Classification and Regression Tree (CART) Techniques

1. Introduction

In certain research studies development of a reliable decision rule, which can be used to classify new observations into some predefined categories, plays an important role. The existing traditional statistical methods are inappropriate to use in certain specific situations, or of limited utility, in addressing these types of classification problems. There are a number of reasons for these difficulties. First, there are generally many possible "predictor" variables which makes the task of variable selection difficult. Traditional statistical methods are poorly suited for this sort of multiple comparisons. Second, the predictor variables are rarely nicely distributed. Many variables (in agriculture and other real life situations) are not normally distributed and different groups of subjects may have markedly different degrees of variation or variance. Third, complex interactions or patterns may exist in the data. For example, the value of one variable (e.g., age) may substantially affect the importance of another variable (e.g., weight). These types of interactions are generally difficult to model and virtually impossible to model when the number of interactions and variables becomes substantial. Fourth, the results of traditional methods may be difficult to use. For example, a multivariate logistic regression model yields a probability for different classes of the dependent variable, which can be calculated using the regression coefficients and the values of the explanatory variable. But practitioners generally do not think in terms of probability but, rather in terms of categories, such as "presence" versus "absence." Regardless of the statistical methodology being used, the creation of a decision rule requires a relatively large dataset.

In recent times, there has been increasing interest in the use of Classification And Regression Tree (CART) analysis. CART analysis is a tree-building technique which is different from traditional data analysis methods. In a number of studies, CART has been found to be quite effective for creating decision rules which perform as well or better than rules developed using more traditional methods. In addition, CART is often able to uncover complex interactions between predictors which may be difficult or impossible using traditional multivariate techniques. It is now possible to perform a CART analysis with a simple understanding of each of the multiple steps involved in its procedure. Classification tree methods such as CART are convenient way to produce a prediction rule from a set of observations described in terms of a vector of features and a response value. The aim is to define a general prediction rule which can be used to assign a response value to the cases solely on the bases of their predictor (explanatory) variables. Tree-structured classifications are not based on assumptions of normality and user-specified model statements, as are some conventional methods such as discriminant analysis and ordinary least square regression. Tree based classification and regression procedure have greatly increased in popularity during the recent years. Tree based decision methods are statistical systems that mine data to predict or classify future observations based on a set of decision rules and are sometimes called rule induction methods because the reasoning process behind them is clearly evident when browsing the trees. The CART methodology have found favour among researchers for application in several areas such as agriculture, medicine, forestry, natural resources management etc. as alternatives to the conventional approaches such as discriminant function method, multiple linear regression, logistic regression etc. In CART, the observations are successively separated into two subsets based on associated variables significantly related to the response variable; this approach

has an advantage of providing easily comprehensible decision strategies. CART can be applied either as a classification tree or as a regressive tree depending on whether the response variable is categorical or continuous. Tree based methods are not based on any stringent assumptions. These methods can handle large number of variables, are resistant to outliers, non-parametric, more versatile, can handle categorical variables, though computationally more intensive. CART can be a good choice for the analysts as they give fairly accurate results quickly, than traditional methods. If more conventional methods are called for, trees can still be helpful if there are a lot of variables, as they can be used to identify important variables and interactions. These are also invariant to the monotonic transformations of the explanatory variables and do not require the selection of the variable in advance as in regression analysis.

1.2 Review

Breiman *et al.*(1984) developed CART (Classification and Regression Trees) which is a sophisticated program for fitting trees to data. A method of choosing multiway partitions for classification and decision trees was given by Biggs *et al.* in 1991 which chooses the best partition on the basis of statistical significance. Breiman, again in 1994, developed the bagging predictors which is a method of generating multiple versions of a predictor and using them to get an aggregated predictor. In the following times, Loh and Shih (1997) developed the QUEST (Quick Unbiased Efficient Statistical Tree) method to take care of the selection bias towards the variables with more possible splits. To increase the statistical reliability of CART, Mola and Siciliano introduced statistical testing approach in the pruning procedure. Chou (1991) proposed an optimal partitioning method in classification tree for categorical explanatory variables with large number of categories based on a k-means clustering procedure. Shih (2001) proposed methods of selecting the best categorical split in a tree based on a family of splitting criterion. These methods were shown to be useful to reduce the computational complexity of the exhaustive search methods. Cappeli *et al.*(2002) suggested the use of statistical significance in the pruning procedure of both classification and regression trees to obtain a statistically reliable tree. Olden and Jackson (2002) provided a comparison between logistic regression analysis, linear discriminant analysis, classification trees and ANN to model fish species distributions and they concluded that classification trees and ANN greatly outperformed traditional approaches. Waheed *et al.*(2006) investigated the potential of hyperspectral remote sensing data of experimental corn plots into categories of water stress, and nitrogen application rates for providing better crop management information in precision farming by using the CART algorithm. The results showed that the accuracy for the irrigation factor was 96% while that of the nitrogen application rate was 83%. Rothwell *et al.* (2008) applied the CART approach to evaluate the key environmental drivers controlling dissolved inorganic nitrogen (DIN) leaching from European forests which successfully classified the sites into the appropriate leaching category.

