MONASH University

ETC3250

Business Analytics

Week 9.
Tree-Based Methods

18 September 2017

Building a regression tree

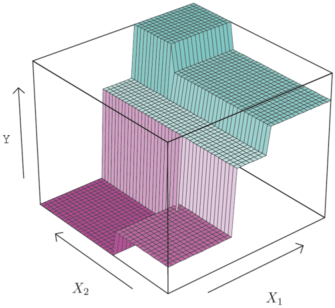
- We divide the predictor space—that is, the set of possible values for $X_1, X_2, \ldots, X_p$—into J **distinct** and **non-overlapping** regions, $R_1, R_2, \ldots, R_J$.
- The regions could have any shape. However, for simplicity and for ease of interpretation, we divide the predictor space into high-dimensional **rectangles**.
- We model the response as a constant c_j in each region

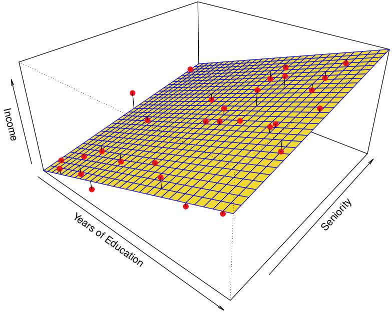
$$f(x) = \sum_{j=1}^{J} c_j \, I(x \in R_j)$$

Trees Versus Linear Models

$$f(X) = \sum_{m=1}^{M} c_m \, I(X \in R_m)$$

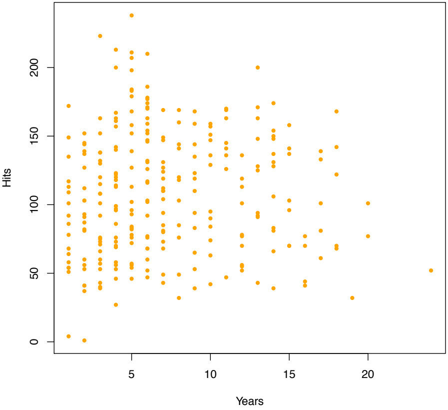
$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$



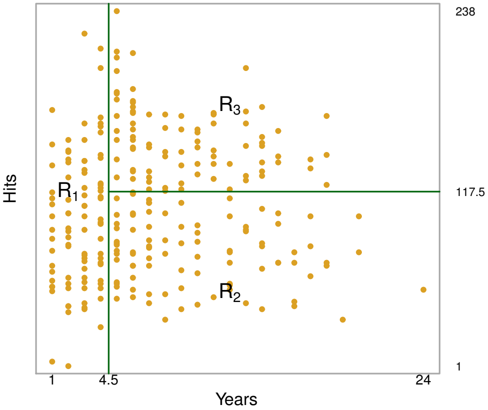


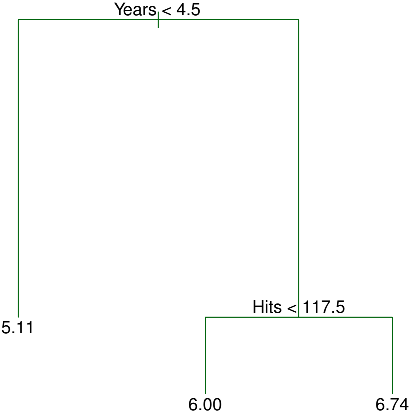
Predicting a baseball player's salary

- Y: log salary of a baseball player (in thousands of dollars)
- X_1: number of years played in the major leagues
- X_2: number of hits made in the previous year



