Tree-Based Methods

Xiaogang Su, Ph.D.
Department of Mathematical Sciences
University of Texas at El Paso (UTEP)
xsu@utep.edu

Contents

1 Introduction

A tree based method (or recursive partitioning) partitions data recursively till it arrives at a number of mutually exclusive groups. When compared with conventional statistical tools, it potentially has the following advantages, as stated in CART:

1. It can be applied to many data structures, especially those involving nonlinear patterns or many predictors that possibly interact in a complicated manner. Tree methods handle both ordered and categorical variables in a simple and natural way through appropriate formulation.

2. It does stepwise variable selection, interactions and complexity reduction in an automatic manner. And it makes powerful use of conditional information in handling nonhomogeneous relationships.

3. It is invariant under all monotone transformations of individual ordered variables.

4. It is extremely robust with respect to outliers and wrong recordings.

5. The tree procedure output gives easily understood and interpreted information regarding the predictive structure of the data.

Tree-based methods are very effective in handling multifaceted data and are gaining acceptance as a methodology for addressing data complexity, which renders them particularly popular in various application fields. The hierarchical tree structures are useful for creating model-based decision rules.

Tree modeling has gained so much popularity that its implementation has been made available in all major statistical computing packages. In R (R Development Core Team, 2010), three main packages are available: `tree`, `rpart`, and `party`; in SAS, tree analysis can be conducted using PROC SPLIT; and in SPSS, it is also implemented, in the Decision Trees module. Now, it is possible to perform tree analysis without a deep understanding of detailed steps in this algorithmic procedure.

1.1 History of Tree Methods

The history of tree methods can be traced back to Morgan and Sonquist (1963), who initiated the idea and implemented the first tree softwareVthe Automatic Interaction Detection. Tree models have been made popular and widely applicable by the introduction of the Classification and Regression Trees (CART; Breiman, Friedman, Olshen, and Stone, 1984) methodology. The tree size selection problem and many other practical issues in tree applications are addressed successfully with CART. The CART paradigm remains the current standard of tree modeling. Other noteworthy tree implementations include the chi-squared Automatic Interaction Detector by Kass (1980) and C4.5 by Quinlan (1993). There are typically two types of trees: regression trees when the outcome or response variable is continuous and classification trees (also known as decision trees) when the outcome is binary or categorical. From a statistical perspective, regression itself is a broader concept that includes the classification problem as a special case. In fact, regression trees have been extended to handle many other types of responses, such as count data, time series, censored failure time data, and longitudinal data. They also have been used for model diagnostics, variance modelling, subgroup analysis in clinical trials, and causal inference with observational data.

At the same time, the recursive partitioning methodologies has motivated a number of cutting-edge modern modelling or learning methods and been extended in a variety of ways. They have formed a very important class of learning tools that are widely applied in many fields. These extensions include the multivariate adaptive regression splines (MARS; Friedma, 1991 *Annals of Statistics*), Hierarchical Mixture of Experts (HME; Jordan and Jacobs, 1994), bagging (Breiman, 1996), boosting (Freund and Schapire, 1997), and random forests (Breiman, 2001).

1.2 Binary Tree Terminology

The whole procedure can be represented by a *binary tree.* Some related notation and terminology for binary trees are introduced as below. A convenient way of denoting the nodes is used 1 and 2. Figure 1 gives a picture of a binary tree structure (Source: *New York Times* on April 16, 2008).

We denote a binary tree as $\mathcal{T}$. We use h to denote a particular element or a *node* of $\mathcal{T}$. The name of each node h contains a combination of **1**'s and **2**'s. The *root* node, denoted as **1**, has no ancestor and contains the whole learning data. After the first splitting, the root node breaks into two parts: the left part **11** and the right part **12**. Subsequently, node **11** is divided into **111** and **112** and **12** into **121** and **122**. This procedure is iterated till a tree is grown. We call a node *terminal* if it does not have any descendants and a node *internal* if it does. The set containing all terminal nodes of $\mathcal{T}$ is denoted as $\tilde{\mathcal{T}}$ and the set containing all internal nodes is denoted as $\mathcal{S}$. Namely, $\mathcal{T} = \tilde{\mathcal{T}} + \mathcal{S}$. We follow the CART convention of picturing a terminal node as an ellipse and an internal node as a rectangle. Other obvious definitions include *mother*, *daughter*, and *sibling.*

Two additional terms, *subtree* and *branch* are particularly helpful in describing the pruning idea which is essential to CART.

Definition *A tree $\mathcal{T}^1$ is a subtree of $\mathcal{T}$ if $\mathcal{T}'$ is a tree with the same root node as $\mathcal{T}$, and if for every $h \in \mathcal{T}'$, $h \in \mathcal{T}$.*

Definition *A tree $\mathcal{T}_h$ is called a branch of $\mathcal{T}$ if $\mathcal{T}_h$ is a tree with root node $h \in \mathcal{T}$ and all descendants of h in $\mathcal{T}$ are descendants of h in $\mathcal{T}_h$.*

Pruning a branch $\mathcal{T}_h$ from a tree $\mathcal{T}$ consists of deleting from $\mathcal{T}$ all descendants of h, that is, cutting off all of $\mathcal{T}_h$ except its root node h. The tree pruned this way is denoted by $\mathcal{T} - \mathcal{T}_h$. The pruned tree

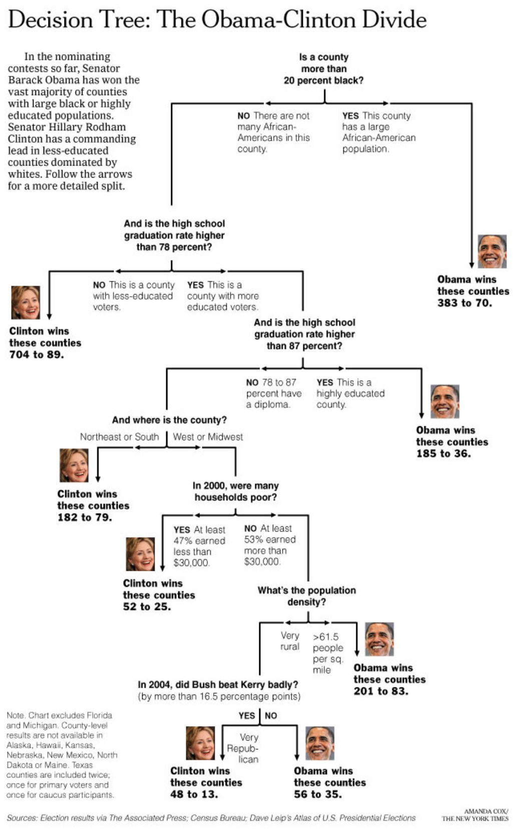
Decision Tree: The Obama-Clinton Divide
In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.
Is a county more than 20 percent black?
NO There are not many African-Americans in this county.
YES This county has a large African-American population.
Obama wins these counties 383 to 70.
And is the high school graduation rate higher than 78 percent?
NO This is a county with less-educated voters.
YES This is a county with more educated voters.
Clinton wins these counties 704 to 89.
And is the high school graduation rate higher than 87 percent?
NO 78 to 87 percent have a diploma.
YES This is a highly educated county.
Obama wins these counties 185 to 36.
And where is the county?
Northeast or South
West or Midwest
Clinton wins these counties 182 to 79.
In 2000, were many households poor?
YES At least 47% earned less than $30,000.
NO At least 53% earned more than $30,000.
Clinton wins these counties 52 to 25.
What's the population density?
Very rural
>61.5 people per sq. mile
Obama wins these counties 201 to 83.
In 2004, did Bush beat Kerry badly? (by more than 16.5 percentage points)
YES
NO
Very Repub-lican
Clinton wins these counties 48 to 13.
Obama wins these counties 56 to 35.
Note. Chart excludes Florida and Michigan. County-level results are not available in Alaska, Hawaii, Kansas, Nebraska, New Mexico, North Dakota or Maine. Texas counties are included twice; once for primary voters and once for caucus participants.
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections
AMANDA COX/ THE NEW YORK TIMES

Figure 1: Decision Trees from *New York Times*: The Obama-Clinton Divide.

