or the alternate specific branches at each generic node. All such trees are related by the fact that the generic part of each is a subtree of the maximally generic supertree. To determine the best tree we minimize a new "specialization phase metric"

$$R_{\alpha\beta} = R_0 + \alpha(N_{\text{leaf}} - \beta N_{\text{generic}})$$

where N_{generic} is the number of generic nodes, and β is a new "generality" parameter. We can find the best tree derivable from the maximally generic tree (for a particular value of β) in computation time proportional to the number of nodes. This is accomplished by starting at the root and comparing the generic main branch with the associated alternate specific branch recursively, choosing the branch with the smaller $R_{\alpha\beta}$. The resulting tree will have the desired structure: generic nodes near the root and specific nodes near the leaves, with each leaf of the tree being separated from the root by a K node (see Figure 2).

G. Finding All Distinct β. There are only a finite number of different optimal solutions that can be obtained by adjusting β. We can find all of them through a combination of bracketing and bisection, typically requiring much less computation time than creating the maximally generic tree. It is generally desirable and feasible to inspect all of the solutions. If a single tree is desired for use in screening, then the best one can be chosen either by intuition or cross-validation.

H. Missing Y Values. Recall from subsection A that if the kth property value for a molecule is missing, then we simply do not make a copy of the row for that molecule when performing the map to single Y's. There is no requirement that even a single molecule have complete property data present in the training set. The number of observations (used in calculating the error rate R_0) is

$$MP - N_{\text{missing}}$$

where M is the number of molecules in the training set and N_{missing} is the number of missing property values. The only

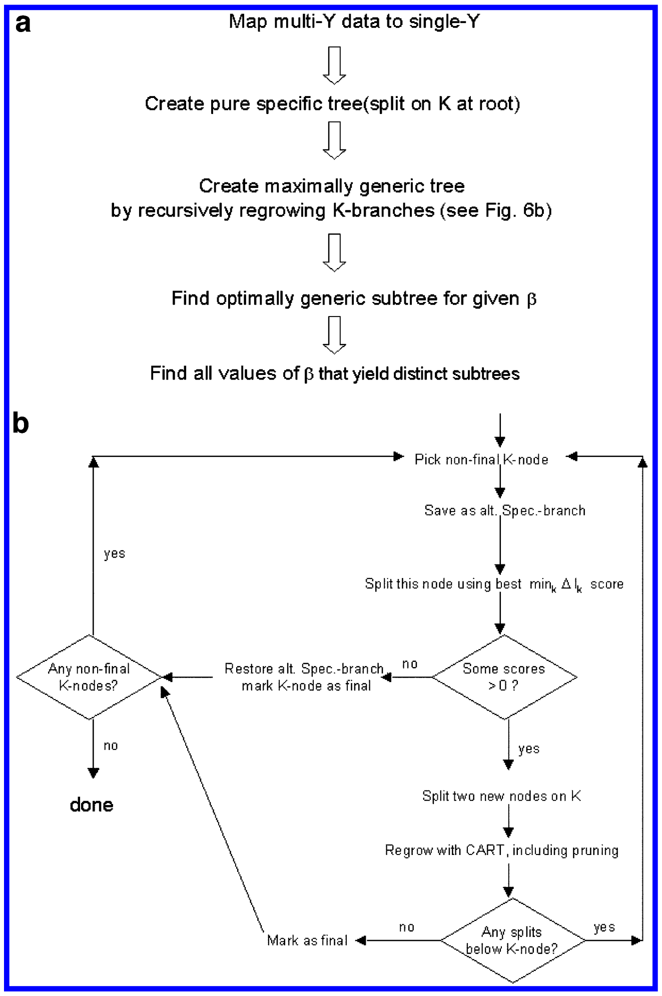
a
Map multi-Y data to single-Y
Create pure specific tree(split on K at root)
Create maximally generic tree by recursively regrowing K-branches (see Fig. 6b)
Find optimally generic subtree for given β
Find all values of β that yield distinct subtrees
b
Pick non-final K-node
Save as alt. Spec.-branch
Split this node using best min_k Δ I_k score
Some scores > 0 ?
no
Restore alt. Spec.-branch mark K-node as final
Any non-final K-nodes?
yes
no
done
yes
Split two new nodes on K
Regrow with CART, including pruning
Any splits below K-node?
no
Mark as final
yes

