JOURNAL OF CHEMOMETRICS
J. Chemometrics 2004; **18**: 275–285
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cem.873

An introduction to decision tree modeling

Anthony J. Myles, Robert N. Feudale, Yang Liu, Nathaniel A. Woody and Steven D. Brown*
University of Delaware, Newark, DE 19716, USA

Received 5 December 2003; Revised 7 September 2004; Accepted 7 September 2004

In this tutorial, traditional decision tree construction and the current state of decision tree modeling are reviewed. Emphasis is placed on techniques that make decision trees well suited to handle the complexities of chemical and biochemical applications. Copyright © 2004 John Wiley & Sons, Ltd.

KEYWORDS: decision tree modeling; pattern recognition; classification; ensemble modeling

1. INTRODUCTION

By its simplest description, decision tree analysis is a divide-and-conquer approach to classification (and regression—not covered within the scope of this review). Decision trees can be used to discover features and extract patterns in large databases that are important for discrimination and predictive modeling. These characteristics, coupled with their intuitive interpretation, are some of the reasons decision trees have been used extensively for both exploratory data analysis and predictive modeling applications for more than two decades. Decision trees have an established foundation in both the machine learning and artificial intelligence literature[1] and are slowly developing a niche in both the chemical and biochemical sciences.

The purpose of this work is to introduce the reader to decision tree modeling for classification with emphasis on adaptations to the traditional decision tree design that make the methodology more useful to chemical and biochemical applications. Section 2 presents an overview of traditional decision tree implementation including a worked analysis of a publicly available microarray dataset. Microarray analysis is just one of many exciting new areas to which decision tree methodologies are being applied. Section 3 discusses the generalization of decision tree modeling to include the directed integration of alternative predictive classification techniques. This section also discusses the extension of decision trees to ensemble classification modeling. Section 4 discusses some alternative uses for decision tree modeling in the chemical and biochemical sciences.

2. TRADITIONAL DECISION TREE IMPLEMENTATION

One advantage that decision tree modeling has over other pattern recognition techniques lies in the interpretability of the constructed model. Owing to this interpretability, information relating to the identification of important features and interclass relationships can be used to support the design of future experiments and data analysis. Microarray data analysis is an example of a biochemical application that is suitable for pattern recognition techniques such as decision trees where the emphasis of the analysis resides in the interpretation and the identification of important biological probes rather than the predictive accuracy of classification (at least initially).

2.1. Graphical representation and terminology

For the following discussion an $m \times n$ response matrix $\mathbf{X}$ contains m samples described by n features. The $m \times 1$ vector $\mathbf{y}$ contains the corresponding categorical property values for the samples in $\mathbf{X}$. The subscripts T and E, in reference to $\mathbf{X}$ and $\mathbf{y}$, indicate the training and evaluation sets respectively.

A publicly available microarray dataset will be used to introduce decision tree methodology. The Whitehead Institute at the Massachusetts Institute of Technology Center for Genome Research has reported the use of microarray gene expression data for discrimination of acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL), two tumor types with similar histopathological appearances[2]. The dataset was produced using Affymetrix microarrays containing 7129 probes for 6817 human genes. $\mathbf{X}_T$ contains 38 bone marrow samples (27 ALL and 11 AML samples). The microarray dataset is available online at http://www.broad.mit.edu/cancer/. The reader is encouraged to obtain the dataset and follow along with the analysis.

Although the following analysis was performed using in-house-developed software programs, decision tree code is available in the Statistics Toolbox for the Matlab® technical computing language (http://www.mathworks.com/) and for the R statistical programming language and environment (http://www.r-project.org/).

To facilitate the introduction to decision tree methodology, 50 of the 7129 probes were selected for analysis here. A stepwise approach incorporating the Wilk's lambda statistic[3] was used to select the probes based on the change in the ratio of the within-class to total-class variances upon the addition of features. The selected probes are provided in the Appendix. Figure 1 illustrates the portion of the microarray dataset that will be used in the methodology discussion. The

*Correspondence to: S. D. Brown, University of Delaware, Newark, DE 19716, USA.
E-mail: sdb@udel.edu
Contract/grant sponsors: Center for Process Analytical Chemistry (CPAC); Defense Advanced Research Projects Agency (DARPA).

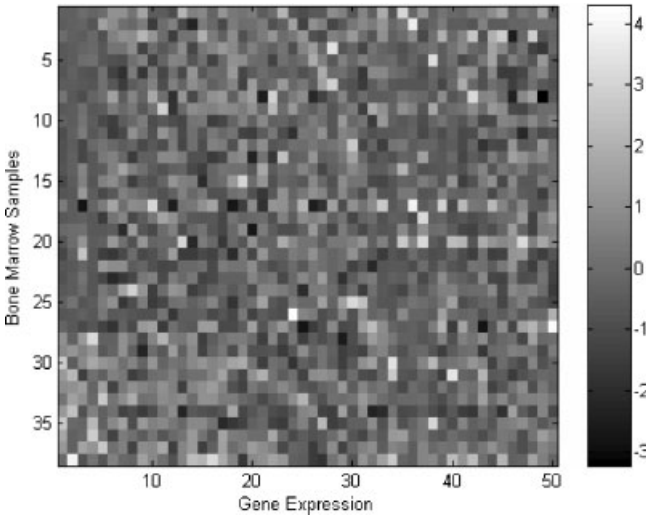

Figure 1. Microarray training set, $\mathbf{X}_T$, used in the decision tree methodology discussion.

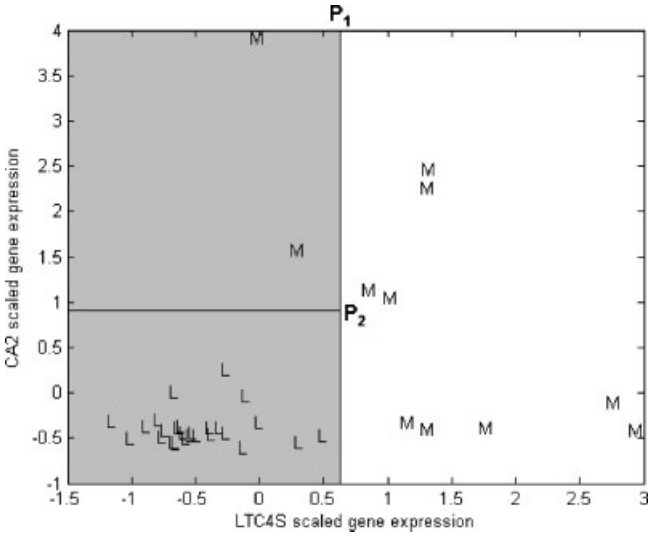

Figure 2. Recursively-partitioned feature space (features 1 and 2) of $\mathbf{X}_T$.

expression values for each gene are autoscaled to improve visualization. The first 27 samples belong to the ALL tumor classification and the last 11 samples belong to the AML classification. The selected genes are provided in the Appendix.

A decision tree will be used to model the microarray dataset. A decision tree is a hierarchical model composed of discriminant functions, or decision rules, that are applied recursively to partition the feature space of a dataset into pure, single class subspaces. The recursively partitioned feature space of the Whitehead dataset is illustrated in Figure 2. ALL and AML tumor samples are denoted by L and M respectively. The figure shows two decision boundaries. The first decision boundary partitions the feature space at **P**1 (the scaled expression of the *leukotriene C4 synthase* (*LTC4S*) gene (feature 1) = 0.62625). Then the second decision boundary partitions the shaded subspace at **P**2 (the scaled expression of the *carbonic anhydrase II* (*CA2*) gene (feature 2) = 0.90977).

