Tree Based Models: Regression Tree, Boosting, Random Forest

Aramayis Dallakyan[1] [2]

[1]Agribusiness teaching Center
Armenia

2019

[2]All errors are my own. armopost@yahoo.com

Outline

Introduction

Introduction

- Recall our main gall is to estimate $\hat{f}(x)$ from $y = f(x) + \epsilon$.
- Last lecture we mostly focused on linear models, where $f(x)$ is linear function.
- Linear models are relatively simple to describe and implement, and the main advantage is it is easy to interpret and make inference.
- However, standard linear regression may have significant limitation in terms of **predictive power**.
- This happens, since the linearity assumption is almost always an approximation.
- By relaxing linearity assumption, one may achieve better results.

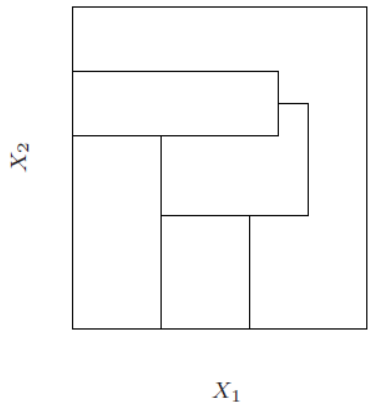
Introduction

- In this lecture, we describe *tree-based* methods for regression problem. The classification problem will be covered later.
- The main idea of this method involve **stratifying** or **segmenting** the predictor space into a number of simple regions. (Recall how you integrate any given function)
- Then given an observation, the prediction is made by use of the mean of the training observations in the region to which it belongs.
- To make tree-methods more competitive, hence we will introduce **random forest** and **boosting** techniques, which involve producing multiple trees to improve prediction accuracy.

Example

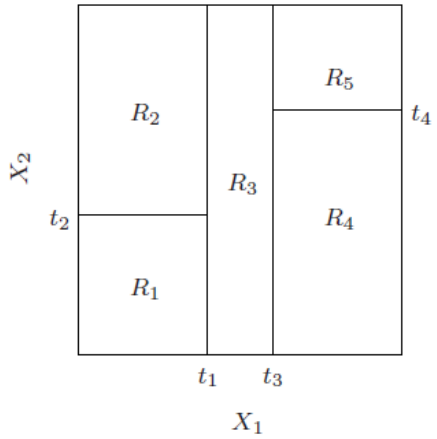

Decision Tree Regression
target
data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
1
2
3
4
5

- Consider a regression problem with *continuous* response Y and inputs X_1 and X_2.
- One possible partition of feature space. (Recall we call X_1 and X_2 as features.)



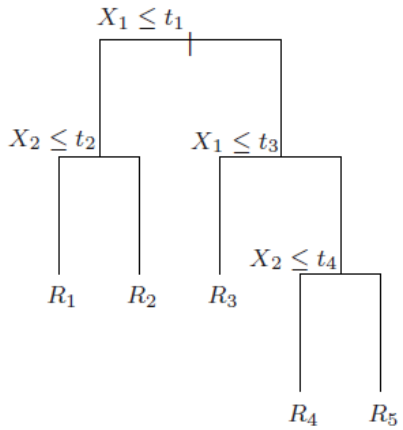
- In each partition we can model Y with a different constant.
- Even though the partitioning line has a simple description $X_1 = c$, some of the resulting regions are complicated to describe.

- To simplify, we restrict attention to **recursive** binary partitions.



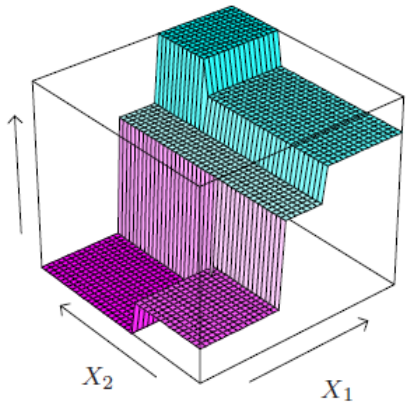
- Basically, we split the space into two regions, and model the response by the means of Y in each region.
- To do this, we need to choose **the variable** and $split - point$ in orser to achieve the **the best fit**.
- Then we continue in the same manner by splitting one or both of these regions into two more regions.
- The process is continued until some **stopping rule** is applied.
- Thus we partition space into the five regions $R_1, R_2, \ldots, R_5$.

- The same model can be represented by the binary tree.



- The full datasets sits at the top of the tree. Observations satisfying the condition at each junction are assigned to the left branch, and the others to the right branch.
- The terminal nodes or leaves of the tree correspond to the regions $R_1, R_2, \ldots, R_5$

- The estimated regression surface will look like something like in the figure below.



- If you are interested what kind of function it is, then it is given by

$$\hat{f}(X) = \sum_{m=1}^{5} c_m I\{(X_1, X_2) \in R_m\},$$

where c_m is some constant, usually the mean value of region R_m, and $I(\cdot)$ is identity function.

Now we introduce the procedure of building a regression tree. Then describe how to implement each step.

1. We devide the predictor space- that is, the set of possible values for $X_1, X_2, \ldots, X_p$ - into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
2. For every observation that falls into the region R_J, we make the same prediction, which is simply the mean of the response values for the training observations in R_j

For example suppose that in Step 1 we obtain two regions, R_1 and R_2, and that the response mean of the training observations in the first region is 10, while the response mean of the training observations in the second region is 20. Then for a given observation $X = x$, if $x \in R_1$ we will predict a value of 10, and if $x \in R_2$ we will predict a value of 20.

