Spring 2015 Statistics 151a (Linear Models) : Lecture Twenty Four

Aditya Guntuboyina

23 April 2015

1 Regression and Classification Trees

We have so far looked at *generalized linear models* for regression. Simple non-linear models are also quite often used. We shall briefly look at tree-based methods.

The basic idea behind these methods is the following. Group the n subjects into a bunch of groups based solely on the explanatory variables. For example, in the seatpos dataset, a possible grouping could be:

- Group 1: Leg < 35.4
- Group 2: Leg ≥ 35.4 and Leg $\geq 37.\ 9$
- Group 3: Leg ≥ 35.4 and Leg < 37.9

Calculate the mean value of the response variable for each group. For example, in the seatpos dataset, for the above grouping, the mean value of the response (hipcenter) equals -107.3 for Group 1, -220.5 for Group 2 and -179.8 for Group 3. Prediction for a future subject is done in the following way. Look at the explanatory variable values for the future subject to figure which group he belongs to. Then predict his response value by the mean response for his group.

When the response is binary, the mean value of the response in each group will be between 0 and 1 and can be interpreted as a probability. The predictions provided by the method can therefore be interpreted as predictions of the probability that the response variable equals 1.

The main thing to understand here is how the grouping is constructed. Finding the *best* grouping is a computationally challenging task. In practice, a greedy algorithm, called *Recursive Partitioning*, is employed which produces a reasonable albeit not the best grouping. Let us first understand this for regression trees. The ideas for classification trees are similar.

1.1 Recursive Partitioning for Regression Trees

The grouping produced by Recursive Partitioning can be nicely represented by a tree. This tree is relatively easy to interpret. How is this constructed?

At each node of the tree, there is a variable and a cut-off. How to choose the variable and how to choose the cut-off?

Given a variable X_j and a cut-off c, we can divide the subjects into two groups: G_1 given by $X_j \leq c$ and G_2 given by $X_j > c$. The *deviance* of this split is defined as:

$$RSS(j,c) := \sum_{i \in G_1} (y_i - \bar{y}_1)^2 + \sum_{i \in G_2} (y_i - \bar{y}_2)^2$$

where $\bar{y}_1$ and $\bar{y}_2$ denote the mean values of the response in the groups G_1 and G_2 respectively.

The values of j and c for which $RSS(j,c)$ is the smallest give the best split. The quantity $\min_{j,c} RSS(j,c)$ should be compared with $TSS = \sum_i (y_i - \bar{y})^2$. The ratio $\min_{j,c} RSS(j,c)/TSS$ is always smaller than 1 and the smaller it is, the greater we are gaining by the split.

The recursive partitioning algorithm for constructing the regression tree proceeds as follows: Find j and c such that $D(j,c)$ is the smallest. Use the jth variable and the cut-off c to divide the data into two groups: G_1 given by $X_j \leq c$ and G_2 given by $X_j > c$. Repeat this process within each group separately.

For a tree T, if we denote the groups generated by $G_1, \ldots, G_k$, its RSS can be defined as

$$RSS(T) := \sum_{j=1}^{m} \sum_{i \in G_j} (y_i - \bar{y}_j)^2$$

where $\bar{y}_1, \ldots, \bar{y}_m$ denote the mean values of the response in each of the groups.

How large should the tree be grown? In other words, when should the above process terminate? Very large trees obviously lead to over-fitting. One strategy is to stop when $\min_{j,c} RSS(j,c)/TSS$ for a group is not small. This would be the case when we are not gaining all that much by splitting further. This is actually not a very smart strategy. Why? Because incremental improvements due to each expansion of the tree may not necessarily always be decreasing.

In practice, one uses an AIC-type method (called cost-complexity pruning) to decide on an optimal tree length. A very large tree T_0 is first grown (until some minimum node size, say 5, is reached) and then, one selects an appropriate subtree of T_0 by cost-complexity pruning. For a tree T, let us denote by $|T|$ its number of terminal nodes. This is also the number of groups that the tree generates. $|T|$ is large for large trees. The cost-complexity criterion for a subtree T of T_0 is defined by

$$C_\alpha(T) = RSS(T) + \alpha\ TSS(|T|).$$

Choose the subtree T_α of T_0 which minimizes $C_\alpha(T)$. α governs the trade-off between goodness of fit and tree size. Large values of α result in smaller trees and small values give large trees. $\alpha = 0$ gives T_0. In practice, α is chosen according to cross-validation. α is referred to as cp in R.