$\mathcal{T} - \mathcal{T}_{11}$ is shown in Figure 2.1. If $\mathcal{T}^1$ is obtained from $\mathcal{T}$ by successively pruning off its branches, then $\mathcal{T}^1$ is called a pruned subtree of $\mathcal{T}$ and denoted by $\mathcal{T}^1 \prec \mathcal{T}$. Note that $\mathcal{T}^1$ and $\mathcal{T}$ have the same root node **1**.

We shall follow the CART convention to develop tree models, which contains the following three major steps:

1. Grow a large initial tree, $\mathcal{T}_0$;

2. Iteratively truncate branches of $\mathcal{T}_0$ to obtain a sequence of optimally pruned (nested) subtrees;

3. Select the best tree size based on validation provided by either the test sample method or cross validation (CV).

2 Splitting Data

A split could be induced by any question of the form "Is $\mathbf{x} \in S$ where $S \subset X$?". Other forms of splits are possible: splits of a single covariate, splits on linear combinations of predictors, and boolean combination splits. The simplest class of splits-splits on a single covariate- can be described by the following rules: (1) each split depends on the value of one predictor X_j; (2) if X_j is an ordered

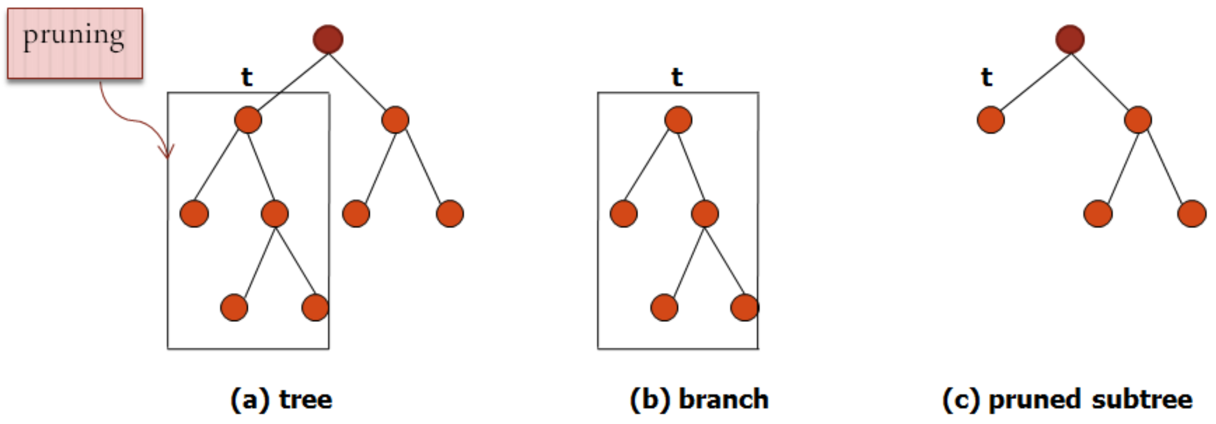

Figure 2: A Typical Tree Graph Illustrating Branch and Subtree.

variable, then splits induced by questions of the form "Is $X_j < c$?" are allowed; and (3) if X_j is nominal with values in $B = \{b_1, b_22, ..., b_r\}$ then splits induced by any question of the form "Is $X_j \in S$?" where $S \subset B$ are allowed. Partitioning a node, t, involves finding the split, s, among all variables that maximizes some performance measure. A tree is grown by finding the best split at each terminal node. The same splitting rule is applied recursively to the resulting nodes until a large tree is grown with a small number of observations falling into each node. The tree is large and overfits the data at this point. And then we need validation methods to identify the best subtree.

In the following discussion, we first focus on how to split data in decision trees with binary responses, i.e., $Y \in \{0, 1\}$. Then we discuss various issues associated with splitting data. Then the methods are extended to data with other types of responses.

2.1 Node Impurity

In general, the impurity $i(t)$ of node t can be defined as a nonnegative (concave) function of $P\{y = 1|t\}$, which is the occurrence rate in node t. More formally,

$$i(t) = \phi(P\{y = 1|t\}),$$

where the function $\phi(\cdot)$, very intuitively, the impurity is the largest when both classes are equally mixed together and it is the smallest when the node contains only one class. Hence, it has the following properties:

1. $\phi(p) \geq 0$;
2. $\phi(p)$ attains its minimum 0 when $p = 0$ or $p = 1$.
3. $\phi(p)$ attains its maximum when $p = 1 - p = 1/2$.
4. $\phi(p) = \phi(1 - p)$, *i.e.*, $\phi(p)$ is symmetric to $p = 1/2$.

Common choices of ϕ include: the Bayes error or the minimum error, the entropy function, and the Gini index.

- **The Bayes Error**
$$\phi(p) = \min(p, 1-p) = 1 - \max(p, 1-p).$$
This measure corresponds to the misclassification rate when majority vote is used. The Bayes error is rarely used in practice due to some undesirable properties as explained by Brieman et al. (1984, p. 99).

- **The Entropy Function**
$$\phi(p) = -p\log(p) - (1-p)\log(1-p).$$
Entropy is a measure of variability for categorical data. It was first developed by Shannon in 1984 to measure the uncertainty of a transmitted message. Quinlan (1993) first proposed to use the reduction of Entropy as a measure of split criteria. Ripley (1996) showed the entropy reduction criterion is equivalent to using the likelihood ratio chi-square statistics for association between the branches and the target categories.

- **The Gini Index**
$$\phi(p) = p(1-p).$$
Gini index is a measure of variability for categorical data developed by Italian statistician Corrado Gini in 1912. Breiman et al. (1984) proposed to use the reduction of Gini index as a measure for split criteria. It has been observed that this rule has an undesirable end-cut preference problem (Breiman et al., 1984, Ch. 11): It gives preference to the splits that result in two daughter nodes of extremely unbalanced sizes. To resolve this problem, a modification called the delta splitting method has been adopted in both the THAID (Morgan and Messenger, 1973) and CART programs.

2.1.1 Computation of $i(t)$

The computation of impurity is simple when the occurrence rate $P\{y = 1|t\}$ in node t is available. In many applications such as prospective studies, this occurrence rate can be estimated empirically from the data. At other times, additional prior information may be required to estimate the occurrence rate.