2. CART methodology

The CART methodology developed by Breiman *et al.*(1984) is outlined here. For building decision trees, CART uses so-called learning set - a set of historical data with pre-assigned classes for all observations. An algorithm known as recursive partitioning is the key to the nonparametric statistical method of CART. It is a step-by-step process by which a decision tree is constructed by either splitting or not splitting each node on the tree into two daughter nodes. An attractive feature of the CART methodology is that because the

algorithm asks a sequence of hierarchical questions, it is relatively simple to understand and interpret the results. The unique starting point of a classification tree is called a root node and consists of the entire learning set $\mathcal{L}$ at the top of the tree. A node is a subset of the set of variables, and it can be terminal or nonterminal node. A nonterminal (or parent) node is a node that splits into two daughter nodes (binary split). Such a binary split is determined by a condition on the value of a single variable, where the condition is either satisfied or not satisfied by the observed value of that variable. All observations in $\mathcal{L}$ that have reached a particular (parent) node and satisfy the condition for that variable drop down to one of the two daughter nodes; the remaining observations at that (parent) node that do not satisfy the condition drop down to the other daughter node. A node that does not split is called a terminal node and is assigned a class label. Each observation in $\mathcal{L}$ falls into one of the terminal nodes. When an observation of unknown class is "dropped down" the tree and ends up at a terminal node, it is assigned the class corresponding to the class label attached to that node. There may be more than one terminal node with the same class label.

To produce a tree-structured model using recursive binary partitioning, CART determines the best split of the learning set L to start with and thereafter the best splits of its subsets on the basis of various issues such as identifying which variable should be used to create the split, and determining the precise rule for the split, determining when a node of the tree is a terminal one, and assigning a predicted class to each terminal node. The assignment of predicted classes to the terminal nodes is relatively simple, as is determining how to make the splits, whereas determining the right-sized tree is not so straightforward. In order to explain these in details, procedure of growing a fully expanded tree and obtaining a tree of optimum size is explained subsequently.

2.1 Tree growing procedure

Let (Y,**X**) be a multivariate random variable where **X** is the vector of K explanatory variables (both categorical and continuous) and Y is the response variable taking values either in a set of classes C(=1,...,j,...,J) or in real line.

Splitting strategy: In determining how to divide subsets of $\mathcal{L}$ to create two daughter nodes from a parent node, the general rule is to make, with respect to the response variable, the data corresponding to each of the daughter nodes "purer" in the sense that the data in each of the daughter nodes is obtained by reducing the number of cases that has been misclassified. For a description of splitting rules, a distinction between continuous and categorical variables is required.

Continuous or numerical variable: For a continuous variable, the number of possible splits at a given node is one less than the number of its distinctly observed values.

Nominal or categorical variable: Suppose that a particular categorical variable is defined by M distinct categories, $l_1,l_2,...,l_M$. The set of possible splits at that node for that variable is the set of all subsets of $\{ l_1,l_2,...,l_M \}$. Denote by τ_L and τ_R the left daughter-node and right daughter-node, respectively, emanating from a (parent) node τ. In general there will be 2^{M-1}-1 distinct splits for an M-categorical variable.

There are several different types of splits that can be considered at each step. For a predictor variable, x_k, which is numerical, a subset of $\mathcal{L}$ can be divided such that one of the

newly created subsets has $x_k \leq s_k$, and the other has $x_k > s_k$, where s_k is some observed value of x_k. While for a predictor variable, x_k, which is nominal or categorical, having class labels belonging to the finite set D_k, a subset of $\mathcal{L}$ can be divided such that one of the newly created subsets has $x_k \in S_k$, and the other has $x_k \notin S_k$, where S_k is a nonempty proper subset of D_k. At each node, the tree-growing algorithm has to decide on which variable it is "best" to split. In order to select best possible split in a node, it is needed to consider every possible split over all variables present at that node, then enumerate all possible splits, evaluate each one, and decide which is best in some sense.