The output of decision tree programs, however, is usually limited to the decision tree itself, not the feature space

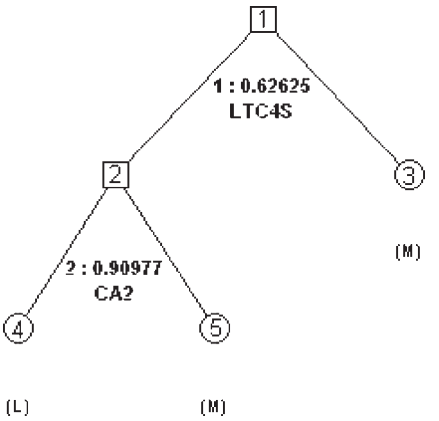

Figure 3. Decision tree corresponding to the partitioned feature space in Figure 2.

representation. The decision tree illustrated by Figure 3, corresponds to the recursively partitioned feature space shown in Figure 2. Using the Matlab decision tree code, a decision tree model similar to that shown below can be generated. First the model is constructed: 'TreeModel = treefit($\mathbf{X}_T$, $\mathbf{y}_T$, 'method', 'classification');'. Then the model is displayed: 'treedisp(TreeModel)'.

Decision trees have two types of nodes: branch nodes and leaf nodes (represented by squares and circles respectively in Figure 3). Node 1 is referred to as the root node and represents the entire feature space. The remaining nodes (nodes 2–5 in this example) each represent a subspace of the original feature space. For example, nodes 2 and 3 represent the disjoint subspaces produced by the first decision boundary (*LTC4S* gene (feature 1) = 0.62625). In Figure 2, nodes 2 and 3 are indicated by the shaded and unshaded regions respectively. Each branch node is the parent to two children nodes (i.e. node 2 is the parent to nodes 4 and 5 in Figure 3).

A decision pathway is represented by a line connecting a parent with one of its children. Decision rules are the inequality expressions that describe the decision boundaries. For example, the line connecting node 1 with node 2 indicates the pathway traveled by a sample if its scaled expression level for the *LTC4S* gene is less than or equal to 0.62625. The line connecting node 1 with node 3 indicates the pathway traveled by a sample if its scaled expression level for the *LTC4S* gene is greater than 0.62625. New samples are directed through the hierarchical model based on the set of decision rules at encountered branch nodes. Upon reaching a leaf node, new samples are classified based on the set of training samples present within the leaf node subspace. Commonly, the class represented by the majority of training samples within the leaf node subspace dictates sample classification. For the microarray example, test samples reaching node 4 would receive an ALL tumor classification (see Figures 2 and 3).

2.2. Growing decision trees

A decision tree is constructed by recursively partitioning the feature space of the training set. The objective is to find a set of decision rules that naturally partition the feature space to provide an informative and robust hierarchical classification model. To help develop a stable foundation for the presentation of more complex decision tree methods, the reader is

first introduced to the univariate partitioning scheme. The Whitehead microarray dataset will be used throughout this section to illustrate the necessary steps.

2.2.1. Possible partition values

A set of possible partition values is determined by calculating the midpoint for each set of consecutive unique responses along each feature (i.e. gene expression). For p unique responses, $p-1$ possible partition values are calculated. A scoring criterion is used to evaluate and compare each of the possible partition values.

2.2.2. Scoring criteria

Several scoring criteria have been developed to evaluate decision tree partitions[4]. To simplify the discussion, this subsection will focus on two scoring criteria called *Information Gain* (*InfoGain*)[5] and *Gini Index* (*Gini*)[6]. The concept of information, *Info*, is described in Equation (1), where N_j is the number of samples belonging to class j, $N(t)$ is the number of samples in node t, and $N_j(t)$ is the number of class j samples in node t.

$$Info = -\sum_j \left(\frac{N_j(t)}{N(t)}\right)\log_2\left(\frac{N_j(t)}{N(t)}\right) \tag{1}$$

The possible partition value that maximizes the 'change in information', called *InfoGain*, is selected. *InfoGain* is calculated in Equation (2), where *Info*(q) is the information of the feature subspace q, and p_k is the proportion of samples passed to the kth subspace.

$$InfoGain = Info(\text{Parent}) - \sum_k (p_k)\, Info(\text{Child}_k) \tag{2}$$

The *Gini Index* measures the reduction in class impurity from partitioning the feature space (see Equations (3) and (4)). For a set of training samples, $p(j)$ is the prior probability that a sample belongs to class j, and $\|\mathbf{g}\|$ represents the normalization of the vector $\mathbf{g}$ to unit length. For the microarray example, proportional priors are used to match the default setting in the Matlab decision tree code, where $p(j) = N_j/N$(1).

$$impurity = 1 - \sum_j \left|\left|p(j)N_j(t)/N_j\right|\right|^2 \tag{3}$$

$$Gini = impurity(\text{Parent}) - \sum_k (p_k)\, impurity(\text{Child}_k) \tag{4}$$

The *InfoGain* and *Gini* scores for the best partition of each of the 50 biological probes are plotted in Figure 4. In this example the best *InfoGain* and *Gini* scores calculated for each feature track each other. In general, different scoring criteria tend to track one another, especially for binary classification systems. Differences in feature selection based on scoring criteria are dependent on the number of samples and the proportion of classes. Even though the absolute values of the two scoring criteria are not directly comparable, both criteria in this example indicate that features 1 and 4 (*LTC4S* and *CD33* respectively) are important (see Figure 4, two features with highest scoring partitions). The *InfoGain* scoring criterion suggests that feature 1 may provide a slightly better partition, but not by much. The *Gini* scoring criterion suggests that features 1 and 4 have the same discriminatory

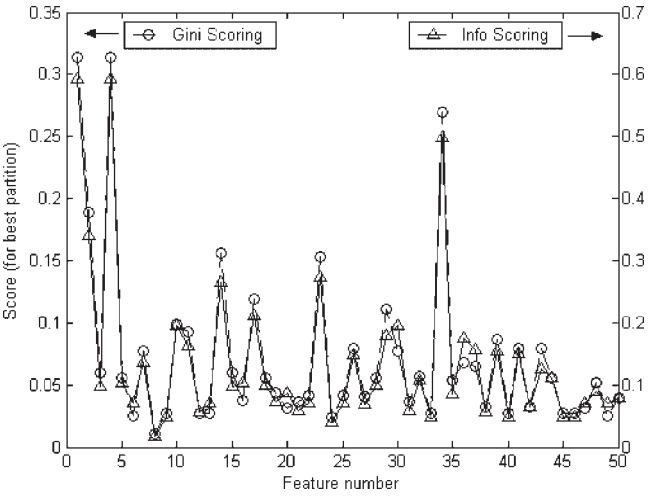

Figure 4. Best *InfoGain* and *Gini* scores for each feature of $\mathbf{X}_T$.

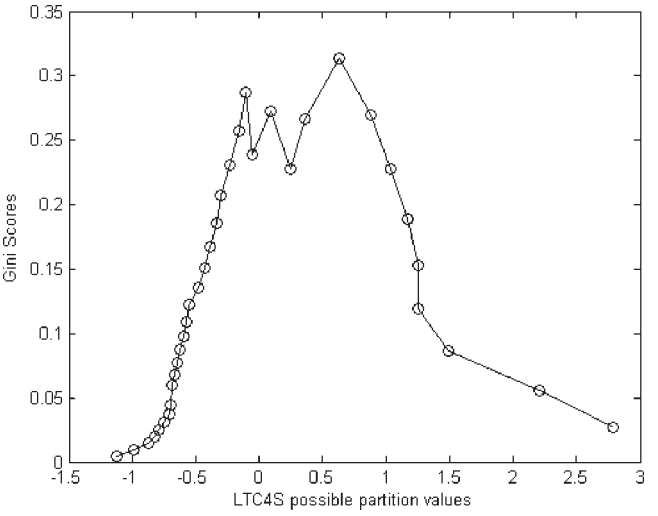

Figure 5. *Gini* scores for the possible univariate partitions of feature 1 (LTC4S gene).

power. When multiple partitions have identical scores and no additional information is available, some decision tree programs, e.g. the Matlab implementation, select the partition on the earliest involved feature (i.e. feature 1 selected before feature 4).