For a given split s, we have the following 2×2 table according to the split and the response.

	response		
node	0	1	
left (t_L)	n_{11}	n_{12}	$n_{1\cdot}$
right (t_R)	n_{21}	n_{22}	$n_{2\cdot}$
	$n_{\cdot 1}$	$n_{\cdot 2}$	n

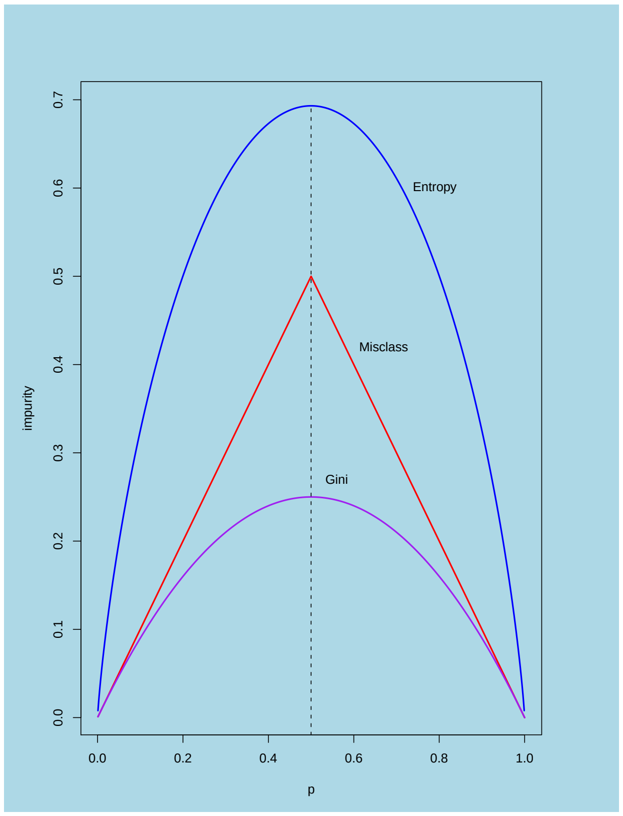

Figure 3: Plot of the three commonly-used impurity measures: entropy, misclassification error rate, and the Gini index.

In Prospective Studies In prospective studies, $P\{y = 1|t_L\}$ and $P\{y = 0|t_L\}$ can be estimated by $n_{12}/n_{1\cdot}$ and $n_{22}/n_{1\cdot}$. And hence

$$i(t_L) = -\frac{n_{12}}{n_{1\cdot}} \log\left(\frac{n_{12}}{n_{1\cdot}}\right) - \frac{n_{11}}{n_{1\cdot}} \log\left(\frac{n_{11}}{n_{1\cdot}}\right).$$

In fact, it can be shown that the above entropy criterion is proportional to the maximized log-likelihood score associated with t_L. In light of this fact, many node-splitting criteria originate from the maximum of certain likelihood functions. The importance of this observation will be appreciated later.

In Retrospective Studies In retrospective studies, Bayes' theorem can be used to compute $P\{y = 1|t_L\}$:

$$\begin{aligned} P\{y = 1 \,|\, t_L\} &= \frac{P\{y = 1,\ t_L\}}{P\{t_L\}} \\ &= \frac{P\{y = 1\}\, P\{t_L \,|\, y = 1\}}{P\{y = 1\}\, P\{t_L \,|\, y = 1\} + P\{y = 0\}\, P\{t_L \,|\, y = 0\}}, \end{aligned}$$

where $P\{y = 1\} = 1 - P\{y = 0\}$ is the prior occurrence rate of the event of interest.

2.1.2 Goodness-of-Split Measure

Suppose s be any candidate split and s divides t into t_L and t_R such that the proportions of the cases in t go into t_L and t_R are $P\{t_L\}$ and $P\{t_R\}$, respectively. Define the reduction in node impurity as

$$\Delta I(s,t) = i(t) - \left[P\{t_L\}i(t_L) + P\{t_R\}i(t_R)\right],$$

which provides a goodness-of-split measure for s. The best split $s^\star$ for node t provides the maximum impurity reduction, i.e.,

$$\Delta I(s^\star, t) = \max_{s \in \mathcal{S}} I(s,t).$$

Then t will be split into t_L and t_R according to the split $s^\star$ and the search procedure for the best split repeated on t_L and t_R separately. A node becomes to a terminal node when the impurity cannot decrease any further based some terminal conditions specified.

2.1.3 Alternative Splitting Criterion

There are two alternative splitting criteria: the twoing rule (CART, Brieman, et al., 1984) and the χ^2 test.

- The twoing rule is a different measure for the goodness of a split as follows:

$$\frac{P\{t_L\}P\{t_R\}}{4}\left[\sum_{j=0,1} \left|P\{y=j \,|\, t_L\} - P\{y=j \,|\, t_R\}\right|\right]^2.$$

  For a binary response, this twoing rule coincides with the use of the Gini index, which has the end-cut preference problem.

- The Pearson chi-square test statistics can be used to measure the difference between the observed cell frequencies and the expected cell frequencies (under the independent assumption). We can use the p-value to make the judgement. When p-vale is too small, we can use logworth:

$$\text{logworth} = -\log_{10}(\text{p-value}).$$

2.2 Induction of a Split

2.2.1 Count the Number of Splits

The next problem in tree construction is how to determine the number of partitions needed to examine at each node. An exhaustive (greedy) search algorithm considers all possible partitions of all input variables at every node in the tree. However, the number of child nodes tends to increase significantly when either too many levels in one variable or too many variables. This makes an exhaustive search algorithm prohibitively expensively.

Multi-way splits are not more flexible than binary splits. Actually, we can always find a binary tree that is equivalent to any multi way tree (see Figure 5.1). Multi-way splits often give a more interpretable tree because split variable tends to be used fewer times. However, the number of possible partitions for a binary split tree is much less than a multi-way split tree and exhaustive search is more feasible in binary split tree.

Examples

1. Suppose x is an ordinal variable with four levels 1, 2, 3, and 4. What is the total number of possible splits including both binary and multiway ones?

   Solution:

   2 way split: 1-234, 12-34, 123-4
   3 way split: 1-2-34, 1-23-4, 12-3-4
   4 way split: 1-2-3-4
   Total number of splits = 3+3+1=7

   Note that there are $2^{L-1} - 1$ possible splits for an ordinal variable with L levels. Splits in a tree only depend on the order statistics of the numerical variables. The same formula can be used to compute the number of possible partitions for both ordinal and numerical variables.

2. Suppose x is a numerical variable with 100 distinct values. What is the total number of possible splits?

   Solution:

   Total number of splits $= 2^{100-1} - 1 \approx 6.34 \times 10^{29}$.

3. Suppose x is a nominal variable with four categories a, b, c, d. What is the total number of possible splits?

   Solution:

   2 way split: ab-cd, ac-bd, ad-bc, abc-d, abd-c, acd-b, a-bcd
   3 way split: a-b-cd, a-bc-d, ac-b-d, ab-c-d, a-c-bd, ad-b-c
   4 way split: a-b-c-d
   Total number of splits = 7+6+1=14

Note that the total number of possible partitions for a nominal variable with L levels is called the *Bell number*

$$B_L = \sum_{i=2}^{L} S(L,i),$$

where $S(L,i)$ is the *Stirling number of the second kind* or also called a Stirling set number, which corresponds to the number of ways of partitioning a set of L elements into i nonempty sets (i.e., i set blocks). Special values of $S(L,i)$ include

$$\begin{aligned} S(L,L) &= S(L,1) = 1 \\ S(L,2) &= 2^{n-1} - 1 \\ S(L,L-1) &= \begin{pmatrix} n \\ 2 \end{pmatrix}. \end{aligned}$$

$S(L,i)$ satisfies

$$S(L,i) = i \cdot S(L-1,i) + S(L-1,i-1).$$

The Stirling numbers of the second kind can be computed from the sum

$$S(L,i) = \frac{1}{i!} \sum_{j=0}^{i-1} (-1)^j \begin{pmatrix} i \\ j \end{pmatrix} (i-j)^n.$$

2.2.2 Strategies to Reduce the Number of Possible Partitions

Since the exhaustive search is too expensive, many methods that can reduce the total number of partitions needed at each node have been developed. Some of these methods are listed below:

- Binary Splits Exclusive (CART like tree)
- Agglomerative Clustering of Levels (CHAID like tree)
- Variable Selection First
- Within-Node Sampling Approach
- Minimum Child Size

1. Binary Splits Exclusive

   One way to reduce the number of possible partitions is to consider only binary splits. If x_1 is an ordinal variable with L levels, then the total number of possible binary splits is $L-1$; if x_2 is an ordinal variable with L levels, then the total number of possible binary splits is $2^{L-1}-1$.