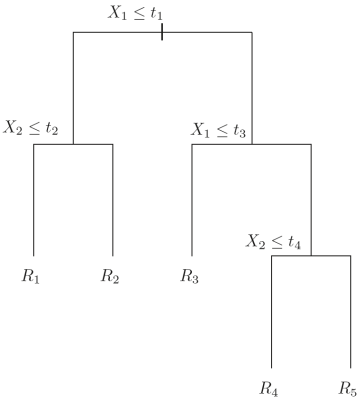
Predicting a baseball player's salary

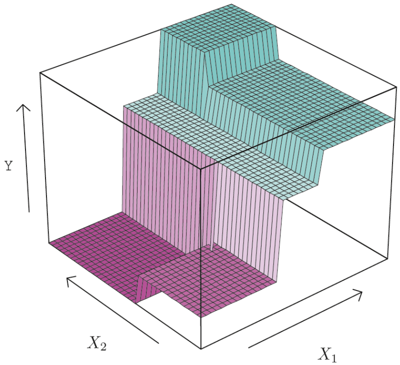




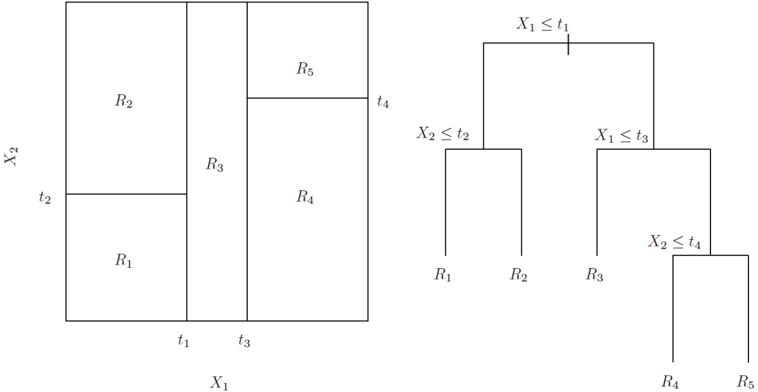
- $\boldsymbol{R_1} = \{X | Years < 4.5\}$
- $\boldsymbol{R_2} = \{X | Years \geq 4.5, Hits < 117.5\}$
- $\boldsymbol{R_3} = \{X | Years \geq 4.5, Hits \geq 117.5\}$

Regression tree



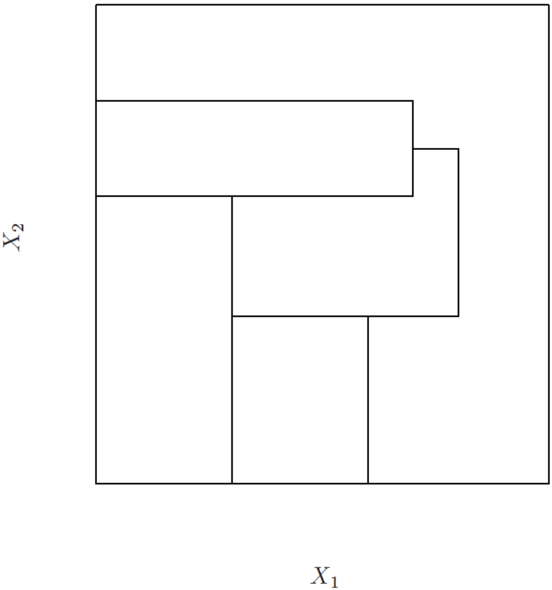


Tree-Based Methods



- $R_1, R_2, \ldots,$ and R_5 are **terminal nodes** or **leaves**.
- The points where we split are **internal nodes**.
- The segments that connect the nodes are **branches**.

Not all partitions are possible



Building a regression tree

1. Given a partition $R_1, R_2, \ldots, R_J$, what are the optimal values of c_j if we want to minimize $\sum_i (y_i - f(x_i))^2$?
2. How do we construct the regions $R_1, ..., R_J$?

1. The best c_j is just the average of y_i in region R_j:

$$\hat{c}_j = \text{average}(y_i | x_i \in R_j).$$

2. Finding the best binary partition in terms of minimum sum of squares is generally *computationally infeasible*. For this reason, we take a *top-down*, *greedy* approach that is known as **recursive binary splitting**.

Building a regression tree

1. Given a partition $R_1, R_2, \ldots, R_J$, what are the optimal values of c_j if we want to minimize $\sum_i (y_i - f(x_i))^2$?
2. How do we construct the regions $R_1, ..., R_J$?

1. The best c_j is just the average of y_i in region R_j:

$$\hat{c}_j = \text{average}(y_i | x_i \in R_j).$$

2. Finding the best binary partition in terms of minimum sum of squares is generally *computationally infeasible*. For this reason, we take a *top-down*, *greedy* approach that is known as **recursive binary splitting**.

Building a regression tree

1. Given a partition $R_1, R_2, \ldots, R_J$, what are the optimal values of c_j if we want to minimize $\sum_i (y_i - f(x_i))^2$?
2. How do we construct the regions $R_1, ..., R_J$?

1. The best c_j is just the average of y_i in region R_j:

$$\hat{c}_j = \text{average}(y_i | x_i \in R_j).$$

2. Finding the best binary partition in terms of minimum sum of squares is generally *computationally infeasible*. For this reason, we take a *top-down*, *greedy* approach that is known as **recursive binary splitting**.