Owing to the similarity between the *InfoGain* and *Gini* scoring results, the *Gini* scoring criterion (with proportional priors) will only be used for the rest of the microarray example. The *LTC4S* gene (feature 1) was determined as optimal based on the *Gini* scoring criterion. For a closer inspection the subspace described by the *LTC4S* gene (see Figure 2) is examined in more detail. The *Gini* scores corresponding to the possible partition values for *LTC4S* are plotted in Figure 5. The maximum score for the *Gini* criterion was 0.3134 for *LTC4S* at 0.62625.

For univariate partitioning techniques it is generally concluded that the choice of scoring criterion (especially in binary classification systems) does not strongly affect the classification accuracy of the overall decision tree model[7]. However, the choice of scoring criterion can affect which features are selected and the complexity (referring to the number of partitions) of the constructed tree. For systems where scoring criteria do not track each other, this means

that different decision tree models may result from the application of different scoring criteria on the same training set. By comparing the differences between decision tree models constructed using different scoring criteria, certain features may be identified as having more or less significance. Feature selection, as with any classification technique, can play a critical role in developing a robust decision tree model, but it can also affect other aspects of the application, such as measurement cost and interpretability. Therefore exploring a variety of scoring criteria and evaluating the resulting, disparate partitioning patterns can be beneficial to an analysis. Some additional resources are available concerning the selection of decision tree partitions[8,9].

After a decision rule has been selected, the feature space is divided into two disjoint subspaces. Then the partitioning procedure is recursively applied to each of the resulting subspaces until all resulting subspaces contain samples belonging to a single class. A decision tree that describes the resulting partitions is the usual output. Figure 3 illustrates the resulting decision tree for the microarray analysis.

2.3. Evaluating decision trees

To evaluate the accuracy of the decision tree model, new samples $\mathbf{X}_E$ are classified, where $\mathbf{X}_E$ consists of 24 bone marrow and 10 peripheral blood samples. A sample is first evaluated by the decision rule at node 1 and is passed along the predicted pathway to the next node. If the sample encounters a branch node, the sample is again evaluated by the corresponding decision rule and is passed along the predicted pathway. The process continues until the sample reaches a leaf node.

For the microarray example the test samples were classified by the previously constructed decision tree model. The *LTC4S* and *CA2* recursively partitioned feature space is plotted in Figure 6. Test samples are shown in bold (**L** and **M**).

From Figure 6, it is clear that several (9/34) of the test samples are misclassified by the decision tree model developed for the microarray dataset. This indicates that either the model has overfit the training set or that the selected partitions are not robust to new samples. The ability to generalize decision tree models is crucial for predictive modeling applications. Next, several approaches to decision tree model generalization are discussed, including pruning, cross-validation and fuzzy partitioning.

2.4. Decision tree generalization (pruning)

Typically, decision trees are grown, at least initially, to overfit the training data by partitioning $\mathbf{X}_T$ into pure, single class subspaces. This means that the decision tree model has probably accounted for variation in the training set that is not generalized or representative of the entire sample population. To reduce overfitting, a second set of samples $\mathbf{X}_E$, independent of $\mathbf{X}_T$, is predicted to evaluate the decision rules. Decision rules that are determined to overfit the data (i.e. decision rules that reduce the predictive accuracy of the decision tree model based on the independent test set) are removed, thereby reducing the complexity of the decision model. When a decision rule is removed, the associated branch node is replaced with a leaf node. This process is referred to as pruning. It is generally accepted that it is better to overgrow a decision tree and then prune back than it is to apply a stopping criterion (i.e. stop partitioning if $N(t) < 5$) to the tree growing phase[6]. The idea is that variation in $\mathbf{X}_T$ that is important to classification may not be modeled by choosing the former approach.

Figure 7 represents the generic relationship between the size of a decision tree (determined as the number of leaf nodes) and its ability to predict on $\mathbf{X}_T$ (broken line) and $\mathbf{X}_E$ (full line). By sequentially pruning branch nodes, the size of the decision tree is reduced. The full and broken lines represent the prediction errors based on the classification of the pruning and training samples respectively. The process of choosing the optimal model complexity for a decision tree is analogous to that of optimizing a multivariate soft-modeled calibration. If $\mathbf{X}_E$ is large and representative of future samples, the tree that minimizes classification error (instead of a root mean squared error) of $\mathbf{X}_E$ is usually selected. Comparative analyses of pruning techniques on several non-chemical datasets are available[10,11].

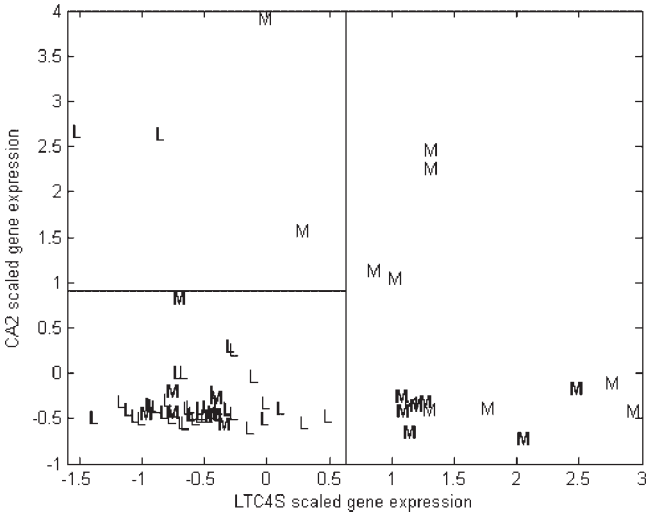

Figure 6. Recursively-partitioned feature space (features 1 and 2) of $\mathbf{X}_T$ with test samples in bold.

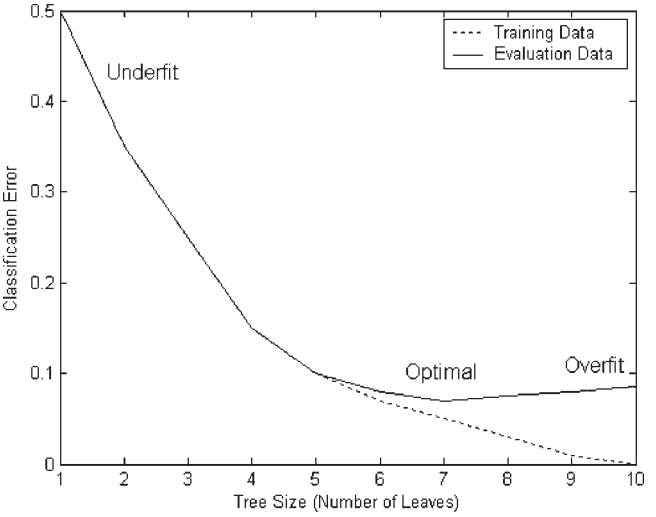

Figure 7. General relationship between prediction error and tree-size (number of leaves).

For many applications an insufficient number of samples exist to construct (train and prune) a robust decision tree model, including the microarray example explored above. Although the decision tree model for the microarray example accounted for all of the variation in $\mathbf{X}_T$, the model was not robust to the variation in $\mathbf{X}_E$. Non-robust classification is not surprising owing to the small sample-to-feature ratio in this dataset. For these circumstances, when an insufficient number of samples is available to adequately train and prune a decision tree model, other approaches can be used to develop robust classification models. A cross-validation approach seems to be an obvious choice. Cross-validation procedures, however, are more appropriately applied to stable (in terms of model estimation) pattern recognition techniques such as linear discriminant analysis. The problem with cross-validating decision tree models, in the classical sense, is that with each subsampling of the $\mathbf{X}_T$ population a different decision tree model may result. Determining the complexity of the final decision tree model based on the cross-validation errors is awkward, since the order in which decision rules are pruned plays a critical role and the stability of the decision rules is uncertain. Owing to its non-parametric nature, the univariate decision rule is inherently prone to instability[12].