1.2 Classification Trees

We looked at regression trees in the class. The idea behind classification trees is similar.

The classification tree is constructed top down. Given a variable X_j and a cut-off c, the subjects are divided into the two groups G_1 where $X_j \leq c$ and G_2 where $X_j > c$. We measured the efficiency of this split by the RSS:

$$RSS(j,c) := \sum_{i \in G_1} (y_i - \bar{y}_1)^2 + \sum_{i \in G_2} (y_i - \bar{y}_2)^2$$

where $\bar{y}_1$ and $\bar{y}_2$ denote the mean values of the response in the Groups G_1 and G_2 respectively. In classification problems, the response values are 0 or 1. Therfore $\bar{y}_1$ equals the proportion of ones in G_1 while $\bar{y}_2$ equals the proportion of ones in G_2. It is better to denote $\bar{y}_1$ and $\bar{y}_2$ by $\bar{p}_1$ and $\bar{p}_2$ respectively.

The formula for $RSS(j,c)$ then becomes:

$$RSS(j,c) = n_1 \bar{p}_1 (1 - \bar{p}_1) + n_2 \bar{p}_2 (1 - \bar{p}_2).$$

This quantity is also called the Gini index of the split corresponding to the jth variable and cut-off c. The function $p(1-p)$ takes its largest value at $p = 1/2$ and it is small when p is close to 0 or 1. Therefore the quantity $n_1\bar{p}_1(1-\bar{p}_1)$ is small if either most of the response values in the group G_1 are 0 (in which case $\bar{p}_1 \approx 0$) or when most of the response values are 1 (in which case $\bar{p}_1 \approx 1$). A group is said to be pure if either most of the response values in the group are 0 or if most of the response values are 1. Thus the quantity $n_1\bar{p}_1(1-\bar{p}_1)$ measures the impurity of a group. If $n_1\bar{p}_1(1-\bar{p}_1)$ is low, then the group is pure and if it is high, it is impure. The group is maximally impure if $\bar{p}_1 = 1/2$.

The Gini Index, $RSS(j, c)$, is the sum of the impurities of the groups given by $X_j \leq c$ and $X_j > c$. The recursive partitioning algorithm determines j and c such that $RSS(j, c)$ is the smallest. This divides the data into two groups with $X_j \leq c$ and $X_j > c$. The process is then continued within each of these groups separately.

The quantity $n_1\bar{p}_1(1-\bar{p}_1)$ is not the only function used for measuring the impurity of a group in classification. The key property of the function $p \mapsto p(1-p)$ is that it is symmetric about $1/2$, takes its maximum value at $1/2$ and it is small near the end points $p = 0$ and $p = 1$. Two other functions having this property are:

1. **Cross-entropy or Deviance**: Defined as $-2n_1\left(\bar{p}_1 \log \bar{p}_1 + (1-\bar{p}_1)\log(1-\bar{p}_1)\right)$. This also takes its smallest value when $\bar{p}_1$ is 0 or 1 and it takes its maximum value when $\bar{p}_1 = 1/2$.
2. **Misclassification Error**: This is defined as $n_1 \min(\bar{p}_1, 1-\bar{p}_1)$. This equals 0 when $\bar{p}_1$ is 0 or 1 and takes its maximum value when $\bar{p}_1 = 1/2$. If we classify all the response values in the group G_1 by the majority value, then the number of response values misclassified in G_1 equals $n_1 \min(\bar{p}_1, 1-\bar{p}_1)$. This explains the name misclassification error.

One can use Deviance or Misclassification error instead of the Gini index while growing a classification tree. The default in R is to use the Gini index.

The remaining aspects of classification trees work in exactly the same way as that of regression trees.