Recursive binary splitting

1. **Top-down**: it begins at the top of the tree (all observations belong to a single region) and then successively splits the predictor space; each split is indicated via two new branches further down on the tree
2. **Greedy**: at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

Recursive binary splitting

1. Start with a single region R_1 (entire input space), and iterate:
    1. Select a region R_m, a predictor X_j , and a splitting point s, such that splitting R_m with the criterion $X_j < s$ produces the **largest decrease in RSS**
    2. Redefine the regions with this additional split.
2. Continues until stopping criterion, e.g. $N_m < 5$.

$$\mathrm{RSS}(T) = \sum_{m=1}^{|T|} N_m Q_m(T), \quad N_m = \#\{x_i \in R_m\},$$

where

$$Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$$

and $|T|$ is the number of terminal nodes in T.

What size of tree?

- The process described above may produce good predictions on the **training set**, but is likely to **overfit** the data (trees are very flexible).
- A smaller tree with fewer splits (that is, fewer regions) might lead to **lower variance** and better interpretation at the cost of a **little bias**.
- Tree size is a tuning parameter governing the **model's complexity**, and the optimal tree size should be adaptively chosen from the data
- One possible alternative is to produce splits only if the decrease in the RSS exceeds some **(high) threshold**. The problem is that a worthless split early on in the tree might be followed by a very good split.

Tree Pruning

Tree pruning grows a **very large tree** T_0, and then **prune** it back in order to obtain a subtree. The weakest link pruning procedure is:

1. Starting with with the initial full tree T_0, replace a subtree with a leaf node to obtain a new tree T_1. Select subtree to prune by minimizing

$$\frac{\mathrm{RSS}(T_1) - \mathrm{RSS}(T_0)}{|T_1| - |T_0|}$$

2. Iterate this pruning to obtain a sequence $T_0, T_1, T_2, \ldots, T_R$ where T_R is the tree with a single leaf node.

3. Select the optimal tree T_j by cross validation

Tree Pruning - equivalent procedure

The cost complexity criterion is given by

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|$$

where $\alpha \geq 0$ is a tuning parameter that governs the tradeoff between tree size and its goodness of fit to the data.

- Large/small values of α result in smaller/larger trees T_α
- The idea is to find, for each α, the subtree $T_\alpha \subseteq T_0$ to minimize $C_\alpha(T)$ (which is unique).
- Fact: the solution for each α is among $T_0, T_1, T_2, \ldots, T_R$ from weakest link pruning

Tree Pruning

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 $k = 1, \ldots, K$:

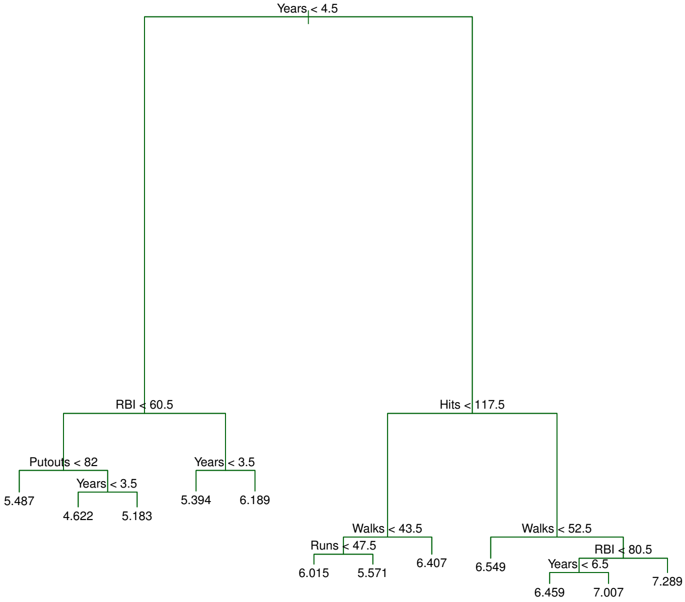
   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