One cross-validation procedure, called minimal cost–complexity pruning[6], has been developed specifically for decision trees. Cost–complexity pruning balances the trade-off between predictive accuracy and tree size based on the complexity parameter α, which is optimized using cross-validation. The complexity parameter is then used at each node to determine which decision rules should be pruned. In the microarray example, however, this type of approach would not be appropriate because there are so few decision rules. For these circumstances, where automated pruning techniques are not applicable, the analyst may choose to prune the decision tree model manually. Manual pruning involves removing decision rules based on some developed understanding of the system in question (i.e. the quality of the selected features). Although rule generalization techniques such as pruning may reduce overfitting in decision tree models, they do not affect the quality of the decision rules that are initially selected.

2.5. Decision rule generalization (fuzzy partitioning)

Recently, a new approach to decision tree modeling, called fuzzy partitioning, has been introduced. Fuzzy partitioning discovers robust decision rules[13] within the data by modeling the inherent instability of possible partitions. The authors suggested a possible implementation to generate a fuzzy partition: (1) randomly select a portion of the training set, $\mathbf{X}_S$; (2) determine the best decision rule for $\mathbf{X}_S$ and store the partition information (feature and value); (3) repeat steps (1) and (2) for I iterations; (4) cluster the selected partitions along each feature into splitting regions; (5) determine the most selected splitting region in the training set based on the collected decision rule information.

Fuzzy partitioning, as described above, was applied to the microarray dataset. Figure 8 shows the frequency of probe selection (the proportion of times a decision rule generated from $\mathbf{X}_S$ involved a given probe).

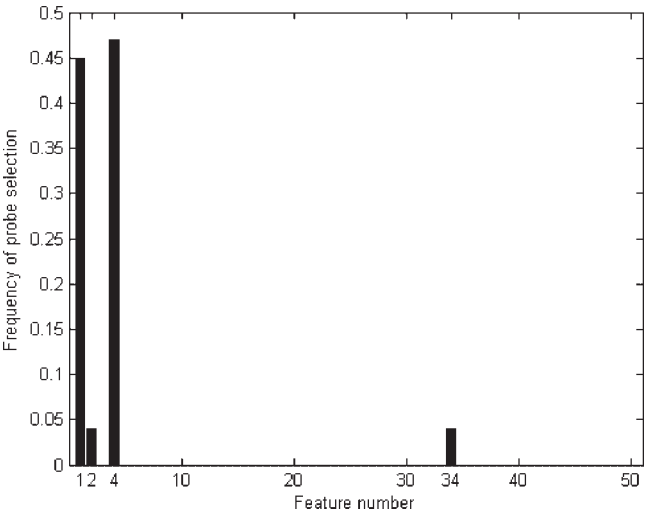

Figure 8. Frequency of probe selection based on fuzzy partitioning.

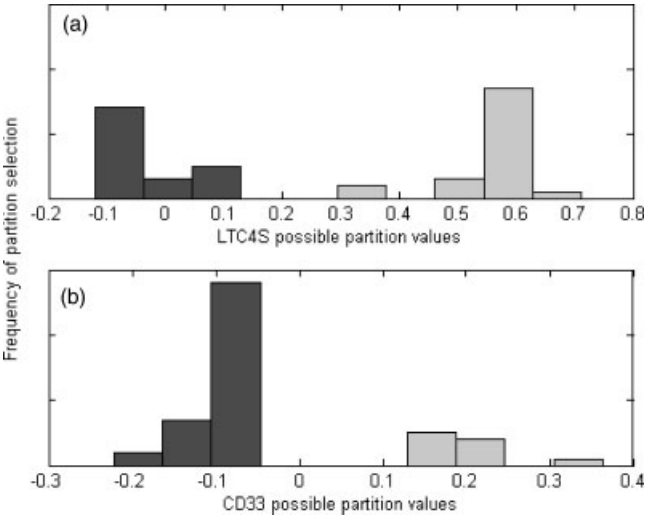

Figure 9. Frequency of partition selection along feature 1 (*LTC4S gene*) and feature 4 (*CD33 gene*) based on fuzzy partitioning.

Fuzzy partitioning identified two dominant biological probes (*LTC4S* (feature 1) and *CD33* (feature 4)) as possible sites for the primary partition (see Figure 8, two features with highest frequency of probe selection). In Figures 9(a) and 9(b), the histograms indicate the frequency of selection for the possible partition values (based on *InfoGain*) for *LTC4S* and *CD33* respectively. Both the *LTC4S* gene and the *CD33* gene have two distinct local splitting regions. Primary and secondary splitting regions are indicated by dark and light shading respectively. The primary splitting region for the *CD33* probe (dark shading) was selected more frequently than any other local splitting region.

A discrete decision rule (scale expression value for $CD33 = -0.055$) was extracted from the distribution illustrated in Figure 9(b) (dark shading). The decision rule partition value was calculated as the mean of the selected partition values within the dark-shaded region. Figure 10 illustrates the discrete decision rule and the resulting partitioned feature space (test samples are included in bold). The new decision rule, based on fuzzy partitioning, misclassifies

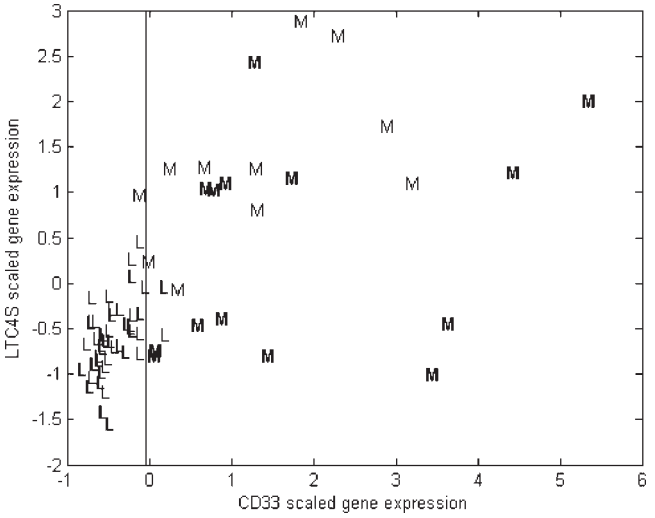

Figure 10. Partitioned feature space (feature 4) of $\mathbf{X}_T$ based on fuzzy partitioning with test samples in bold.

three training samples (3/38) and one test sample (1/34). In contrast with pruning, fuzzy partitioning was used to obtain a more robust decision rule.

3. DECISION TREES FOR PREDICTIVE CLASSIFICATION MODELING

3.1. Ensemble modeling techniques

Ensemble methods[14–17] have been developed that can be used in conjunction with decision trees (but not exclusively) to improve generalization. These methods combine the predictions from multiple classification models constructed from different representations of the $\mathbf{X}_T$. Generally speaking, ensemble modeling negatively affects model interpretation. These methods are typically used when improvements to the accuracy of a predictive model are more important to the analysis than retention of model interpretation. Ensemble modeling is a form of high-level (or decision-level) data fusion and is a current trend in machine learning research. Two of the more popular ensemble techniques are addressed below.

Bagging[18], or bootstrap aggregating, is probably the simplest ensemble method. In bagging, one creates a series of N models, each based on a different bootstrap[19] sampling of the $\mathbf{X}_T$. Each bootstrap sample is created by randomly selecting samples from $\mathbf{X}_T$ with replacement (note that the same sample can be selected more than once). The outputs (predictions) of the N individual models are combined using a voting technique. Each prediction corresponds to a vote for a specific class and the class that receives the most votes is selected. Bagging is generally appropriate for unstable classifiers such as decision trees (see Section 2.4). Similarly to the fuzzy technique described above, bagging is dependent on perturbed versions of $\mathbf{X}_T$ to produce unstable (different) predictions.

Boosting[20], another popular ensemble method, is based on the sequential construction of N models. Boosting utilizes a sample reweighting function to adaptively alter $\mathbf{X}_T$ for each successive model based on the accuracy of the preceding model. A sample that is predicted poorly by the nth model is 'boosted' by increasing its weight before the construction of the $(n+1)$th model. Each successive model focuses more on poorly predicted samples to improve generalization. To classify a new sample, the sample is predicted by each of the N individual models. The prediction made by each model is weighted based on the accuracy of the nth model. The outputs of N individual models are combined using a voting technique.