Node impurity function: At each stage of recursive partitioning, all of the allowable ways of splitting a subset of $\mathcal{L}$ are considered, and the one which leads to the greatest increase in node purity is chosen. This can be accomplished using what is called an "impurity function", which is nothing but a function of the proportion of the learning sample belonging to the possible classes of the response variable. To choose the best split over all variables, first the best split for a given variable has to be determined. Accordingly, a goodness of split criterion is defined. The impurity function should be such that it is maximized whenever a subset of $\mathcal{L}$ corresponding to a node in the tree contains an equal number of each of the possible classes (since in that case it is not possible or, is too difficult to sensibly associate that node with a particular class). The impurity function should assume its minimum value for a node that is completely pure, having all cases from the learning sample corresponding to the node belonging to the same class. Two such functions that can serve as the impurity function are the Gini diversity index and the Entropy function.

Let, $\Pi_1, \Pi_2, \ldots, \Pi_K$ be the $K \geq 2$ classes. For node τ, the node impurity function is defined as

$$i(\tau) = \phi(p(1|\tau), \ldots, p(K|\tau)),$$

where $p(k|t)$ is an estimate of $P(X \in \Pi_k | \tau)$, the conditional probability that an observation $\mathbf{X}$ is in Π_k given that it falls into node τ. Under this set up, the Entropy function is given by,

$$i(\tau) = -\sum_{k=1}^{K} p(k|\tau) \log p(k|\tau)$$

When there are only two classes, the entropy function reduces to

$$i(\tau) = -p \log p - (1-p)\log(1-p), \quad \text{where, } p = p(1|\tau)$$

The other impurity function, i.e. the Gini diversity index is defined as,

$$i(\tau) = \sum_{k \neq k'} p(k|\tau) p(k'|\tau) = 1 - \sum_{k} \{p(k|\tau)\}^2$$

In the two class case the Gini index reduces to

$$i(\tau) = 2p(1-p).$$

Gini's index of impurity measure has been used to calculate the node impurity.

Choosing the best split for a variable: To assess the goodness of a potential split, the value of the impurity function can be calculated using the cases in the learning sample corresponding to the parent node, and subtract from this the weighted average of the impurity for the two daughter nodes, with the weights proportional to the number of cases of the learning sample corresponding to each of the daughter nodes, to get the decrease in the overall impurity that would result from the split. To select the way to split a subset of $\mathcal{L}$ in the tree growing procedure, all allowable ways of splitting can be considered, and the one which will result in the greatest decrease in node impurity (or, in other words, greatest increase in the node purity) can be chosen. The splitting procedure is further elaborated here.

Suppose, at node τ, a split s is applied so that a proportion p_L of the observations drops down to the left daughter node τ_L and the remaining proportion p_R drops down to the right daughter node τ_R. For example, suppose there is a dataset in which the response variable Y has two possible values, 0 and 1. Suppose that one of the possible splits of the explanatory variables X_j is $X_j \leq c$ vs. $X_j > c$, where c is some value of X_j. Then the 2×2 table can be prepared as follows

Split	**Class of Y**		**Row total**
	1	0	
$X_j \leq c$	n_{11}	n_{12}	n_{1+}
$X_j > c$	n_{21}	n_{22}	n_{2+}
Column total	n_{+1}	n_{+1}	n_{++}

Consider, first, the parent node τ. If p_L is estimated by n_{+1}/n_{++} and p_R by n_{+2}/n_{++}, and Gini's index is used as the impurity measure, then the estimated impurity function is,

$$i(\tau) = 2\left(\frac{n_{+1}}{n_{++}}\right)\left(1-\frac{n_{+1}}{n_{++}}\right) = 2\left(\frac{n_{+1}}{n_{++}}\right)\left(\frac{n_{+2}}{n_{++}}\right)$$