Figure 6. Flowchart: (a) Major steps of the PUMP-RP algorithm. (b) Chart of the step in which a maximally generic tree is created from the pure specific tree.

qualitative difference in the models produced is that it is possible one of the nodes below a multiway K split does not contain any examples, resulting in an empty terminal node. If a molecule falls into this node, then the prediction of the tree will be "unknown" for that property. In calculating error rates a prediction of "unknown" is always counted as a miss.

I. Summary. The specific part of the tree is grown with the specific node growth metric and pruned back with the pruning metric R_α. Both of these metrics come from single-Y RP. The generic part of the tree is grown with the generic node growth metric, and the size of generic part relative to the specific part is determined by the specialization metric, $R_{\alpha\beta}$. The generic part of the tree is not pruned directly. Instead, parts of it are replaced by previously saved alternate specific branches. The entire algorithm is displayed as a flowchart in Figure 6.

4. TESTING WITH ARTIFICIAL DATA

A. Artificial Data Set. Many new statistical methods are first tested on real-world data, perhaps in the belief that this will prove the method's validity. However, this approach has drawbacks, the chief one being that the true model that governs the real-world system is not known, even for famous training sets, and the initial success of the method in this limited domain is too likely to be an artifact. Problems with

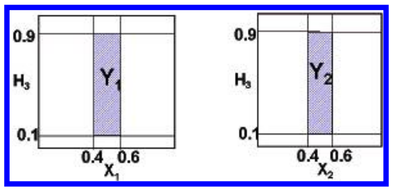

Figure 7. Artificial data set The region in blue contains all of the actives, though only 50% of the samples in this region are active.

the method can be obscured by debates about the real physics of the problem. In other realms of inquiry it is traditional to try a new methodology on a system for which the correct answer is known with near certainty. The present study is an attempt to extend that tradition to statistical modeling. Fortunately, artificial data sets in any quantity can be created with known a priori probability distributions. To make them appropriate for the intended application areas some of the known statistical properties of the problem area can be built in. High throughput screening data is characterized by the following: (1) a small percentage of actives, (2) correlation among descriptors, (3) descriptors which are irrelevant, (4) noise, and (5) incomplete information preventing any model, no matter how good, from avoiding frequent false positives. The following artificial problem has elements of all five of these characteristics built into it. We start with eight uniform random variables on the interval [0,1]: X_1, X_2, H_3, H_4, H_5, X_{rand}, N_a, and N_b. Two more variables are defined in terms of these:

$$X_{3a} = 0.9H_3 + 0.1N_a$$

$$X_{3b} = 0.9H_3 + 0.1N_b$$

The property Y_1 is assigned "A" if $0.4 < X_1 < 0.6$ and $0.1 < H_3 < 0.9$ and $H_4 < 0.5$. Otherwise it is "I".

The property Y_2 is assigned "A" if $0.4 < X_2 < 0.6$ and $0.1 < H_3 < 0.9$ and $H_5 < 0.5$. Otherwise it is "I".

The descriptor set is X_1, X_2, X_{3a}, X_{3b}, and X_{rand}. Several features of this model should be noted. Although Y_1 and Y_2 depend critically on H_3, this hidden variable is not directly present in the descriptor set. Instead, X_{3a} and X_{3b} "contain" H_3 as a hidden variable, while also containing 10% noise and being highly correlated to each other. Y_1 and Y_2 also depend on H_4 and H_5, respectively, yet neither of these is available in the descriptor set. A perfect Bayesian probability model would therefore have at least a 50% false "A" rate. Note that Y_1 and Y_2 have the same dependence on X_{3a} and X_{3b}; Y_1 depends on X_1 and not X_2; and Y_2 depends on X_2 and not X_1. X_{rand} is a pure noise descriptor that contains no information. The active regions are illustrated in Figure 7. Overall, 9% of the classifications are active (i.e. are assigned property "A").