3.2. Generalizing feature space partitioning

One approach to successful predictive classification is matching the appropriate modeling technique with the target system, since, for any given system, different inter- and intra-class relationships will inevitably exist. The traditional decision tree implementation is not particularly suited to handle high-dimensional datasets for predictive classification modeling, unlike other established chemometric techniques. Pattern recognition techniques such as soft independent modeling of class analogies (SIMCA)[21] and partial least squares discriminant analysis (PLS-DA)[22,23], have been developed to handle underdetermined systems. These techniques depend on feature transformation and reduction to simplify the feature space and subsequent classification. Fortunately, when it becomes necessary to use more complex discriminant techniques (even those that integrate feature transformation techniques), the decision tree structure can be adapted.

The definition of a decision tree partition can be generalized as any discriminant that separates two sets of samples. Equations (5)–(7) represent a generalized view of the individual decision tree processes.

$$\mathbf{X}_T,\ \mathbf{y}_T^{\text{true}} \xrightarrow{\text{learn}} \textit{Model} \tag{5}$$

$$\mathbf{X}_T,\ \textit{Model} \xrightarrow{\text{predict}} \mathbf{y}_T^{\text{pred}} \tag{6}$$

$$\mathbf{X}_E,\ \textit{Model} \xrightarrow{\text{predict}} \mathbf{y}_E^{\text{pred}} \tag{7}$$

The superscript labels 'true' and 'pred' refer to the known and predicted property values respectively. In Equation (5) a predictive classification model, *Model*, is learned from an input matrix and known property vector $\mathbf{y}_T^{\text{true}}$. For hierarchical combined predictive models such as decision trees, training samples are passed from model to model. Equation (6) determines the pathway assignments for the training samples $\mathbf{y}_T^{\text{pred}}$ for successive model construction. In Equation (7), evaluation samples are classified by the *Model* producing $\mathbf{y}_E^{\text{pred}}$, establishing the pathway assignments of new samples leading to classification.

Seen in this way, a decision tree may be better viewed as a hierarchical platform for the directed integration of predictive classification models. There are several examples of discriminant techniques in the machine learning, chemometrics and statistics literature that are suitable for decision tree partitioning. However, the purpose for the rest of Section 3 is not to critically evaluate the integration of every possible partitioning technique, but to investigate the issues associated with their implementation. Critical issues such as feature selection, feature transformation, property vector encoding and partition scoring will all be addressed.

3.3. Feature selection and transformation

Two attractions of traditional, univariate decision tree modeling, as discussed before, are its inherent ability to select important features for classification and the interpretability of the resulting models. When multivariate partitioning techniques are employed (i.e. quadratic discriminant analysis), the process of feature selection is no longer inherent and needs to be addressed externally. There are several external approaches to the selection of informative features, including forward feature selection, backward feature elimination, genetic algorithms and variance-based techniques. Although commonly used with other pattern recognition techniques, these approaches are equally applicable within the decision tree structure (as seen in the determination of multivariate linear partitions [24,25]). Additionally, with the use of multivariate partitioning techniques the interpretability of the resulting model becomes more difficult. One goal of feature selection in multivariate decision tree analysis is the balance between predictive accuracy and model complexity for applications where model interpretation is still desired.

Standard approaches to feature selection may not be sufficient or appropriate for system domains that are described by high-dimensional correlated feature spaces, such as those found in chemical and biochemical applications. To handle the complexity of high-dimensional correlated feature spaces, feature transformation and reduction techniques such as principal component analysis (PCA) [26] and partial least squares (PLS) decomposition [27] can be used in tandem with predictive modeling techniques [28]. Coupling feature transformation techniques with decision tree modeling has not been fully developed and is only sparsely commented on in the literature [29]. In the context of decision tree modeling, there are two approaches to the integration of feature transformation techniques. These two approaches, described below as passive and active, allow for the modeling of high-dimensional correlated feature spaces within the decision tree architecture.

The passive feature space transformation is the easier implementation of the two approaches. To implement a passive feature transformation: (1) $\mathbf{X}_T$ is transformed via PCA or PLS, where $\mathbf{tX}_T$ represents the transformed feature space of the training set (i.e. principal components or latent variables); (2) next the decision tree is constructed from $\mathbf{tX}_T$; (3) then $\mathbf{X}_E$ (pruning or test set) is projected using the loadings or weights determined from the $\mathbf{X}_T$ transformation, where $\mathbf{tX}_E$ represents the transformed feature space of the evaluation set. To integrate passive transformation with existing decision tree software, $\mathbf{X}_T$ and $\mathbf{X}_E$ can be transformed beforehand, without any necessary alterations to the existing software. The passive transformation, although simple, has the restriction that only a single set of original features can be transformed before each decision tree analysis.

To implement active feature transformation, on the other hand: (1) the original feature representation of $\mathbf{X}_T$ at node t, $\mathbf{X}_T(t)$, is transformed via PCA or PLS, where $\mathbf{tX}_T(t)$ represents the original feature representation of the training set at node t; (2) a decision rule is constructed from $\mathbf{tX}_T(t)$ and the training samples are passed to the children of node t; (3) steps (1) and (2) are repeated until the entire decision tree is constructed; (4) then $\mathbf{X}_E(t)$ (pruning or test set) is projected using the local loadings or weights determined from the $\mathbf{X}_T(t)$ transformation, where $\mathbf{tX}_E(t)$ represents the transformed original feature space of the evaluation set at node t; (5) $\mathbf{tX}_E(t)$ samples are passed to the children of node t based on the local decision rule; (6) steps (4) and (5) are repeated until the samples of $\mathbf{X}_E$ are classified. In step (1), $\mathbf{X}_T(t)$ is transformed only when it is appropriate (i.e. sufficient samples, reasonable proportion of two or more classes). Active feature transformation may allow for local sources of variation to be modeled in different parts of the decision tree. The main difficulty in implementing the active transformation is the integration of feature transformation techniques into existing decision tree software, which may be impossible or impractical depending on the software. Active transformation does, however, allow for feature selection at each node prior to transformation.

3.4. Property vector encoding

When certain multivariate techniques are used to create binary partitions in multiclass (where j, the number of classes, is greater than two) systems, the issue of property vector encoding needs to be addressed. Property vector encoding refers to the representation of the property vector. A multigroup property vector can be represented by multiple, binary vector encodings, including one-versus-all, pairwise [30] and unsupervised clustering [8]. The one-versus-all design defines j binary vector encodings. For each binary representation, one class, encoded as zeros, is paired against all $j-1$ other classes, jointly encoded as ones. The one-versus-all approach is useful in determining which classes (if any) separate the easiest. The pairwise design defines $j(j-1)/2$ encodings. For each binary vector representation, one class, encoded as zeros, is paired against one other class, encoded as ones, and the remaining $j-2$ classes are ignored. The pairwise design is useful in decomposing a many-class system into more manageable binary decision tasks. Additionally, the pairwise design has the potential to help locate useful partitions that might otherwise be obscured by the additional classes. The unsupervised clustering approach can be used to develop discriminant rules for naturally clustering sets of objects. For this approach a clustering algorithm (i.e. k-means clustering [31]) is used to divide the system into two sets of samples, which are then encoded by zeros and ones.

When various feature selection techniques, feature transformation techniques and property vector encodings are explored, several competing models can be generated. One approach to evaluate competing models (decision rules) is to use traditional decision tree scoring criteria such as *InfoGain*. *InfoGain*, like most scoring criteria, is dependent on the property vector encoding (see Equations (1) and (2)). To calculate the scoring criterion in a multiclass system, one has the choice to use the original encoding $\mathbf{y}_T$ or the binary encoding implemented in model construction. For the multiclass example, scores for different types of decision rules are only comparable when they are determined using the same property vector encoding.