Now consider the daughter nodes, τ_L and τ_R. For $X_j \leq c$, p_L is estimated by n_{11}/ n_{1+} and p_R by n_{12}/ n_{1+}, and for $X_j > c$, p_L is estimated by n_{21}/ n_{2+} and p_R by n_{22}/ n_{2+}. Then the following two quantities are computed,

$$i(\tau_L) = 2\left(\frac{n_{11}}{n_{1+}}\right)\left(1-\frac{n_{11}}{n_{1+}}\right) = 2\left(\frac{n_{11}}{n_{1+}}\right)\left(\frac{n_{22}}{n_{2+}}\right)$$

The goodness of a split s at node τ is given by the reduction in impurity gained by splitting the parent node τ into its daughter nodes, τ_R and τ_L,

$$\Delta i(\tau) = i(\tau) - \{p_L\, i(\tau_L) + p_R\, i(\tau_R)\} \qquad \ldots (1)$$

The best split for the single variable X_j is the one that has the largest value of $\Delta i(s, \tau)$ over all $s \in S_j$, the set of all possible distinct splits for X_j.

Recursive partitioning: In order to grow a tree, the starting point is the root node, which consists of the learning set $\mathcal{L}$ Using the "goodness of split" criterion for a single variable, the tree algorithm finds the best split at the root node for each of the variables. The best

split *s* at the root node is then defined as the one that has the largest value of (1) over all single-variable best splits at that node.

Next is to split each of the daughter nodes of the root node in the same way. The above computations are repeated for each of the daughter nodes except that this time only the observations in that specific daughter node are considered for the calculations rather than all the observations. When these splits are completed, the splitting is continued with the subsequent nodes. This sequential splitting procedure of building a tree layer-by-layer is called recursive partitioning. If every parent node splits in two daughter nodes, the result is called a binary tree. If the binary tree is grown until none of the nodes can be split any further, then the tree is said to be saturated. It is very easy in a high-dimensional classification problem to let the tree get overwhelmingly large, especially if the tree is allowed to grow until saturation making it unmanageable. One way to counter this type of situation is to restrict the growth of the tree. For example, one can declare a node to be terminal if it fails to be larger than a certain critical size, i.e. if $n(\tau) \leq n_{min}$, where $n(\tau)$ is the number of observations in node τ and n_{min}, is some previously declared minimum size of a node. Because a terminal node cannot be split into daughter nodes, it acts as a brake on tree growth; the larger the value of n_{min}, the more severe the brake. Another early action is to stop a node from splitting by determining whether the largest goodness-of-split value at that node is smaller than a certain predetermined limit. However, it does not work very well to use such sorts of stopping rules to determine that a node should be declared a terminal node and the corresponding subset of $\mathcal{L}$ not split any further, because it can be the case that the best split possible at a certain stage may decrease impurity by only a small amount, but if that split is made, each of the subsets of $\mathcal{L}$ corresponding to both of the descendant nodes can be split to produce an appreciable decrease in impurity. Because of this phenomenon, what works better is to first grow a very large tree, splitting subsets in the current partition of $\mathcal{L}$ even if a split does not lead to an appreciable decrease in impurity. Then a sequence of smaller trees can be created by "pruning" the initial large tree, where in the pruning process, splits that were made are removed and a tree having a fewer number of nodes is produced. This aspect of "pruning" will be discussed in the later sections.

Thereafter, assignment of a class with a terminal node is done by associating a class with each of the terminal node by the rule of majority. Suppose at terminal node τ there are $n(\tau)$ observations, of which $n_k(\tau)$ are from class Π_k, k=1,2,…,K. Then, the class which corresponds to the largest of the $\{n_k(\tau)\}$ is assigned to τ. This is called the plurality rule i.e. the node τ is assigned to class Π_i if $p(i|\tau)=\max_i p(k|\tau)$.

2.2 Estimating the misclassification rate and pruning procedure

The crucial part of creating a good tree-structured classification model is determining how complex the tree should be. If nodes continue to be created until no two distinct values of **X** for the cases in the learning sample belong to the same node, the tree may be overfitting the learning sample and not be a good classifier of future cases. On the other hand, if a tree has only a few terminal nodes, then it may be that it is not making enough use of information in the learning sample, and classification accuracy for future cases will suffer. Initially, in the tree-growing procedure, the predictive accuracy typically increases as more nodes are created and the partition gets finer. But it is usually seen that at some point the misclassification rate for future cases will start to get worse as the tree becomes more complex. In order to compare the prediction accuracy of various tree-structured models,