B. Complete Y Results. A training set of 1000 examples of $\{X_1,X_2,X_{3a},X_{3b},X_{rand},Y_1,Y_2\}$ was generated using a pseudo-random number generator. If we apply single-Y RP to Y_1 and Y_2 separately and determine α through cross-validation, we get the trees shown in Figure 8. The first thing to note is that despite the fact that Y_1 and Y_2 are correlated, we get two completely different trees. The common dependence of Y_1 and Y_2 on the correlated descriptors X_{3a} and X_{3b} is masked by the fact that one tree happens to pick X_{3a}, and the other happens to pick X_{3b}. We get the correct dependence of Y_1 on X_1 and Y_2 on X_2, but there is no hint that these descriptors

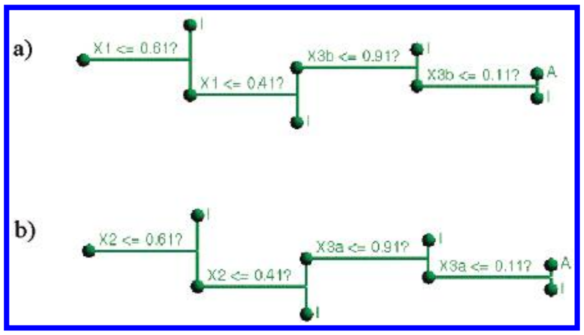

Figure 8. Single-Y RP applied to artificial data set. Two trees resulting from applying single-Y RP separately to each property. Note that one tree involves X_{3a} and the other X_{3b}, masking the commonality of the properties.

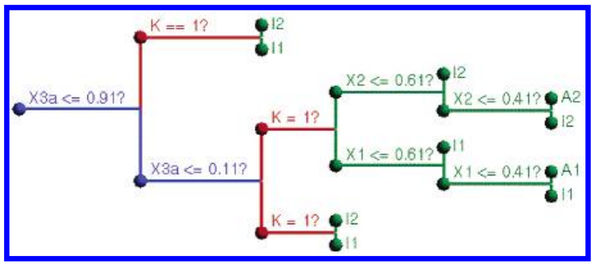

Figure 9. PUMP-RP applied to artificial data set. Note the common use of X_{3a} for both properties in the generic portion and the specialized use of X_1 and X_2 in the specific portion.

are involved in selectivity. The results of PUMP-RP on this set (with α and β determined through cross-validation) are shown in Figure 9. Note that the common dependence of both targets on X_{3a} is readily apparent from its appearance in the generic part of the tree. Also note that X_1 and X_2 appear in the specific part of the tree.

C. Maximally Sparse Y Results. To test the ability of PUMP-RP to accommodate sparse property data, a second data set was generated. This set consisted of 1200 examples with the same descriptors and Y_1 but with Y_2 missing and 800 examples with Y_2 but Y_1 missing. Not a single observation contained both properties resulting in a perfectly sparse set and a considerable challenge for the handling of missing data. Nevertheless, the same final tree shown in Figure 9 was obtained.

5. DISCUSSION

The principle advantage of PUMP-RP for drug selectivity is that as long as progress can be made on all targets simultaneously (i.e. generic tree growth) then no specialization to a particular target is performed (i.e. specific tree growth). Once this is no longer possible, a multiway split on target type is made, and the tree is allowed to adapt to any remaining unexplained target-specific particulars that are important enough to survive pruning. This is a considerable advantage from a model building point of view over separate models because unnecessary specialization to particular targets is avoided. Moreover, it can be inferred from the structure of the tree which characteristics are shared and which distinguish the targets. The results in section 4 confirm that PUMP-RP is able to form a combined representation of multiple targets to the extent that they are similar, together with separate representations to the extent that they are different. Also shown is the same ability even for a data set with sparse target information. This suggests that information