The thyroid dataset [32] (http://www.ics.uci.edu/~mlearn/MLRepository.html) will be used to illustrate some of the issues surrounding property vector encoding

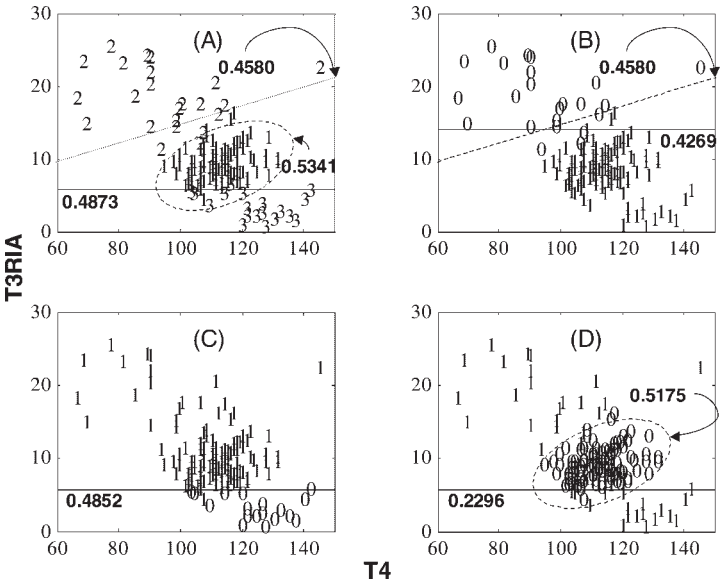

Figure 11. For the T3RIA and T3 feature space: A) Original class encoding; B) Class 2 versus all encoding; C) Class 3 versus all encoding; D) Class 1 versus all encoding.

and the comparison between various partitioning techniques. The thyroid dataset contains 215 samples representing various patients that are classified into one of three categories (euthyroid, hyperthyroid and hypothyroid functional states) defined by their thyroid activity. There are five features describing laboratory test results relating to thyroid function: total thyroid activity (T4), total serum triiodothyronine (T3RIA), T3-resin uptake (RT3U), basal thyroid simulating hormone (TSH) and the maximal absolute difference of TSH after injection (ΔTSH). The goal of this dataset is to predict thyroid functional state from these test results. (*Note*. For the example below, every third sample of the thyroid dataset was removed. The training set consisted of the remaining samples.) Figure 11 shows the original multigroup encoding (Figure 11(a)) and each of the g one-versus-all encodings (Figures 11(b)–11(d)) for the same feature space (T3RIA vs T4). In Figure 11(a) the euthyroid, hyperthyroid and hypothyroid functional states are denoted 1, 2 and 3 respectively.

In Figure 11(a) the horizontal line represents the best partition calculated using the *InfoGain* scoring criterion for the multigroup representation. In Figure 11(c) the horizontal line represents the best partition calculated using the *InfoGain* scoring criterion for the hypothyroid-versus-all binary representation. The decision rules in Figures 11(a) and 11(c) are identical; however, given the different property vector encodings, the *InfoGain* scores are different.

In Figure 11(b) the oblique decision rule represents the best multivariate, linear partition calculated using the *InfoGain* scoring criterion for the hyperthyroid-versus-all binary representation. For the hyperthyroid-versus-all binary representation the multivariate, linear partition (*InfoGain* = 0.4580) outscores the best univariate partition (*InfoGain* = 0.4269). This benefit, however, does not translate into the multigroup representation. In Figure 11(a) the oblique decision rule represents the best multivariate, linear partition calculated using the *InfoGain* scoring criterion for the hyperthyroid-versus-all binary representation. For the multigroup representation the multivariate, linear partition (*InfoGain* = 0.4580) does not outscore the best univariate partition (*InfoGain* = 0.4873).

In Figure 11(d) the broken decision rule represents the best multivariate, quadratic partition calculated using the *InfoGain* scoring criterion for the euthyroid-versus-all binary representation. For the euthyroid-versus-all binary representation the multivariate, quadratic partition (*InfoGain* = 0.5175) outscores the best univariate partition (*InfoGain* = 0.2296). The benefit in this case translates into the multigroup representation. In Figure 11(a) the broken decision rule represents the best multivariate, quadratic partition calculated using the *InfoGain* scoring criterion for the euthyroid-versus-all binary representation. For the multigroup representation the multivariate, quadratic partition (*InfoGain* = 0.5341) outscores the best univariate partition (*InfoGain* = 0.4873). In this example, based solely on *InfoGain* scoring (neglecting the issue of interpretation), the quadratic discriminant would be chosen to partition the feature space.

3.5. Alternative extensions and uses of decision tree modeling

Traditionally, decision trees use a majority vote classifier at each leaf node to produce the final predictions. A majority vote classifier returns the class label associated with the most represented class within the leaf. However, any predictive classification technique may be integrated at leaf nodes to model the local subspace environment for sample classification. Alternative leaf node classification augments the decision tree model by attempting to model small portions of the ungeneralized feature space. Alternative leaf node classifiers can improve classification accuracy and reduce the size of decision tree models [33,34].

Adding alternative leaf node classifiers to decision tree models can be easy. The only information necessary to construct a leaf node classification model is knowledge of

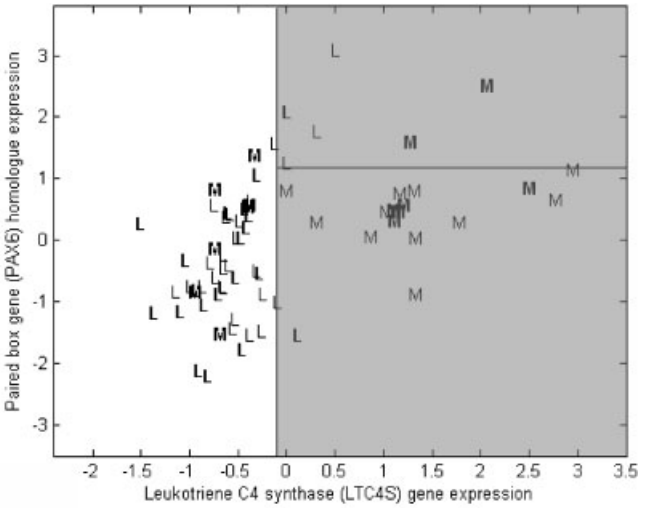

Figure 12. Recursively-partitioned feature space (features 1 and 10) of $\mathbf{X}_T$ with test samples in bold.

which training samples are present in the local subspace. A more integrated approach to the inclusion of leaf node classifiers, however, would involve linking their use into existing pruning routines. Regardless of the level of integration, the selection of appropriate leaf node classification techniques can be the more difficult task. The local subspace characteristics, e.g. dimensionality, distribution of class membership and overall sample size, will play a major role in selecting an appropriate alternative leaf node classifier. Because of these characteristics (especially the reduced sample size), overfitting within the leaf nodes of a decision tree model is very problematic.

To illustrate alternative leaf node classification, the microarray dataset will be revisited. The subspaces resulting from the first univariate decision rule (as determined by the *InfoGain* scoring criterion: scaled expression for the *LTC4S* gene $= -0.1085$) will be explored. For this example it is assumed that the second decision rule (as determined by the *InfoGain* scoring criterion: scaled expression for the *paired box* (*PAX6*) gene homologue (feature 10) $= 1.186$) has been removed via a pruning routine. The subspace described by node 2 (*LTC4S* gene ≤ -0.1085) is pure, containing only 'M' training samples (see Figure 12, unshaded region). Test samples are shown in bold. Alternative leaf node classifiers are of no benefit to pure subspaces. The subspace described by node 2 (*LTC4S* gene > -0.1085) is not pure, containing both 'L' (three) and 'M' (11) training samples, where the values in parentheses indicate the number of samples in the previously labeled class (see Figure 12, shaded region).