there needs to be a way to estimate a given tree's misclassification rate for the future observations, which is sometimes referred to as 'generalization error'. Another measure is the 'resubstitution estimate' of the misclassification rate, which is obtained by using the tree to classify the members of the learning sample (that were used to create the tree), and observing the proportion that are misclassified. The resubstitution estimate of the misclassification rate $R(\tau)$ of an observation at node τ is calculated as follows:

$$r(\tau) = 1 - \max_k p(k|\tau)$$

which, for the two class case, reduces to

$$r(\tau) = 1 - \max\left(p, 1-p\right) = \min\left(p, 1-p\right) \qquad (2)$$

However it does not work well to use the resubstitution estimate of the misclassification rate. Because, if no two members of the learning sample have the same value of **X**, then a tree having a resubstitution misclassification rate of zero can be obtained by continuing to make splits until each case in the learning sample is by itself in a terminal node. This may be due to the condition that the class associated with a terminal node will be that of the learning sample case corresponding to the node, and when the learning sample is then classified using the tree, each case in the learning sample will drop down to the terminal node that it created in the tree-growing process, and will have its class match the predicted class for the node. Thus the resubstitution estimate can be a very poor estimate of the tree's misclassification rate for the future observations, since it can decrease as more nodes are created, even if the selection of splits is just responding to "noise" in the data, and not to the real structure. This phenomenon is similar to R^2 increasing as more terms are added to a multiple regression model, with the possibility of R^2 nearing one if enough terms are added, even though more complex regression models can be much worse predictors than simpler ones involving fewer variables and terms.

Let T be the classification tree and let $\tilde{T}=\{\tau_1, \tau_2, \ldots, \tau_L\}$ denote the set of all terminal nodes of T. The misclassification rate for T can now be estimated by,

$$R(T) = \sum_{\tau \in \tilde{T}} R(\tau) P(\tau) = \sum_{l=1}^{L} R(\tau_l) p(\tau_l)$$

for T, where $P(\tau)$ is the probability that an observation falls into node τ. If $P(\tau_l)$ is estimated by the proportion $p(\tau_l)$ of all observations that fall into node τ_l, then, the resubstitution estimate of R(T) is

$$R^{re}(T) = \sum_{l=1}^{L} r(\tau_l) p(\tau_l) = \sum_{l=1}^{L} R^{re}(\tau_l),$$

where, $R^{re}(\tau_l) = r(\tau_l) p(\tau_l)$.

A better estimate of a tree's misclassification rate can be obtained using an independent "test set", which is a collection of cases coming from the same population or distribution as the learning set. Like the learning set, for the test set the true class for each case is known in addition to the values for the predictor variables. The test set estimate of the misclassification rate is just the proportion of the test set cases that are misclassified when

predicted classes are obtained using the tree created from the learning set. The learning set and the test set are both composed of cases for which the true class is known in addition to the values for the predictor variables. Generally, about one third of the available cases should be set aside to serve as a test set, and the rest of the cases should be used as learning set. But sometimes a smaller fraction, such as one tenth, is also used. If one has enough data, using an independent test set is the best thing to do. Otherwise, obtaining a "cross-validation" estimate of the misclassification rate is preferable. For a V-fold cross validation, the entire data is used as learning set, and it is divided into V parts of approximately equal sizes. This is usually done randomly. V is typically taken to be 5 or 10. In many cases it has been seen that a little is to be gained by using a larger value of V, and the larger the value of V, the more is the time taken to build the model. In some situations, the quality of the estimates is reduced by making V too large.

To obtain a cross-validation estimate of the misclassification rate, each of the V groups is in turn set aside to serve temporarily as an independent test set and a tree is grown, according to certain criteria, using the other set consisting of all the other V-1 groups. In all, V trees are grown in this way, and for each tree the set aside portion of the data are used to obtain a test sample estimate of the tree's misclassification rate. Then the V test set estimates are averaged to obtain the misclassification rate for the tree grown from the entire learning set using the same criteria. Regardless of the method used to estimate the misclassification rate of the classification tree model, it still remains a matter of concern, how to grow the best tree, or how to create the set of candidate trees from which the best one can be selected based on their estimated misclassification rate. As mentioned previously, a stopping rule does not work very well to determine the size of the tree and it seem better to grow a fully expanded tree and then "prune" it back or, remove some of its nodes by some rule to produce a tree with fewer number of terminal nodes. Pruning or removing of nodes from a tree thus produces a finite sequence of nested subtrees, since the first tree produced by pruning is a subtree of the original tree, and a second pruning step creates a subtree of the first subtree, and so on. The accuracies of the members of this sequence of subtrees are then compared using good estimates of their misclassification rates (either based on a test sample or obtained by cross-validation), and the best performing tree in the sequence is chosen as the final model.

