# Regression and Classification Trees

Prof. Thomas B. Fomby
Department of Economics
Southern Methodist University
March 2008

The basic idea of regression and classification trees is build a step function characterization of the relationship between a set of input variables and a target output variable. This is done by partitioning the input space into rectangular regions with the predicted value of the output variable being the average of the output values in each region. In purpose, it is hoped that the chosen step function will characterize (predict) well the output values in independent data sets.

Let us be more specific about the step function characterization of output values. Let $(\mathbf{x}_i, y_i)$, denote observations on the p input variables $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{ip})$ and corresponding output values $y_i$, $i = 1,2,\cdots,N$. Assume that we have M distinct regions $R_1, R_2, \cdots, R_M$ that partition the input space $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{ip})$. Regression (Classification) trees model the response $y$ as a constant $c_m$ in each region:

$$f(\mathbf{x}) = \sum_{m=1}^{M} c_m I(\mathbf{x} \in R_m) .$$

If we adopt the criterion of minimizing the sum-of-squared errors $\sum_i (y_i - f(\mathbf{x}_i))^2$, it turns out that the best estimate of $c_m$ in each region is $\hat{c}_m = ave(y_i \mid \mathbf{x}_i \in R_m)$. Finding the best partition of the input space is accomplished by the method of **recursive binary partitioning**. Starting with all of the data, consider a splitting (input) variable j and split point s, and define the pair of half-planes

$$R_1(j,s) = \{\mathbf{x} \mid x_j \le s) \text{ and } R_2(j,s) = \{\mathbf{x} \mid x_j > s\} .$$

Then we seek the splitting variable j and split point s that solve the following minimization problem:

$$\min_{i,j}\left[\min_{c_1} \sum_{\mathbf{x}_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{\mathbf{x}_i \in R_2(j,s)} (y_i - c_2)^2\right].$$

For any choice of j and s, the inner minimization is solved by

$$\hat{c}_1 = ave(y_i \mid \mathbf{x}_i \in R_1(j,s)) \text{ and } \hat{c}_2 = ave(y_i \mid \mathbf{x}_i \in R_2(j,s)) .$$

Having found the best split, we partition the input data into the two resulting regions and search separately over these two regions for a best splitting variable j and splitting point s that would produce 3 regions whose average output values produce the greatest possible reduction in the sum of squared errors over the three regions. Then, given these three regions, we search separately over these three regions for a best splitting variable j and splitting point s whose average output values produce the greatest possible reduction in the sum of squared errors over the four regions. This process continues until a "full" tree is built. This process is called the **recursive binary partitioning** process.

[Some pictures in here depicting the recursive search procedure.]

## Pruning

The idea behind pruning is that a very large tree is likely to overfit the training data set in the sense that if a "full" tree is used to score an independent data set, say the validation data set, its accuracy would be much attenuated. There are essentially two ways of pruning a full tree: (1) One can use the validation data set to score various sized trees and thereby choose the "pruned" tree (i.e. a simplified tree with fewer decision nodes than the full tree) that produces the greatest accuracy of prediction or classification in the validation data set. (2) One can limit the growth of the tree by choosing a tuning parameter that can potentially limit the size of a grown tree. Two popular **tuning parameters** are **the minimum number of cases that are required of all final nodes** and the **maximum number of decision nodes that are allowed in building the tree**. XLMINER supports the use of either of these tuning parameters. Here we will focus on the use of the minimum number of cases that are required of all final nodes. We will represent this minimal number by the parameter $n_t$ .