In this example the *k*-nearest neighbor (KNN) classifier is explored as an alternative to majority vote classification. The KNN classifier is probably the easiest and most appropriate alternative leaf node classifier for the majority of decision tree subspaces, since most classes are under-represented (if they are represented at all) within decision tree leaf nodes. Some issues associated with KNN classification are feature selection, feature scaling, distance metric selection and choosing the number of nearest neighbors. Owing to the minimal 'L' class representation, cross-validated feature selection is not an option. In this example the single feature (*LTC4S*) used in the decision rule is also used for KNN leaf node classification. The Euclidean distance metric was selected because of the small number of 'L' class training samples. The nearest neighbor ($k = 1$) to each of the nine test samples reaching node 3 is determined. All nine samples are correctly classified based on KNN leaf node classification. Nine samples (seven class 'M' and two class 'L' samples), of course, do not represent a large enough test set to validate KNN (arguably the simplest classifier) as an alternative leaf node classifier. These nine samples just demonstrate that an alternative classification technique can be applied to an impure leaf node subspace.

Decision tree modeling can also be effectively combined with other predictive classification modeling techniques. Generally, this involves the use of decision trees on the front-end of a hybrid or fused classification model, where the purpose of the decision tree is to reduce the representation of the training data (feature selection or discretization) to accommodate a second classification technique. In the first example, decision trees are used to discover informative features, or 'emerging patterns', in microarray data [35]. In this example, emerging patterns can be thought of as the combination of two consecutive decision tree partitions. Discriminant analysis was then applied to the reduced feature space defined by the emerging patterns. This hybrid classification technique was applied to colon and leukemia cancer experiments.

For the purposes of illustration, the 'emerging patterns' idea is applied to the microarray example. Using the *InfoGain* scoring criterion (in contrast with the *Gini* scoring criterion with proportional priors), the *LTC4S* and *PAX6* (feature 10) genes were selected. Then multivariate, linear and quadratic discriminant models were developed on the two selected features. Figure 13 illustrates the multivariate, linear and quadratic discriminants (broken lines) compared with the decision tree-selected partitions (full lines). In this application a decision tree was used to identify important features, not for predictive modeling.

In another example of a hybrid classification technique, decision trees were coupled with Bayesian networks [37].

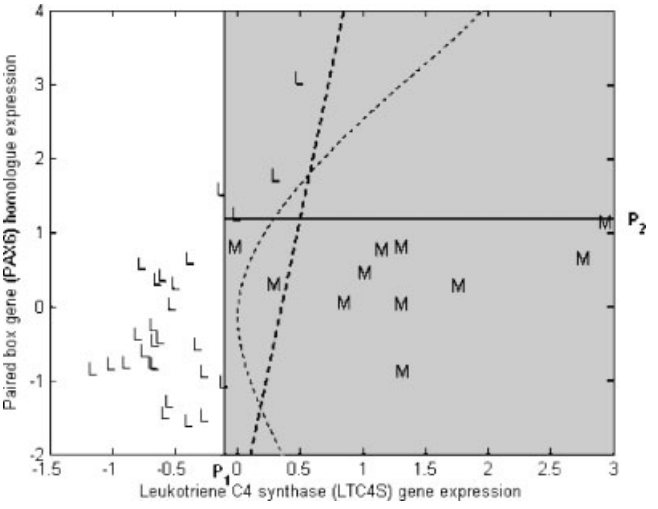

Figure 13. Recursively-partitioned feature space (features 1 and 10) of $\mathbf{X}_T$ with overlaid linear (dashed) and quadratic (dotted) discriminants based on decision tree-selected features.

For this hybrid classification approach a decision tree is constructed from $\mathbf{X}_T$. The decision tree-selected features are discretized based on the generated decision rules. The resulting simplified feature space representation (discretized feature space) is then used to train a Bayesian network predictive classification model. The hybrid 'decision tree–Bayesian network' modeling approach was applied to the thyroid gland dataset explored above, but it could be applied to a variety of systems. This is another application where decision trees were used to identify important features. The only difference here between the 'decision tree–Bayesian network' approach and the 'emerging patterns' technique is that the decision tree partitions were used as thresholds to label discrete regions within the feature space, simplifying the continuously valued features. These types of applications reinforce the idea that decision trees play multiple roles in pattern recognition.

Decision trees can also be used on the back-end of hybrid or fused classification models, although these types of decision tree implementations are not as common as the front-end hybrid designs discussed earlier. The purpose of the decision tree on the back-end of a hybrid classification technique is to model an alternative representation of $\mathbf{X}_T$ produced by the front-end classifier. For example, decision trees can be used to model the predictions made by a set of classifiers [38,39], representing a form of high-level data fusion. The predictions of N classifiers on m training samples are encoded in a new $\mathbf{X}_T$ representation. Prior to this point the explored features have been continuously valued (i.e. expression level, total thyroid activity) or at least discretely valued (i.e. 'decision tree–Bayesian network' modeling approach). Although not explicitly stated, decision trees are capable of modeling categorical features too [6]. The ability to model continuously valued and categorically valued features is an attribute not shared by many other pattern recognition techniques. This means that decision trees can be used to model multiclass predictions from a variety of diverse classifiers (a categorical representation of $\mathbf{X}_T$).

Decision trees can also be adapted for use in regression applications (regression trees), where the property vector $\mathbf{y}_T$ is continuously valued (e.g. biological activity). For regression applications, heterogeneous feature spaces are recursively partitioned into more homogeneous subspaces upon which regression techniques (or simple estimators) can be more appropriately applied. A regression tree partitions a feature space based on a change in the sum of squared deviations of $\mathbf{y}_T$, behaving in a similar manner to a hierarchical clustering algorithm. Although regression trees are more common in other scientific fields, regression trees have potential in chemical or biochemical systems. Regression trees can be applied to quantitative structure–activity relationship (QSAR) modeling applications [40]. For QSAR applications, regression trees could be used to identify important structural or property descriptors for the prediction of biological activity for a set of diverse compounds. Decision trees (classification or regression) could also be used indirectly, via rule discovery, to design compound libraries from large databases [41].

4. CONCLUSION

This review provides a fundamental understanding of decision trees and related methodologies as applied to chemical and biochemical systems. Additional resources are available from the machine learning and artificial intelligence communities that present alternative philosophies of decision tree construction [42, 43, 44]. Interested readers are encouraged to investigate the application of decision trees to chemical and biochemical systems.

Acknowledgements

This research is funded by the Center for Process Analytical Chemistry (CPAC) and by the Defense Advanced Research Projects Agency (DARPA).

APPENDIX: SELECTED PROBES FROM THE WHITEHEAD INSTITUTE MICROARRAY DATASET

1. Leukotriene C4 synthase (LTC4S) gene
2. CA2 Carbonic anhydrase II
3. ALPHA-AMYLASE 2B PRECURSOR
4. CD33 CD33 antigen (differentiation antigen)
5. ERF-2 mRNA
6. BETA-2-MICROGLOBULIN PRECURSOR
7. Na,K-ATPase subunit alpha 2 (ATP1A2) gene
8. GB DEF = CD202 protein
9. C1QB Complement component 1, q subcomponent, beta polypeptide
10. Paired box gene (PAX6) homologue
11. Actin bundling protein mRNA
12. HBG2 Hemoglobin gamma-G
13. HLA-DNA Major histocompatibility complex, class II, DN alpha
14. 3-HYDROXYANTHRANILATE 3,4-DIOXYGENASE
15. GUCA2 Guanylate cyclase activator 2 (guanylin, intestinal, heat-stable)
16. Homeodomain protein HOXB13 mRNA
17. GB DEF = HMGI-C chimeric transcript mRNA, partial cds
18. WD repeat protein HAN11 mRNA
19. GZMK Granzyme K (serine protease, granzyme 3)
20. Heat shock transcription factor 4
21. Tryptophan oxygenase (TDO) mRNA
22. GB DEF = Hs-TBX2 = T-box gene {T-box region} [human, fetal kidney, mRNA Partial, 283 nt]
23. KIAA0033 gene, partial cds
24. GB DEF = H2A/g gene
25. GB DEF = T(3;5)(q25.1;p34) fusion gene NPM-MLF1 mRNA
26. CYTOCHROME P450 XVIIA1
27. TYRO3 Receptor protein-tyrosine kinase sky
28. ISL1 ISL1 transcription factor, LIM/homeodomain, (islet-1)
29. GKP2 Glycerol kinase 2 (testis specific)
30. SOD1 Superoxide dismutase 1 (Cu/Zn)
31. TIE Tyrosine kinase with immunoglobulin and epidermal growth factor homology domains
32. SELE Selectin E (endothelial adhesion molecule 1)
33. SLC15A2 Solute carrier family 15 (H + /peptide transporter), member 2
34. Lysozyme gene (EC 3.2.1.17)