Pruning procedure: A specific way to create a useful sequence of different-sized trees is to use "minimum cost-complexity pruning". In this process, a nested sequence of subtrees of the initial large tree is created by "weakest-link cutting". With weakest-link cutting (pruning), all of the nodes that arise from a specific nonterminal node are pruned off (leaving that specific node itself as terminal node), and the specific node selected is the one for which the corresponding pruned nodes provide the smallest per node decrease in the resubstitution misclassification rate. If two or more choices for a cut in the pruning process would produce the same per node decrease in the resubstitution misclassification rate, then pruning off the largest number of nodes is preferred. In some cases (minimal pruning cases), just two daughter terminal nodes are pruned from a parent node, making it a terminal node. But in other cases, a larger group of descendant nodes are pruned all at once from an internal node of the tree.

Instead of using the resubstitution measure $R^{re}(\tau)$ as the estimate of R(T), it is modified for tree pruning. Let $\alpha \geq 0$ be a complexity parameter. For any node $\tau \in T$, the cost-complexity measure $R_\alpha(\tau)$ is given by,

$$R_\alpha(T) = R^{re}(T) + \alpha \qquad (3)$$

From (3), a cost-complexity pruning measure for a tree T is defined as

$$R_\alpha(T) = \sum_{l=1}^{L} R_\alpha(T_l) = R^{re}(T) + \alpha|\tilde{T}| \qquad (4)$$

where, $|\tilde{T}| = L$ is the number of terminal nodes in the subtree T, which is a measure of tree complexity, and α is the contribution to the measure for each terminal node. One can think of $\alpha|\tilde{T}|$ as a penalty term for tree size, so that $R_\alpha(T)$ penalizes $R^{re}(T)$ for generating too large a tree. For each α, the subtree T(α) of T_{max} that minimizes $R_\alpha(T)$, is selected. To minimize this measure, for small values of α, trees having a large number of nodes, and a low resubstitution estimate of misclassification rate, will be preferred. Thus, the value of α determines the size of the tree. When α is very small, the penalty term will be small, and so the size of the minimizing subtree T(α), which will essentially be determined by $R^{re}(T(\alpha))$, will be large. For large enough values of α, a one node tree will minimize the measure. For example, suppose α is set to zero, i.e. α=0 and the tree T_{max} is grown so large that each terminal node contains only a single observation; then, each terminal node takes on the class of its solitary observation, every observation is classified correctly, and $R^{re}(T_{max})=0$. So, T_{max} minimizes $R_0(T)$. As the value of α is increased, the minimizing subtree T(α) will have fewer and fewer terminal nodes. When α is very large, it results in a tree having only the root node. Since the resubstitution estimate of misclassification rate is generally overoptimistic and becomes unrealistically low as more nodes are added to a tree, it is expected that there is some value of α that properly penalizes the overfitting of a tree which is too complex, so that the tree which minimizes $R_\alpha(T)$, for the proper value of α, will be a tree of about the right complexity (to minimize the misclassification rate of the future observations). Even though the proper value of α is unknown, utilization of the weakest-link cutting procedure explained earlier guarantees that for each value of α≥0, a subtree of the original tree that minimizes $R_\alpha(T)$ will be a member of the finite nested sequence of subtrees produced. It is worth noting that although α is defined on the interval $[0,\infty)$, the number of subtrees of T is finite. Suppose that, for $\alpha=\alpha_1$, the minimizing subtree is $T_1=T(\alpha_1)$. As the value of α is increased, T_1 continues to be the minimizing subtree until a certain point, say, $\alpha=\alpha_2$, is reached, and a new subtree, $T_2=T(\alpha_2)$, becomes the minimizing subtree. As α is increased further, the subtree T_2 continues to be the minimizing subtree until a value of α is reached, $\alpha=\alpha_3$, say, when a new subtree $T_3=T(\alpha_3)$ becomes the minimizing subtree. This argument is repeated a finite number of times to produce a sequence of minimizing subtrees $T_1, T_2, T_3, \ldots$.