2. Agglomerative Clustering of Levels

   The procedure for clustering levels is as follows:

Step 1 : Start with an L-way split

Step 2 : Collapse the two levels that are closest based on some splitting criterion

Step 3 : Repeat Step 1 and Step 2 on the set of L-1 consolidated levels if $L-1 \geq 2$.

Step 4 : Step 1 to Step 3 will give the best split for each size and can choice the best split for this variable.

Step 5 : Repeat this process for all other input variables and choose the best possible split for all variables

3. Variable Selection First

   We can choice only these variables that are highly correlated to the target variable. Suppose there are 2000 binary variables and only 200 variables are highly correlated to the target variables, the number of possible partitions to be considered are reduced by 90%.

4. Within-Node Sampling

   The number of possible partitions to be considered depends on the sample size in a node. For example, there are 10000 observations in one node, the maximum possible distinct values for a numerical variable is 10000. Thus, the possible partitions for this numerical variable are 4999 if we use binary split exclusive if we only consider a sample of 5000 observations.

5. Set Minimum Number of Observations in Each Node

   Suppose the minimum number of observation for a child node is set to 50. We will not split this node further if the number of the observations in this node is below 50.

Besides, for categorical predictors that has many levels $\{B_1, \ldots, B_L\}$, one way to reduce the number of splits is to rank the levels as $\{B_{l_1}, \ldots, B_{l_L}\}$ according to the occurrence rate within each level

$$P\{1|B_{l_1}\} \leq P\{1|B_{l_2}\} \leq \cdots \leq P\{1|B_{l_L}\}$$

and then treat it as an ordinal input. (See CART, p. 101).

2.3 Other Issues

2.3.1 Bias Adjustment for Multi-Way Split

For the χ^2 test with multi-way splits, the large table has higher chance to produce a large chi-square statistics because the expectation of the chi-square statistics with γ degrees of freedom is γ and the large table has large degrees of freedom. However, the p-value does not depend on the size of the contingency table.

Both the Gini Index and the Entropy tend to increase as the number of branches increased. For example, the maximum possible value for a Gini index with 20 branches is 0.95 and the maximum possible value for a Gini index with two branches is 0.5. In addition, there is not any analogous adjustment similar to p-value in chi-square test statistics that can be used.

The use of splitting criterion can be thought as a two-step process. First, we can use split criterion to choice the best split among all possible splits for a given input variable. Then, we can use the split criterion to choice the best split among all the "best" split from each input variable. Both steps required some adjustment to ensure the fair comparison.

1. Adjustment Within the Same Input Variable

   - Adjustment is not necessary if only binary split is considered.
   - Adjustment can not be done if either Gini or Entropy split criterion is used
   - P-value (or logworth) adjustment can be used if the chi-square split criterion is used
   - Entropy tends to favor balance branches (Breiman, 1996)
   - Gini index tends to favor isolating the largest target class (Breiman, 1996)

2. Adjustment among Best Splits from Different Input Variables

   There are more splits to consider on a variable with more levels. Therefore, the maximum possible value for Gini index, Entropy, and logworth tends to become large as the number of possible splits, m, increases. For example, there is only one split for a binary input variable and there are 511 possible binary splits for a nominal input variable with 10 levels. Thus, all three split-criteria are favor variables with large number of levels. Nominal variables are favored than ordinal variables with the same number of levels. This problem has been identified as '*variable selection bias*' problem by Loh, WY.

   - Adjustment for Gini index is unavailable
   - The information gain ratio can be used to adjust Entropy (Quinlan, 1993). However, it is not available in Enterprise miner.

$$\text{information gain ratio} = \frac{\Delta\text{Entropy}}{\text{input levels in parent node}}.$$

- Bonferroni type of adjustment can be used to adjust the logworth (Kass, 1980). Kass adjustment is multiply the p-value by m, the number of possible splits. This adjustment is equivalent to subtract $\log_{10}(m)$ from the logworth.
- In order to identify the 'unbiased' split, Loh (2002) proposed a residual-based method of selecting the most important variable first, and then applying greedy search only on this variable to find the best cutpoint.

2.3.2 Variable Combination

Another major deficiency for the tree modeling introduced so far is we only consider splits on a single variable. Suppose that there is a linear structure existed in the data, tree structure modeling without considering the linear combination might not be able to pick the best model. Moreover, it might be worse than the existing method such as linear discriminate analysis. We use the next example (figure 5.2) to show illustrate the deficiency of ignoring linear combination of variables.

Discriminant analysis can do a perfect job to separate these two classes. However, the tree program without considering the linear combination of variable x_1 and x_2 would take many splits to separate these two classes because it tries to separate the plane into rectangular regions. However, this problem can be solved very easily if the tree can consider to separate the plane with linear combination of variables such as $x' = ax_1 + bx_2$. In the computer science literature, tree models that allows for linear combinations of inputs are terms as multivariate decision trees. One is referred to Li et al. (PHDRT, *JASA*, 2000) for recent research on this topic.

3 CART Procedure

Originally, the final tree was determined by use of stopping rules as done in the Automatic Interaction Detection procedure (AID). The initial stop-splitting rule was simple. Set a threshold $C > 0$, and declare a node h terminal if the maximum decrease in impurity was less than C, that is,

$$\max_s R(s, h) \ < \ C.$$

However, determining tree size by such kind of stopping rules generally produces unsatisfactory results. Early research work was centered on finding better criteria for declaring a node terminal. However, this tends out to be the wrong way of looking at the problem according to the inventors of CART. As they demonstrated, an arbitrary stopping rule has two main problems. If C is set too low, then there is too much splitting and the tree is too large. On the other hand, increasing C results in the following difficulty: There may be a node h such that $\max_s R(s, h)$ is small. But the descendant nodes h_L, h_R of h may have splits with large decreases in impurity. Declaring h terminal results in loss of important structure.

In order to circumvent the overfitting or underfitting problems due to stopping rules, Brieman et al. (1984) presented a procedure for regression and classification trees (CART). The essential idea of CART is selecting the best-sized tree by evaluating the subtrees of a large tree. They used a pruning algorithm to identify a sequence of nested subtrees, and then validated the performance of each subtree with test sample or resampling depending on the sample size available.

3.1 Growing a Large Tree

First a large initial tree $\mathcal{T}_{\max}$ is grown so that no significant structure would be missed out. This large tree can also be used to explore the data structure.

At this stage, CART claims a node terminal if one of the following situations occurs:

- The node contains less than $N_{\min}$, say, 10 observations. This threshold sample size $N_{\min}$ could be made case by case. But usually it is set small enough to construct a sufficiently large initial tree.
- The node is pure. By saying a node *pure*, it means that all y_i's in that node are identically valued.
- The node contains only identical covariate values.