35. CDW52 CDW52 antigen (CAMPATH-1 antigen)
36. Transcription Factor Iiia
37. HLA CLASS II HISTOCOMPATIBILITY ANTIGEN, DQ(1) BETA CHAIN PRECURSOR
38. GB DEF = Annexin II, 5′UTR (sequence from the 5′cap to the start codon)
39. MAGE-9 antigen (MAGE9) gene
40. Interferon regulatory factor 7 (humirf7) mRNA
41. C-erbA-1 mRNA for thyroid hormone receptor alpha
42. Cocaine and amphetamine regulated transcript CART (hCART) mRNA
43. Recepin mRNA
44. GB DEF = Retinoblastoma susceptibility protein (RB1) I66D 4 bp deletion mutant (resulting in premature stop at amino acid 134) gene, exon 4 (L11910 bases 41867–42075)
45. Integrin, Beta 3 Subunit
46. Fetal beta-MHC binding factor
47. Unknown protein mRNA within the p53 intron 1
48. IL1A Interleukin 1, alpha
49. RDX Radixin
50. Gag 2 protein from Human endogenous retrovirus HERV-K10./ntype = DNA/annot = CDS

REFERENCES

1. Russell S, Norvig P. *Artificial Intelligence. A Modern Approach*. Prentice-Hall: Englewood Cliffs, NJ, 1995.
2. Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA, Bloomfield CD, Lander ES. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* 1999; **286**: 531–537.
3. McLachlan GJ. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley: New York, 1992.
4. Mingers J. An empirical comparison of selection measures for decision-tree induction. *Mach. Learn. 1989:* **3**: 319–342.
5. Quinlan JR. *C4.5 Programs for Machine Learning*. Morgan Kaufmann: San Mateo, CA, 1993.
6. Breiman L, Friedman JH, Olshen RA, Stone CJ. *Classification and Regression Trees*. Wadsworth: Belmont, CA, 1984.
7. Breiman L. Technical note: some properties of splitting criteria. *Mach. Learn.* 1996; **24**: 41–47.
8. Loh WY, Shih YS. Split selection methods for classification trees. *Statist. Sinica* 1997; **7**: 815–840.
9. Buntine W. A further comparison of splitting rules for decision tree induction. *Mach. Learn.* 1992; **8**: 75–85.
10. Esposito F, Malerba D, Semeraro G. A comparative analysis of methods for pruning decision trees. *IEEE Trans. Pattern Anal. Mach. Intell.* 1997; **19**: 476–491.
11. Mingers J. An empirical comparison of pruning methods for decision tree induction. *Mach. Learn.* 1989; **4**: 227–243.
12. Li RH. Instability of decision tree classification algorithms Phd Thesis. University of Illinois (Urbana-Champaign), 1992.
13. Myles AJ, Brown SD. Induction of decision trees using fuzzy partitioning. *J. Chemometrics* 2003; **17**: 531–536.
14. Bauer E, Kohavi R. An empirical comparison of voting classification algorithms: bagging, boosting, and variants. *Mach. Learn.* 1999; **36**: 105–139.
15. Dietterich TG. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Mach. Learn.* 2000; **40**: 139–157.
16. Opitz D, Maclin R. Popular ensemble methods: a survey. *J. AI Res.* 1999; **11**: 169–198.
17. Quinlan JR. Bagging, boosting and C4.5. *Proc. 13th Natl Conf. on AI*, 1996; 725–730.
18. Breiman L. Bagging predictors. *Mach. Learn.* 1996; **24**: 123–140.
19. Efron B, Tibshirani RJ. *An Introduction to the Bootstrap.* Chapman and Hall: New York, 1993.
20. Freund Y, Schapire R. A short introduction to boosting. *J. Jpn. Soc. AI* 1999; **14**: 771–780.
21. Wold S. Pattern recognition by means of disjoint principal component models. *Pattern Recogn.* 1976; **8**: 127–139.
22. Vong R, Geladi P, Wold S, Esbensen K. Source contributions to ambient aerosol calculated by discriminant partial least squares regression (PLS). *J. Chemometrics* 1988; **2**: 281–296.
23. Barker M, Rayens W. Partial least squares for discrimination. *J. Chemometrics* 2003; **17**: 166–173.
24. Murthy SK, Kasif S, Salzberg S. A system for induction of oblique decision trees. *J. AI Res.* 1994; **2**: 1–32.
25. Brodley CE, Utgoff PE. Multivariate decision trees. *Mach. Learn.* 1995; **19**: 45–77.
26. Malinowski ER. *Factor Analysis in Chemistry*. Wiley: New York, 2002.
27. De Jong S. SIMPLS: an alternative approach to partial least squares regression. *Chemometrics Intell. Lab. Syst.* 1993; **18**: 251–263.
28. Higdon R, Foster NL, Koeppe RA, DeCarli CS, Jagust WJ, Clark CM, Barbas NR, Arnold SE, Turner RS, Heidebrink JL, Minoshima S. A comparison of classification methods for differentiating fronto-temporal dementia from Alzheimer's disease using FDG-PET imaging. *Statist. Med.* 2004; **23**: 315–326.
29. Yeh CH, Spiegelman CH. Partial least squares and classification and regression trees. *Chemometrics Intell. Lab. Syst.* 1994; **22**: 17–23.
30. Furnkranz J. Round robin classification. *J. ML Res.* 2002; **2**: 721–747.
31. Hartigan JA. *Clustering Algorithms*. Wiley: New York, 1975.
32. Coomans D, Broeckaert I, Jonckheer M, Massart DL. Comparison of multivariate discrimination techniques for clinical data—application to the thyroid functional state. *Methods Info. Med.* 1983; **22**: 93–101.
33. Seewald AK, Petrak J, Widmer G. Hybrid decision tree learners with alternative leaf classifiers: an empirical study. *Proc. 14th Int. FLAIRS Conf.*, 2001.
34. Kohavi R. Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid. *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining*, 1996; 202–207.
35. Boulesteix AL, Tutz G, Strimmer K. A CART-based approach to discover emerging patterns in microarray data. *Bioinformatics* 2003; **19**: 2465–2472.
36. Fukanaga K. *Statistical Pattern Recognition*. Academic Press: San Diego, CA, 1990.
37. Mello KL, Brown SD. Novel 'hybrid' classification method employing Bayesian networks. *J. Chemometrics* 1999; **13**: 579–590.
38. Todorovski L, Dzeroski S. Combining classifiers with meta decision trees. *Mach. Learn.* 2003; **50**: 223–249.
39. Selbig J, Mevissen T, Lengauer T. Decision tree-based formation of consensus protein secondary structure prediction. *Bioinformatics* 1999; **15**: 1039–1046.
40. Martin YC. *Quantitative Drug Design*. Marcel Dekker: New York, 1978.
41. Blower PE, Cross KP, Fligner MA, Myatt GJ, Verducci JS, Yang C. Systematic analysis of large screening sets in drug discovery. *Curr. Drug Discov. Technol.* 2004; **1**: 37–47.
42. Buntine W. Learning classification trees. *Statistics and Computing* 1992; **2**: 63–73.
43. Safavian SR, Landgrebe D. A survey of decision tree classifier methodology. *IEEE Trans. Systems, Man, and Cybernetics* 1991; **21**: 660–674.
44. Murthy SK. Automatic construction of decision trees from data: a multidisciplinary survey. *Data Mining and Knowledge Discovery* 1998; **2**: 345–389.