The above discussion states that a finite increasing sequence of complexity parameters,

$$0 = \alpha_0 < \alpha_1 < \alpha_2 < \alpha_3 < \cdots < \alpha_M$$

corresponds to a finite sequence of nested subtrees, say, M in number, of the fully grown tree,

$$T_{max} = T_0 > T_1 > T_2 > \cdots > T_M$$

2.3 Selecting the right sized tree among the candidate sub-trees

The sequence of subtrees produced by the pruning procedure serves as the set of candidate subtrees for the model, and to obtain the classification tree, all that remains to be done is to

select the one which will hopefully have the smallest misclassification rate for future observations. The selection is based on estimated misclassification rates, obtained using a test set or by cross validation.

Independent test set selection: If an independent test set is available, it is used to estimate the error rates of the various trees in the nested sequence of subtrees, and the tree with minimum estimated misclassification rate can be selected to be used as the tree-structured classification model. For this purpose, the observations in the learning dataset ($\mathcal{L}$) are randomly assigned to two disjoint datasets, a training dataset ($\mathcal{D}$) and a test set ($\mathcal{T}$), where $\mathcal{D} \cap \mathcal{T} = \Phi$. Suppose there are $n_{\mathcal{T}}$ observations in the test set and that they are drawn independently from the same underlying distributions as the observations in $\mathcal{D}$. Then the tree T_{max} is grown from the learning set only, and it is pruned from bottom up to give the sequence of subtrees $T_1 > T_2 > \cdots > T_M$, and a class is assigned to each terminal node.

Once a sequence of subtrees has been produced, each of the $n_{\mathcal{T}}$ test-set observations are dropped down the tree T_k. Each observation in $\mathcal{T}$ is then classified into one of the different classes. Because the true class of each observation in $\mathcal{T}$ is known, $R(T_k)$ is estimated by $R^{ts}(T_k)$, which is (4) with $\alpha=0$; i.e., $R^{ts}(T_k)=R^{re}(T_k)$, the resubstitution estimate computed using the independent test set. When the costs of misclassification are identical for each class, $R^{ts}(T_k)$ is the proportion of all test set observations that are misclassified by T_k. These estimates are then used to select the best pruned subtree T_* by the rule,

$$R^{ts}(T_*)=\min_k R^{ts}(T_k)$$

and $R^{ts}(T_*)$ is its estimated misclassification rate.

A popular alternative is to recognize that since all of the error rates are not accurately known, but only estimated, it could be that a simpler tree with only a slightly higher estimated error rate is really just as good as or better than the tree having the smallest estimated error rate.

The standard error of $R^{ts}(T)$ is estimated as follows. When test set observations are dropped down the tree T, the chance that any one of these observations are misclassified is $p^*=R(T)$. Thus, it is a binomial sampling situation with $n_{\mathcal{T}}$ Bernoulli trials and probability of success p^*. If $p=R^{ts}(T)$ is the proportion of misclassified observations in $\mathcal{T}$, then p is unbiased for p^* and the variance of p is $p^*(1-p^*)/n_{\mathcal{T}}$. The standard error of $R^{ts}(T)$ is, therefore, estimated by,

$$\widehat{SE}\left(R^{ts}(T)\right)=\left\{\frac{R^{ts}(T)(1-R^{ts}(T))}{n_T}\right\}^{1/2}$$

Cross-validation error estimate: Use of cross-validation rather than test-sample for estimating the misclassification error makes things more complicated. In V-fold cross-validation (CV/V), the learning dataset is randomly divided into V roughly equal sized, disjoint subsets. The first of the V portions of data is set aside to serve as a test set for a sequence of trees created from the other V-1 portions of the data. These V-1 portions of the data are collectively used to grow a large tree, and then the same pruning process is applied to create a sequence of subtrees. Each subtree in the sequence is the optimal subtree in the sequence, according to the $R_\alpha(T)$ criterion. Similar sequences of subtrees are created and the corresponding values of α for which each tree is optimal are determined, by setting

aside, one at a time, each of the other V-1 portions of the data, resulting in V sequences in all. Then, for various values of α, cross-validation estimates of the misclassification rates of the corresponding trees created using all of the data are determined as follows: for a given value of α the misclassification rate of the corresponding subtree in each of the V sequences is estimated using the associated set aside portion as a test set, and the V estimates are averaged to arrive at a single error rate estimate corresponding to that value of α, and this estimate of serves as the estimate of the true misclassification rate for the tree created using all of the data and pruned using this value of α. Finally, these cross-validation estimates of the error rates for the trees in the original sequence of subtrees are compared, and the subtree having the smallest estimated misclassification rate is selected to be the final tree-based classification model.