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

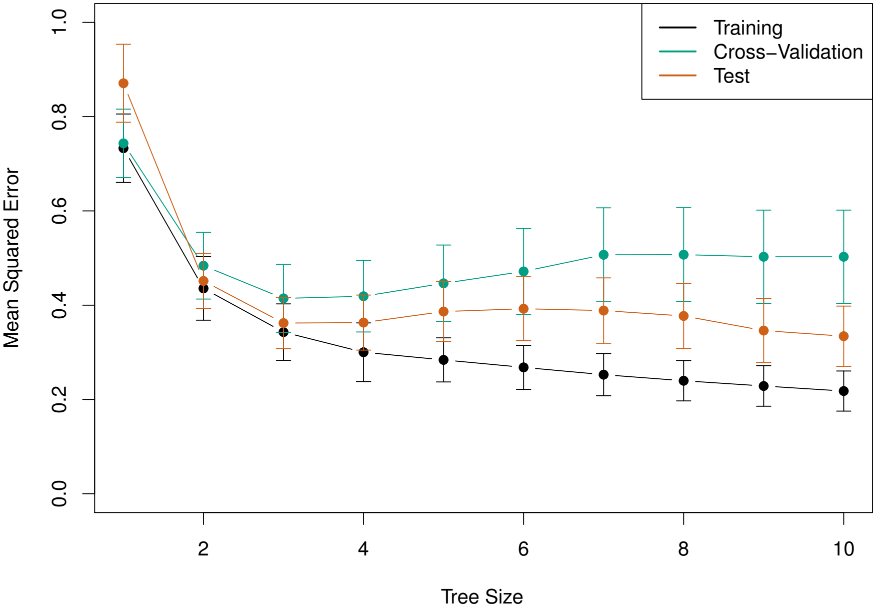
   Average the results for each value of α, and pick α to minimize the average error.

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

Example


Years < 4.5
RBI < 60.5
Hits < 117.5
Putouts < 82
Years < 3.5
Years < 3.5
5.487
4.622
5.183
5.394
6.189
Walks < 43.5
Walks < 52.5
Runs < 47.5
RBI < 80.5
Years < 6.5
6.015
5.571
6.407
6.549
6.459
7.007
7.289

Example


Training
Cross−Validation
Test
Mean Squared Error
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Tree Size

Classification trees

- A classification tree is used to predict a **qualitative response** rather than a quantitative one
- We predict that each observation belongs to the **most commonly occurring class** of training observations in the region to which it belongs
- Just as in the regression setting, we use recursive binary splitting to grow a classification tree
- However RSS cannot be used as a criterion for making the binary splits. A natural alternative to RSS is the classification error rate:

$$1 - \max_k \hat{p}_{mk},$$

where $\hat{p}_{mk}$ is the proportion of training observations in the mth region that are from the kth class.

Classification trees

- Classification error is not sufficiently sensitive for tree-growing. In practice two other measures are used
- The *Gini index* measures total variance across the K classes:

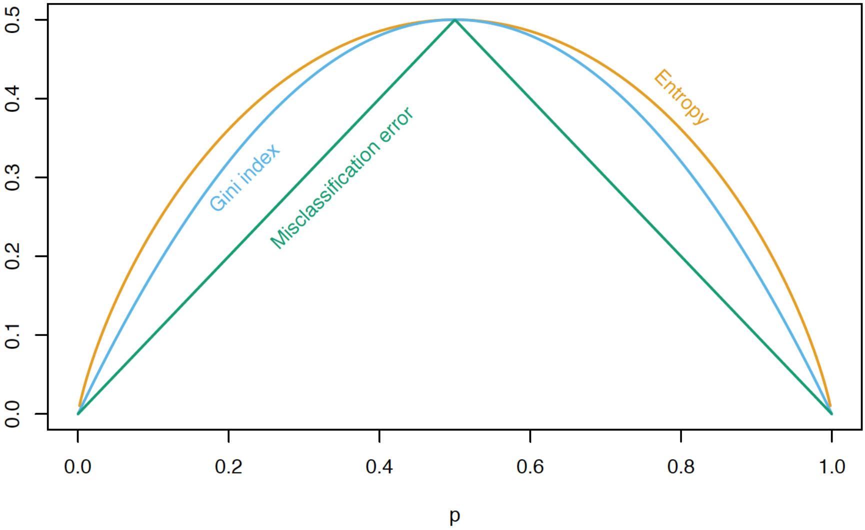
$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

- An alternative to the Gini index is *entropy*, given by

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log(\hat{p}_{mk})$$

- If all of the $\hat{p}_{mk}$'s are close to zero or one, both G and D are small. It is a measure of **node purity**, i.e. a small value indicates that a node contains predominantly observations from a single class.