3.2 Pruning

Now an optimally sized subtree needs to be chosen from the subtrees of the large tree $\mathcal{T}_{\max}$ initially grown. One possibility is to consider all its possible subtrees. But this would be too computationally overwhelming since the total number of subtrees increases much more rapidly as the size of the initial tree grows. To observe how fast the number of subtrees grows, consider trees, $\mathcal{T}_g$, for which all terminal nodes have g ancestors. Let $N(\mathcal{T})$ denote the number of subtrees that tree $\mathcal{T}$ has. We have $N(\mathcal{T}_0) = 1$, $N(\mathcal{T}_1) = 2$, $N(\mathcal{T}_2) = 5$, $P(\mathcal{T}_3) = 26$, $N(\mathcal{T}_4) = 677$, and $N(\mathcal{T}_5) = 458,330$. Note that $\mathcal{T}_5$ has only 32 terminal nodes. Because trees substantially larger than this may be grown, an efficient algorithm is needed if optimal trees are desired

The idea of CART is to obtain a small sequence of subtrees via a computationally efficient pruning method. This method is termed as *minimal cost-complexity pruning* algorithm.

The cost complexity of a subtree is a linear combination of the cost of a tree and a complexity penalty for each terminal node. The cost at node h, $R(h)$, is defined as

$$R(h) \;=\; \frac{1}{N} \sum_{i:\mathbf{x_i} \in h} (y_i - \bar{y}(h))^2$$

where N is the total sample size in the whole learning data, not in this particular node h.

Definition *For any subtree $\mathcal{T}$ of $\mathcal{T}_{\max}$, define its complexity as $|\tilde{\mathcal{T}}|$, the number of terminal nodes in $\mathcal{T}$. Let $\alpha \geq 0$ be a real number called the complexity parameter. Define the cost-complexity measure $R_\alpha(\mathcal{T})$ as*

$$R_\alpha(\mathcal{T}) \;=\; R(\mathcal{T}) + \alpha|\tilde{\mathcal{T}}|$$

where $R(\mathcal{T})$ is the sum of the costs at each terminal node:

$$\begin{aligned} R(\mathcal{T}) &= \sum_{h \in \tilde{\mathcal{T}}} R(h) \\ &= \frac{1}{N} \sum_{h \in \tilde{\mathcal{T}}} \sum_{i:\mathbf{x_i} \in h} (y_i - \bar{y}(h))^2. \end{aligned}$$

The definition of $R_\alpha(\mathcal{T})$ is such that for small α the tree $\hat{\mathcal{T}}$ that minimizes $R_\alpha(\mathcal{T})$ will be large. If α is large, the tree $\hat{\mathcal{T}}$ that minimizes $R_\alpha(\mathcal{T})$ will have fewer nodes. For large enough α, $\hat{\mathcal{T}}$ consists of only the root node. In all, this penalty α controls the trade-off between the parsimony of the model and the fidelity of the model to the data.

Now, for each value of α, find the subtree $\mathcal{T} \prec \mathcal{T}_{\max}$ which minimizes $R_\alpha(\mathcal{T})$.

Definition *The smallest minimizing subtree $\mathcal{T}(\alpha)$ for complexity parameter α is defined by the conditions*

1. $\mathcal{T}(\alpha) \;=\; \mathit{argmin}_{\{\mathbf{T}:\mathcal{T}\prec\mathcal{T}_{\max}\}} R_\alpha(\mathcal{T})$,
2. *If* $R_\alpha(\mathcal{T}) \;=\; R_\alpha\left(\mathcal{T}(\alpha)\right)$, *then* $\mathcal{T}(\alpha) \prec \mathcal{T}$.

Although α runs through a continuum of values, there are at most a finite number of subtrees of $\mathcal{T}_{\max}$. Because of this finiteness, what happens is that if $\mathcal{T}(\alpha)$ is the minimizing tree for a given value of α, then it continues to be minimizing as α increases until a jump point α' is reached, and a new tree $\mathcal{T}(\alpha')$ becomes minimizing and continues to be the minimizer until the next jump point α''.

The existence of $\mathcal{T}(\alpha)$ is guaranteed by the following proposition in CART.

Proposition 1. *For every value of α, there exists a unique smallest minimizing subtree $\mathcal{T}(\alpha)$.*

Consider a branch $\mathcal{T}_h$ with root node h. It is always true that $R(\mathcal{T}_h) < R(\{h\}) = R(h)$. The cost-complexity of $\mathcal{T}_h$ is

$$R_\alpha(\mathcal{T}_h) = R(\mathcal{T}_h) + \alpha|\tilde{\mathcal{T}}_h|,$$

and the cost-complexity of $\{h\}$ is

$$R_\alpha(\{h\}) = R(\{h\}) + \alpha.$$

As α increases from zero, there will a value where the cost-complexity of the branch $\mathcal{T}_h$ will become greater than the cost-complexity of the single node $\{h\}$. Therefore, at this point, $\{h\}$ is preferable to the branch $\mathcal{T}_h$. To find this critical value of α, solving the equation

$$R_\alpha(\mathcal{T}_h) = R_\alpha(\{h\})$$

leads to

$$\alpha = \frac{R(h) - R(\mathcal{T}_h)}{|\tilde{\mathcal{T}}_h| - 1}. \tag{1}$$

The heart of minimal cost-complexity pruning lies in the idea of *weakest-link cutting.* Starting with $\mathcal{T}_{\max}$, define a function $g(h)$, $h \in \mathcal{T}_{\max}$, by

$$g(h) = \begin{cases} \frac{R(h)-R(\mathcal{T}_h)}{|\tilde{\mathcal{T}}_h|-1} & t \notin \tilde{\mathcal{T}}_{\max}; \\ +\infty & t \in \tilde{\mathcal{T}}_{\max}. \end{cases}$$

Then define the weakest link $\bar{h}_1$ in $\mathcal{T}_{\max}$ as the node satisfying

$$g(\bar{h}_1) = \min_{h \in \mathcal{T}_{\max}} g(h),$$

and set

$$\alpha_2 = g(\bar{h}_1).$$

The node $\bar{h}_1$ is the weakest link in the sense that as the parameter α increases, it is the first node such that $R_\alpha(\{h\})$ becomes equal to $R_\alpha(\mathcal{T}_h)$. Then $\{\bar{h}_1\}$ becomes preferable to $\mathcal{T}_{\bar{h}_1}$, and α_2 is the value of α at which equality occurs.

Repeat the procedure by defining a new tree $\mathcal{T}^2 \prec \mathcal{T}_{\max} = \mathcal{T}^1$ after pruning away the branch $\mathcal{T}_{\bar{h}_1}$, i.e.,

$$\mathcal{T}^2 = \mathcal{T}^1 - \mathcal{T}_{\bar{h}_1},$$

and then finding the weakest link $\bar{h}_2$ in $\mathcal{T}^2$ and the corresponding parameter value α_3. This procedure is repeated until only the root node remains. If at any stage, there is a multiplicity of weakest links, for instance, if $g(\bar{h}_k) = g(\bar{h}_{k'})$, then get $\mathcal{T}^{k+1}$ by cutting out all the weakest links, namely,

$$\mathcal{T}^{k+1} = \mathcal{T}^k - \mathcal{T}_{\bar{h}_k} - \mathcal{T}_{\bar{h}_{k'}}.$$

The pruning procedure results in a decreasing nested sequence of m pruned subtrees

$$\mathcal{T}^1 = T_{\max} \succ T^2 \succ \cdots \succ T^m = \{\mathbf{1}\}$$

and an increasing sequence of parameters

$$0 = \alpha_1 < \alpha_2 < \cdots < \alpha_{m-1} < \alpha_m,$$

where $\{\mathbf{1}\}$ is the tree containing the root node only.