- We know give details on how to construct the regions $R_1, \ldots, R_J$. That is how to **grow a regression tree**.
- Our data consists of inputs and a continuous response, for each of N observations :i.e x_i, y_i for $i = 1, 2, \ldots, N$, with $x_i = (x_{i1}, \ldots, X_{ip})$.
- We need from algorithm automatically decide on the
  - Splitting variables and split points
  - Topology(shape) the tree should have.
- Basically, we want from algorithm select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the greatest possible reduction in RSS.

- That is we consider all predictors $X_1, \ldots, X_p$, and all possible values of the cutpoint s for each predictors, and then choose the predictor and cutpoint such that the resulting tree has the lowest RSS
- That is for any j and s, we define the pair of half-plane

$$R_1(j,s) = \{X | X_j < s\} \text{ and } R_2(j,s) = \{X | X_j \geq s\}$$

and we seek value of j and s that minimize the equation

$$RSS = \sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

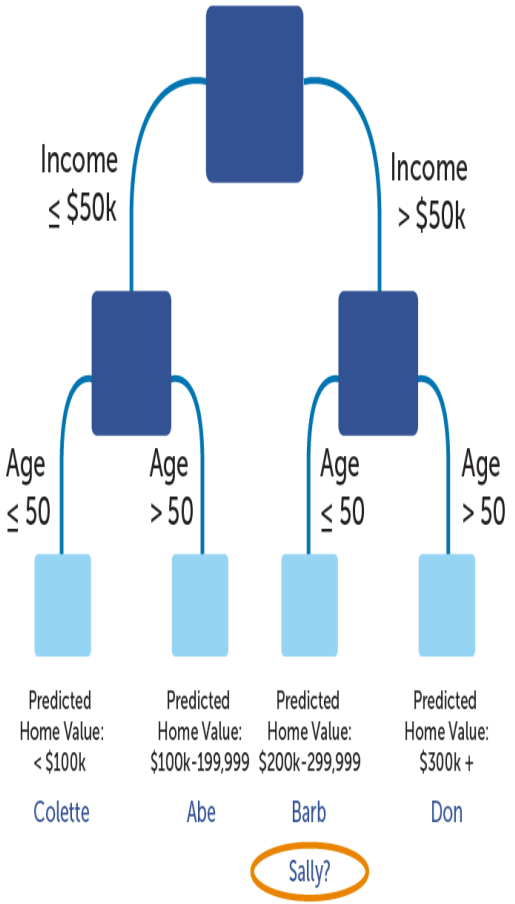
where $\hat{y}_{R_1}$ is the mean response for the observations in $R_1(j,s)$, and $\hat{y}_{R_2}$ is the mean response in $R_2(j,s)$.

- Next, the process is repeated to look for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions.

target indicator

	Age	Income	Home Value
Abe	70	$30,000	$150,000
Barb	30	$70,000	$250,000
Colette	25	$45,000	$98,000
Don	57	$90,000	$300,000
Sally	33	$55,000	$175,000

training data



- Once the regions $R_1, \ldots, R_j$ have been created, we predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs.
- Now the question is: **How large should we grow the tree?**
- Since the big tree might overfit the training data, while the small tree might not capture the important structure.
- The idea is to choose the optimal tree size adaptively from the data, i.e consider the tree-size as a **tuning prameter.**

- We start by growing a large tree T_0.
- We define a subtree $T \subset T_0$ to be any tree that can be obtained by pruning T_0, i.e by collapsing any number of its internal nodes.
- Then for each α, we find subtree $T_\alpha \subset T_0$ to minimize

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|,$$

  where $|T|$ indicates the number of terminal nodes and $\hat{y}_{R_m}$ is the predicted response associated with R_m.
- The tuning parameter α controls trade-off between the subtree's complexity and its fit to the training data, i.e as α increase we pay price for having a tree with many terminal nodes. (**Does this sound familiar?**)

Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Bagging

- The idea of bagging is to run multiple Regression Trees and then average the final result to improve the variance and prediction.
- That is we generate B different **bootstrapped** training data sets and then construct B regression tree for each dataset.
- Then average the resulting predictors.

Random Forest

- Random Forest provides an improvement over bagged trees by so-called *decorrelating* the trees.
- The idea is following, we build a number of decision trees on bootstrapped training samples but when building this decision trees each time a split in a tree is considered, *a random sample* of m predictors is chosen to split candidates from the full set of p predictors.
- Typical choice of m is $m = \sqrt{p}$
- This procedure improves the prediction accuracy. For details see Chapter 8 of ISLR.

Variable Importance

- RF and bagging typically results in improved accuracy over prediction using a single tree.
- Unfortunately, however, it can be difficult to interpret the resulting model. Recall that one of the advantages of decision trees is the attractive and easily interpreted diagram
- However, when we bag a large number of trees, it is no longer possible to represent the resulting statistical learning procedure using a single tree, and it is no longer clear which variables are most important to the procedure.
- Thus, bagging improves prediction accuracy at the expense of interpretability.

- Although the collection of bagged trees is much more difficult to interpret than a single tree, one can obtain an overall summary of the importance of each predictor using the RSS
- We can measure importance by recording the total amount that the RSS is decreasing due to splits over a given predictor, averaged over all B trees.
- A large value indicates an important predictor and vice verse.
- We will see example when we do actual data analysis

Advantages of Trees

- Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Trees can easily handle qualitative predictors without the need to create dummy variables.

Disadvantages

- Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.
- Additionally, trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree.
- Fortunately, by aggregating many decision trees, using methods like bagging, and random forests, the predictive performance of trees can be substantially improved.