For a V-fold cross-validation (CV/V), the learning dataset $\mathcal{L}$ is randomly divided into V roughly equal sized, disjoint subsets, $\mathcal{L}=\bigcup_{v=1}^{V}\mathcal{L}_v$, where, $\mathcal{L}_v \bigcap \mathcal{L}_{v'}=\emptyset, v\neq v'$, and V is usually taken to be 5 or 10. Next, V different datasets are obtained from the $\{\mathcal{L}_v\}$ by taking $\mathcal{D}_v=\mathcal{L}-\mathcal{L}_v$ as the v-th training set and $\mathcal{T}_v=\mathcal{L}_v$ as the v-th test set, v=1,2,…,V. The v-th tree $T^{(v)}_{max}$ is grown using the v-th training set $\mathcal{D}_v$, v=1,2,…,V. The value of the complexity parameter α is fixed to a certain value. Let, $T^{(v)}(\alpha)$ be the best pruned subtree of $T^{(v)}_{max}$. Now, each observation in the v-th test $\mathcal{T}_v$ is dropped down the $T^{(v)}(\alpha)$, v=1,2,…,V. Let, $n_{ij}^{(v)}(\alpha)$ be the number of j-th class observations in $\mathcal{T}_v$ that are classified as being from the i-th class, i,j=1,2,…,J. Because, $\mathcal{L}=\bigcup_{v=1}^{V}\mathcal{T}_v$ is a disjoint sum, the total number of j-th class observations that are classified as being from the i-th class is

$$n_{ij}(\alpha)=\sum_{v=1}^{V} n_{ij}^{(v)}(\alpha), i,j=1,2,\dots,J .$$

If Nj is the number of cases observations in L that belong to the j-th class, j=1,2,…,J, and assuming equal misclassification cost for all classes, then for a given α,

$$R^{CV/V}(T(\alpha))=N^{-1}\sum_{i=1}^{J}\sum_{j=1}^{J} n_{ij}(\alpha)$$

is the misclassification rate over $\mathcal{L}$, where, T(α) is a minimizing subtree of T_{max}. The final step in this process is to find the right sized subtree. For different values of α, $R^{CV/V}$ is evaluated. If for a sequence of values α_k, corresponding cross-validation error of the minimizing subtree $T(\alpha)=T_k$ is given by,

$$R^{CV/V}(T_k)=R^{CV/V}(T(\alpha_k))$$

Then, the best-pruned subtree T_* is selected by the rule,

$$R^{CV/V}(T_*)=\min_k R^{CV/V}(T_k)$$

and $R^{CV/V}(T_*)$ is used as its estimated misclassification rate.

References:

Biggs, D., Ville, B.D., and Suen, E. (1991). A method of choosing multiway partitions for classification and decision trees. *Journal of Applied Statistics*,18(1),49-62.

Breiman, L., Freidman, J.H., Olshen, R.A. and Stone, C.J. (1984). *Classification and regression trees*. Wadsworth, Belmont CA.

Cappeli, C., Mola, F., and Siciliano, R. (2002). A statistical approach to growing a reliable honest tree. *Computational statistics and data analysis*, 38. 285-299.

Chou, P. A. (1991). Optimal partitioning for classification and regression trees, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4), 340-354.

Loh, W.Y. and Shih, Y.S. (1997). Split selection methods for classification trees. *Statistica Sinica*, 7, 815-840.

Olden, J.D. and Jackson, D.A. (2002). A comparison of statistical approaches for modelling fish species distributions. *Freshwater biology*, 47(10). 1976-1995.

Rothwell, J.J., Futter, M.N., and Dise, N.B. (2008). A classification and regression tree model of controls on dissolved inorganic nitrogen leaching from European forests. *Environmental Pollution*, 156(2), 544-552.

Shih, Y-S. (2001). Selecting the best categorical split for classification trees, *Statisics and Probablity Letters*, 54, 341-345.

Waheed, T., Bonnell, R. B., Prasher, S. O. and Paulet E. (2006). Measuring performance in precision agriculture : CART-A decision tree approach. *Agricultural Water Management*, 84. 173-185.