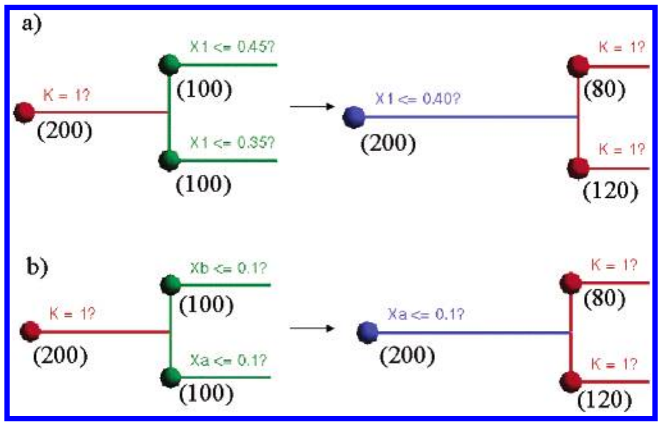

Figure 10. How more generic trees are simpler models. (a) Replacement of two specific nodes with different cut values by a single generic node with the same cut value. The population of each node is shown in parentheses. The common cut value in the generic node is better determined (statistically) due to the greater population. (b) Replacement of two specific nodes with different descriptors by a single generic node using a single descriptor. Besides being determined by a larger number of samples the generic version is arguably simpler since only one descriptor is utilized.

from similar targets can be used as a constraint on tree construction when starting work on a new target for which little activity data has yet been collected. Previously, information on other targets could only be used based on a researcher's intuition as to whether a tree for the new target was physically reasonable.

Besides better interpretability it can be argued that the multi-Y RP trees should be more predictive than single-Y. How this can occur is illustrated in Figure 10. The potential improvement occurs when two splits below a K node (split on property type) are merged into a single generic one. In Figure 10a two splits occur on the same descriptor but for different cutoff values. Note that the common generic split is determined by twice as many examples as the two specific splits. In Figure 10b two splits on different but related descriptors are replaced by a generic split on only one descriptor. This results in a smaller number of descriptors used. PUMP-RP should prove useful in modeling selectivity, pharmacokinetic properties such as ADME, and combinations thereof in conjunction with a target activity. The interpretation of the generic part of the tree for multitarget studies is clear: it is the set of criteria which the targets have in common. Since it is common for toxicity and absorption to be measured for only a subset of the molecules for which activity has been measured, the ability of the algorithm to utilize incomplete data should be quite valuable. An interesting question is the meaning of the generic part for a multiphysical study (e.g. activity and absorption). One possible interpretation is that these nodes describe drug-like character for the target(s) in question. Future applications of the method to such data should clarify this matter.

REFERENCES AND NOTES

(1) van Rhee, M.; Stocker, J.; Printzenhoff, D.; Creech, C.; Wagoner, P. K.; Sear, K. L. Retrospective Analysis of an Experimental High Throughput Screening Data Set by Recursive Partitioning *J. Comb. Chem*. **2000**, *3*, 267−277.
(2) Chen, X.; Rusinko, A., III; Young, S. S. Recursive Partitioning Analysis of a Large Structure−Activity Data Set Using Three-Dimensional Descriptors *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 1054−1062.
(3) Rusinko, A., III; Farmen, M. W.; Lambert, C. G.; Brown, P. L.; Young, S. S. Analysis of a Large Structure/Biological Activity Data Set Using Recursive Partitioning *J. Chem. Inf. Comput. Sci*. **1999**, *39*, 1017−1016.
(4) Talley, J. J.; Bertenshaw, S. R.; Brown, D. L.; Carter, J.; Graneto, M. J.; Koboldt, C. M.; Masferrer, J. L.; Norman, B. H.; Rogier, D. J., Jr.; Zweifel, B. S.; Seibert, K. 4, 5 Diaryloxazole Inhibitors of Cyclooxygenase-2 (COX-2) *Med. Res. Rev.* **1999** *19*, 199−208.
(5) Patent pending.
(6) Rao, S.; Stockfisch, T. P. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 1614−1622.
(7) Breiman, L.; Friedman, J. H.; Olshen, R. A.; Stone, C. J. In *Classification and Regression Trees*; Chapman & Hall/CRC: Boca Raton, U.S.A., 1984.
(8) C2.CSAR, version 4.6, 2001, Accelrys, Inc., San Diego, CA.

CI0203794