The following proposition in CART is particularly useful in selecting the best sized tree via resampling techniques.

Proposition 2. *For any α such that $\alpha_k \leq \alpha < \alpha_{k+1}$, $\mathcal{T}(\alpha) = \mathcal{T}(\alpha_k) = \mathcal{T}^k$.*

This implies that we can get the best pruned subtree for any penalty α from the pruning algorithm by simple interpolation.

3.3 Selection of the Best Sized Tree

Now we need to select one or several appropriately sized trees from the nested sequence. A reasonable way is to base the selection on the mean squared prediction error (MSE) for each subtree.

However, since the MSE depends on the scale in which the response was measured, CART uses the *relative mean squared error* to guide the tree selection.

Defintion *Let $d(\mathbf{x})$ be a real-valued function of $\mathbf{x} \in \mathcal{X}$, which is used to predict the response y. The relative mean squared error $RE(d)$ of the prediction rule $d(\mathbf{x})$ is defined as*

$$RE(d) = \frac{E(y - d(\mathbf{x}))^2}{E(y - E(y))^2}.$$

The denominator is usually estimated as

$$R(\bar{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2.$$

In tree-based regression, $d(\mathbf{x})$ is the response average in the node where $\mathbf{x}$ falls and hence $R(\mathcal{T}^k)$ actually gives an estimate of $E(y - d(\mathbf{x}))^2$. However, this substitution estimate is too optimistic to be reasonable as the tree is grown based on minimizing this quantity. Therefore, an honest estimate of the MSE, $E(y - d(\mathbf{x}))^2$, is critical in choosing the right tree size. Two methods have been suggested for this purpose in CART: *test sample* and *cross validation*.

3.3.1 Test Sample

When the sample size is large enough, test sample method is a convenient way to get an estimate of $R(\mathcal{T}^k)$. To achieve this, the whole sample $\mathcal{L}$ is randomly divided into a *learning sample* $\mathcal{L}_1$ and a *test sample* $\mathcal{L}_2$. Usually, the proportion is 2 : 1. Let N_1 and N_2 denote the sample size for $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively.

The learning sample $\mathcal{L}_1$ is used to grow a large tree $\mathcal{T}_{\max}$ and then get the sequence $\{\mathcal{T}_k\}$ of optimally pruned subtrees. Send the test sample down to $\mathcal{T}_{\max}$ and recalculate $R(h)$ as

$$R(h) = \frac{1}{N_2} \sum_{\{i:(\mathbf{x}_i, y_i) \in \mathcal{L}_2 \cap h\}} (y_i - d(\mathbf{x}_i))^2$$

for each terminal node h of $\mathcal{T}_{\max}$ based on the test sample. Accordingly, the test sample estimate, $R^{ts}(\mathcal{T}^k)$ for subtree $\mathcal{T}^k$ can also be recalculated by summing up new estimates of $R(h)$'s. Note that the predicted value, $d(\mathbf{x})$, for each $\mathbf{x}$, is computed solely based on the learning sample. Namely,

$$d(\mathbf{x}) \;=\; \frac{1}{n(h(\mathbf{x}))} \sum_{\{i:(\mathbf{x}_i, y_i) \in \mathcal{L}_1 \cap h(\mathbf{x})\}} y_i.$$

So the resulting $R^{ts}(\mathcal{T}^k)$ does not necessarily decrease with the tree complexity.

3.3.2 Cross Validation

In practice, taking observations is costly and the sample available is often of moderate size. In such cases, cross-validation is the preferred method.

In V-fold cross-validation, the data $\mathcal{L}$ is randomly divided into V subsamples $\mathcal{L}_v$ of approximately the same size. Let the vth learning sample be $\mathcal{L}^{(v)} \equiv \mathcal{L} - \mathcal{L}_v$, namely, the whole sample excluding the vth subset $\mathcal{L}_v$.

First, grow and prune using all the observations in the whole sample $\mathcal{L}$. This results in the sequences $\{(\mathcal{T}_k, \alpha_k), k = 1, \ldots, m\}$. Define $\alpha'_k = \sqrt{\alpha_k \alpha_{k+1}}$.

For each v, repeat the tree growing and pruning procedures using $\mathcal{L}^{(v)}$. And hence, the minimal error-complexity subtree for any parameter value α, $\mathcal{T}^{(v)}(\alpha)$ can be produced easily by recalling proposition 2. In particular, those corresponding to the parameters $\{\alpha'_k\}$, $\{\mathcal{T}^{(v)}(\alpha'_k)\}$, are available.

For each k, $k = 1, \ldots, m$, $\mathcal{L}_v$ is sent down the tree $\mathcal{T}^{(v)}(\alpha'_k)$ and the fitted value for each $y_i \in \mathcal{L}_v$ is simply the (learning sample based) node average, namely,

$$d_k^{(v)}(\mathbf{x}_i) = \frac{1}{|h(y_i)|} \sum_{\{(\mathbf{x}_j, y_j) \in h(y_i) \cap \mathcal{L}^{(v)}\}} y_j,$$

where $h(y_i)$ denotes the node into which y_i falls and $|h(y_i)|$ is its node size.

Therefore, the cross-validation estimates $R^{cv}(\mathcal{T}_k)$ is given as

$$R^{cv}(\mathcal{T}_k) = \frac{1}{N} \sum_{v=1}^{V} \sum_{\{i:(\mathbf{x}_i, y_i) \in \mathcal{L}_v\}} \left(y_i - d_k^{(v)}(\mathbf{x}_i) \right)^2 .$$

3.3.3 Tree Selection via 1 SE Rule

The best sized tree can be chosen as the subtree minimizing the test-sample or cross-validation estimate of prediction error, MSE or RE.

Breiman et al. (1984) noticed that the estimate of MSE or RE tends to decrease rapidly as the tree size increases from the root node. Then there is a wide flat valley with the estimated prediction error rising slowly as the number of terminal nodes gets large. To remedy this, they introduced the "1 SE" rule to keep the tree as simple as possible without sacrificing much accuracy, and to reduce the instability in tree selection. CART chooses the smallest tree $\mathcal{T}_k$ such that

$$R^{cv}(\mathcal{T}_k) \leq R^{cv}(\mathcal{T}_{k_0}) + SE,$$

where

$$R^{cv}(\mathcal{T}_{k_0}) = \min_k R^{cv}(\mathcal{T}_k),$$

and SE is the standard error estimate for $R^{cv}(\mathcal{T}_{k_0})$. This estimate is obtained by assuming that the terms in $R^{cv}(\mathcal{T}_{k_0})$ are independent. More details can be found in CART.

4 Other Types of Responses

As we have seen with binary responses, the splitting statistics generally can be sought in two different ways: either by minimizing within-node impurity or by maximizing the between-node differences. The former one is the conventional approach. With other types of responses, one basically needs to find an appropriate measure of impurity. The latter one is more statistically oriented. In this approach, any two-sample statistic could be a good choice used for the splitting statistic.

In the following, we shall discuss the extensions with categorical (nominal and ordinal) responses and continuous responses; extensions to other types of responses can be made in the same spirits.

4.1 Categorical Responses

Suppose that the response y has J classes: $\{1, 2, \ldots, J\}$. If the goodness-of-split criterion is used for comparing splits, the best split $s^\star$ achieves the greatest reduction in terms of node impurity. Namely,

$$\Delta i(s^\star, t) = \max_{s \in \mathcal{S}} \Delta i(s, t),$$

where

$$\Delta i(s, t) = i(t) - \{p(t_L) i(t_L) + p(t_R) i(t_R)\} .$$

The node impurity measure $i(t)$ is a nonnegative concave function of $p(j|t)$ for $j = 1, 2, \ldots, J$, which are the probability that an individual falls into class j in node t. Computation of $p(j|t)$ depends on the availability of the prior probabilities.