Classification trees



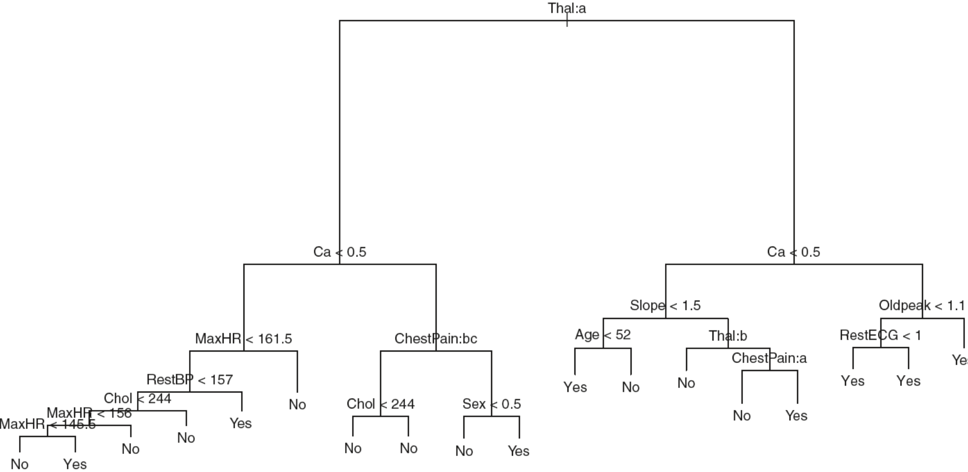
Classification trees

- When building a classification tree, either the Gini index or the entropy are typically used to evaluate the quality of a particular split
- Any of these three approaches might be used when pruning the tree, but the classification error rate is preferable if prediction accuracy of the final pruned tree is the goal

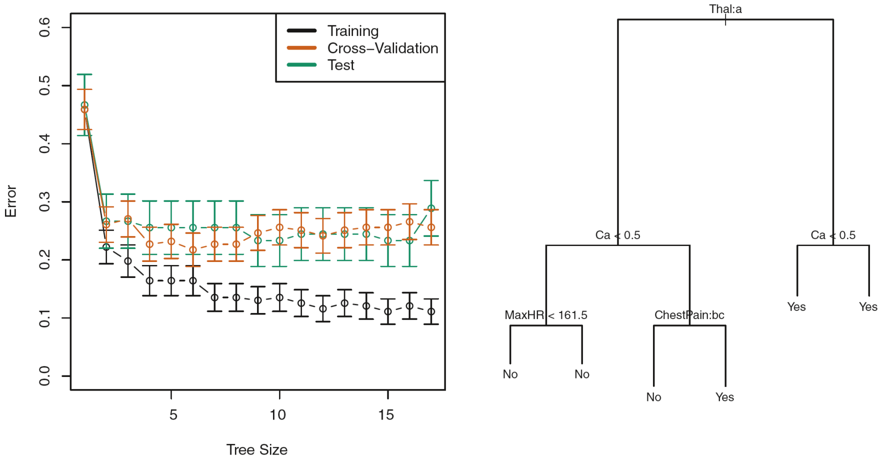
Example

Y: presence of heart disease (Yes/No)

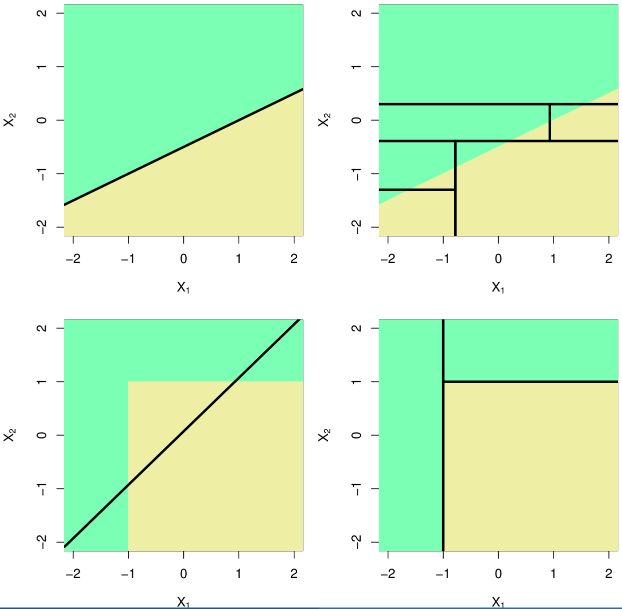
$\boldsymbol{X}$: heart and lung function measurements



Example


Training
Cross−Validation
Test
Error
0.0
0.1
0.2
0.3
0.4
0.5
0.6
5
10
15
Tree Size
Thal:a
Ca < 0.5
Ca < 0.5
MaxHR < 161.5
ChestPain:bc
Yes
Yes
No
No
No
Yes

Trees Versus Linear Models



Advantages and Disadvantages of Trees

- **(+)** Trees are flexible models
- **(+)** Trees can be easily interpreted
- **(+)** Trees can easily handle qualitative predictors without the need to create dummy variables + missing values
- **(-)** Trees are unstable and can be very non-robust: a small change in the data can cause a large change in the final estimated tree
- $\rightarrow$ The predictive performance of trees can be substantially improved by **aggregating many decision trees**, using methods like bagging, random forests, and boosting.