4.1.1 Entropy

Entropy-Based Node Impurity is given by

$$i(t) = -\sum_{j=1}^{J} p(j|t)\log\{p(j|t)\}$$

The entropy based node impurity is preferable in the splitting stage as it favors balanced splits. Same as in the binary response case, the above entropy measure has a correspondence with the maximum likelihood score; and the goodness-of-split criterion based on entropy is equivalent to maximizing the likelihood ratio test statistic.

4.1.2 Gini Index

CART gives lots of credit to Gini index mainly because of its interpretation and easy incorporation of misclassification cost. Gini Index -Based Node Impurity is given by

$$\begin{aligned} i(t) &= \sum_{j=1}^{J} p(j|t)(1-p(j|t) \\ &= 1-\sum_{j=1}^{J} p^2(j|t) \\ &= \left\{\sum_{j=1}^{J} p(j|t)\right\}^2 - \sum_{j=1}^{J} p^2(j|t) \\ &= \sum_{j\neq i} p(j|t)p(i|t). \end{aligned}$$

using $\sum_{j=1}^{J} p(j|t) = 1$. The Gini index has the following interpretation. Instead of using the plurality (majority) rule to classify objects in a node t, one may use the rule that assigns an object at random from the node to class i with probability $p(i|t)$. The estimated probability that the item is *actually* in class j is $p(j|t)$. Therefore, the estimated probability of misclassification under this rule is the Gini index $\sum_{j\neq i} p(j|t)p(i|t)$. The above property allows easy incorporation of misclassification cost into the Gini index and makes the Gini index a useful criterion frequently used in the final tree size selection.

Another interpretation for expression $i(t) = \sum_{j=1}^{J} p(j|t)(1-p(j|t)$ can be made in terms of the variances of dummy variables that indicate memberships. Gini index as a function $\phi(p_1,\ldots,p_J)$ of $p_1,\ldots,p_J$ is concave in the sense that for $r+s=1$, $r,s\geq 0$,

$$\phi(rp_1+sp'_1,\ldots,rp_J+sp'_J) \geq r\phi(p_1,\ldots,p_J)+s\phi(p'_1,\ldots,p'_J).$$

This ensures that for any split s,

$$\Delta(s,t)\geq 0.$$

However, the Gini index based splitting criterion tends to favor unbalanced splits. Again, the delta splitting rule can be applied: the values of cut points for x_k are restricted to the interval $(\min_i x_{ik}+\delta,\ \max_i x_{ik}-\delta)$.

4.1.3 Twoing Rule

The twoing rule also can be used. Denote the whole set of all classes by $\mathcal{C}$, i.e., $\mathcal{C} = \{1, \ldots, J\}$. At each node, separate the classes into two superclasses or subsets,

$$\mathcal{C}_1 = \{j_1, \ldots, j_{n_1}\} \quad \text{and} \quad \mathcal{C}_2 = \mathcal{C} - \mathcal{C}_1.$$

Assign all objects in $\mathcal{C}_1$ as level 1 and all objects in $\mathcal{C}_2$ as level 0. For any given split s of the node t, compute the Gini index based $\Delta i(s,t)$ *as if it were a two-class problem.* And find the superclass $\mathcal{C}_1$ that maximizes $\Delta i(s,t,\mathcal{C}_s)$. The best split $s^\star$ maximizes $\Delta i(s,t,\mathcal{C}_s)$:

$$\Delta i(s^\star, t, \mathcal{C}_{s^\star}) = \max_{s \in \mathcal{S}} \Delta i(s,t,\mathcal{C}_s).$$

The key idea of the twoing rule is to transfer the multi-class problem to the two-class (binary response) problem. Note that one significant advantage of this approach is that it gives "strategic" splits and informs the user of class similarities. Near the top of the tree structure, this criterion attempts to group together large numbers of classes that are similar in some characteristic; near the bottom of the tree it attempts to isolate single classes. The twoing rule has particular use for ordinal responses. Apparently, the twoing rule has disadvantage in computational efficiency. However, COROLLARY 4.11 of CART (Brieman, et al., 1984) shows, rather surprisingly, that twoing can be made into an overall criterion, as restated in Theorem 4.1.

Theorem 4.1. *For any node t and split s of t into t_L and t_R, define the twoing criterion function $\psi(s,t)$ by*

$$\psi(s,t) = \frac{P(t_L)P(t_R)}{4} \left\{ \sum_{j=1}^{J} |p(j|t_L) - p(j|t_R)| \right\}^2 .$$

Then the best splits $s^\star$ $(\mathcal{C}_{1s^\star})$ is given by the split that maximizes $\psi(s,t)$ and $\mathcal{C}_{1s^\star}$ is given by

$$\{j : \ p(j|t_L^\star) \geq p(j|t_R^\star)\},$$

where $t_L^\star, t_R^\star$ are the nodes given by the best split $s^\star$.

The twoing rule also can be naturally modified to incorporate ordinal responses. It is natural to consider the *ordered twoing criterion* given by

$$\psi(s,t) = \max_{\mathcal{C}_1} \Delta i(s,t,\mathcal{C}_1),$$

where $\mathcal{C}_1$ and $\mathcal{C}_2$ are partitions of $\mathcal{C} = \{1, \ldots, J\}$ into two superclasses restricted by the condition that they be of the form

$$\mathcal{C}_1 = \{1, \ldots, j_1\} \quad \text{and} \quad \mathcal{C}_2 = \{j_1 + 1, \ldots, J\},$$

and $j_1 = 1, \ldots, J - 1$.

node	**response** 1	2	$\cdots$	J	
left	n_{11}	n_{12}	$\cdots$	n_{1J}	$n_{1\cdot}$
right	n_{21}	n_{22}	$\cdots$	n_{2J}	$n_{2\cdot}$
	$n_{\cdot 1}$	$n_{\cdot 2}$	$\cdots$	$n_{\cdot J}$	n

4.1.4 Splitting with χ^2 Test

Alternatively, one may split data by maximizing the between-node difference. The χ^2 test would be a natural choice for measuring the between-node difference in categorical responses. A split s induces the following $2 \times J$ contingency table:

The chi-square test statistic can be constructed as

$$\chi^2 = \sum_{i=1}^{2}\sum_{j=1}^{J} \frac{n_{ij} - E_{ij}}{E_{ij}} \quad \sim \quad \chi^2_{(J-1)},$$

where $E_{ij} = n_{i\cdot}n_{\cdot j}/n$ is the expected count under the null of no association between split and response.

The chi square test can be easily extended to split data with multi-class responses. For binary splits, the best split maximizes the chi square test statistic. For multi-way splits, one may compute the logworth associate with the chi square test

$$\text{logworth} = -\log(\text{p-value}) = -\log \Pr\left\{\chi^2_{(J-1)} > \chi^2\right\}$$

and select the split associated with largest logworth. Note that chi square test is the only way of performing multi-way splits in SAS Enterprise Miner (SEM). Another advantage of splitting data with chi square tests is that it allows Bonferroni-type adjustments for number of inputs and depth.

4.1.5 Misclassification Cost

For pruning and tree size selection purposes, there are two ways of defining the misclassification cost for a particular node t, depending on the classifying criterion.

1. If the plurality or majority rule is used as classification rule, one may define the total misclassification cost for node t as
$$R(t) = \sum_{i \in t} c(j_t | y_i),$$
where j_t is the assigned class for node t by the plurality rule and $c(i|j)$ denotes the cost for misclassifying a class-j object as class i. Note the $c(i|i) = 0$.

2. Alternative, one may use the Gini index to incorporate misclassification cost
$$R(t) = \sum_{i \neq j} c(j|i)p(i|t)p(j|t),$$
for $i, j = 1, \ldots, J$. Recall that $p(i|t)p(j|t)$ is the probability that misclassifies i as j and vice versa.

The total misclassification cost for tree T is then

$$R(T) = \sum_{t \in \widetilde{T}} R(t).$$

The cost-complexity pruning can then be readily extended to the multi-class responses. Furthermore, the misclassification cost, after validation, can be used to evaluate the subtrees and select the best tree size.

For ordinal responses, usually the misclassification cost does the special handling, e.g., assigning class 6 as 1 costs more than assigning class 2 as 1 since class 2 is closer to class 1 than class 6. To incorporate the ordering, one sets $c(1|6) > c(1|2)$.

4.2 Continuous Responses

With continuous responses, a natural choice of node impurity for node t is the within-node variation of the response

$$i(t) = \sum_{i \in t} (y_i - \bar{y}_t)^2,$$

where $\bar{y}_t$ is the average of y_i's within node t. To evaluate a split that bisects a node t into its two child nodes t_L and t_R, we maximize the goodness-of-split criterion

$$\Delta i(s, t) = i(t) - \{i(t_L) + i(t_R)\}.$$

Note that, unlike decision trees, the goodness-of-split criterion does not need weights.

Alternatively, one may also use F test to evaluate binary splits

$$F = \frac{SSB/1}{SSW/(n-2)} \quad \sim \quad F(1,\ n-2),$$

where $SSW = i(t_L) + i(t_R)$ is the total within-node variation and $SSB = n_L \cdot (\bar{y}_L - \bar{y})^2 + n_R \cdot (\bar{y}_R - \bar{y})^2$ is the total between-node variation. Equivalently, the two-sample t test can be used.

It can be easily shown that maximizing the goodness-of-split criterion is equivalent to maximizing the F test statistic, both also equivalent to maximizing the likelihood ratio test statistic. However, the F test statistic has more advantages because it has a distribution to reference.

The logworth associated with F is $-\log \Pr\{F(1, n-2) > F\}$. The best split maximizes the logworth. This is particularly useful to evaluate multi-way splits, in which case

$$F = \frac{SSB/(m-1)}{SSW/(n-m)} \quad \sim \quad F(m-1,\ n-m)$$

for an m-way split. The logworth also allows for Bonferroni type adjustment.

Recall that for the F test to be valid, there are three statistical assumptions: independnece, normality, and homoscedasticity. Among them, the violation of homoscedasticity could seriously bias the split selection. In order to detect heteroscedasticity, one may plot the residuals versus predicted response and look for systematic patterns. Also, there are various statistical tests available for checking heteroscedasticity such as score tests. There are two treatments to this problem. The first is to transform the response. For example the logarithm transformation often helps stabilize the variance. The second is to use weighted least squares:

$$i(t) = \sum_{i \in t} w_i \cdot (y_i - \bar{y}_t)^2.$$

To prune, we can mae use of $i(t)$ to define the tree cost as

$$R(T) = \sum_{t \in \widetilde{T}} i(t) = \sum_{t \in \widetilde{T}} \sum_{i \in t} (y_i - \bar{y}_t)^2$$

and then form the cost-complexity measure. In size selection, the prediction mean squared error (PMSE) is used

$$PMSE = \sum \sum_{t \in \widetilde{T}} \sum_{i \in t \cap \mathcal{L}_2} (y_i - \bar{y}_t)^2,$$

where y_i's are responses in the test sample $\mathcal{L}_2$ and $\bar{y}_t$'s are the node averages computed using the learning sample $\mathcal{L}_1$.

5 Discussion: The Cons

Earlier, we have discussed the advantages of tree methods. Then what are their disadvantages and common pitfalls in tree analysis? A few are listed below. First of all, one common mistake in interpreting a tree structure is the over-emphasis of the identified interaction. The hierarchical structure is the very natural way that tree models approximates the underlying regression function. Although the earlier implementation of trees is called automatic interaction detection, given a tree structure, it remains dazzling to tell which covariates interact with which. For this purpose, one may refer to Su et al. (2009) and Su et al. (2012) for explicit detection of treatment-by-covariates interactions. Secondly, a single tree may not be doing so well for estimation or prediction tasks. Also tree models are notorious for their instability in the sense that a small altercation in the data may result in dramatic changes in the final tree structure. These two weaknesses can be utilized and improved by MARS, bagging, boosting, and random forests (i.e., perturb & ensemble procedures). Among others, there are two additional quite inherent weakness of recursive paritioning. First, the best split is locally optimal at each node ? the tree structure may not be globally optimal. The second is about the biased variable selection, e.g., a categorical variable with many levels is more likely to be selected than a binary variable even though neither is related to the response. Interested readers may refer to Loh and colleagues for research endeavors in correcting the variable selection bias.

REFERENCES

[1] Breiman, L. (1996). Bagging predictors. *Machine Learning*, **24**(2): 123?-140.

[2] Breiman, L. (2001), Random Forests. *Machine Learning*, **45**(1): 5–32.

[3] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984) Classification and Regression Trees, Chapman and Hall.

[4] Friedman, J. (1991). Multivariate Adaptive Regression Splines. *Annals of Statistics*, **19**: 1–141.

[5] Freund, Y. and Schapire, R. E. (1997); A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, **55**(1):119–139.

[6] Hothorn, T., Hornik, K., and Zeileis, A. (2006). Unbiased Recursive Partitioning: A Conditional Inference Framework. *Journal of Computational and Graphical Statistics*, **15**(3), 651–674.

[7] Kass, G. V. (1980). An Exploratory Technique for Investigating Large Quantities of Categorical Data. *Applied Statistics*, **29**: 119–127.

[8] Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical Mixture of Experts and the EM Algorithm. *Neural Computation*, **6**(2): 181–214.

[9] LeBlanc, M. and Crowley, J. (1993). Survival Trees by Goodness of Split. *Journal of the American Statistical Association*, **88**, 457–467.

[10] Loh, W. and Vanichsetakul, N. (1988), "Tree-Structured Classification via Generalized Discriminate Analysis," Vol. 83, pp. 715-727.

[11] Loh, W. and Shih, Y. (1997) Split Selection Methods for Classification Trees, Statistical sinica, Vol. 7, pp 815-840.

[12] Quinlan, J. R. (1993), Programs for Machine Learning, San Mateo, CA: Morgan Kaufmann.

[13] SAS Institute (2000) Decision Tree Modeling Course Notes.

[14] Su, X. G., Wang, M., and Fan, J. J. (2004). Maximum Likelihood Regression Trees. *Journal of Computational and Graphic Statistics*, **13**: 586–598.

[15] Su, X. G., Tsai, C.-L., Wang, H., Nickerson, D., and Li, B. (2009). Subgroup Analysis via Recursive Partitioning. *Journal of Machine Learning Research*, **10**: 141–158.

[16] Su, X. G., Kang, J., Fan, J., Levine, R., and Yan, X. (2012). Facilitating Score and Causal InferenceTrees for Large Observational Data. *Journal of Machine Learning Research*, **13**: 2955?-2994.