Tree-Based Methods

UNIVERSITY OF
SAN FRANCISCO

James D. Wilson

MATH 373

Plan for this Lecture

- Decison Trees
  - Regression trees
  - Classification trees
- Bagging
- Random Forests

Reference: ISL Sections 8.1; 8.2

Decision Trees

Given:

- Training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$
- Test data of the form $(\mathbf{x}_o, y_o)$

Aim:

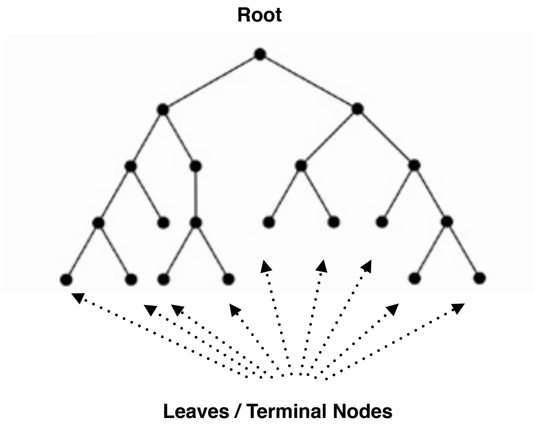
- Stratify or segment the predictor space $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ into a number of simple, non-overlapping regions $R_1, \ldots, R_J$ for some $J \leq n$
- Predict new observation $\hat{y}_o$ based on the *mean* or *mode* of the training observations in the region to which $\mathbf{x}_o$ belongs

Decision Trees

Properties:

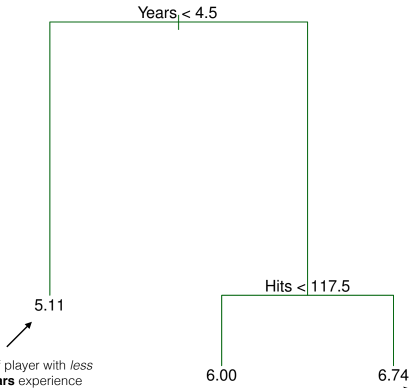
- Can be used for either discrete or continuous-valued y
  - Classification Trees: y is discrete-valued
  - Regression Trees: y is continuous-valued
- The set of splitting rules used to segment the predictor space can be represented by a tree

Trees



Regression Trees: Example

Aim: Segment **Years** and **Hits** to predict **Salary** of a player



log(**Salary**) of player with *less than* 4.5 **years** experience

log(**Salary**) of player with *more than* 4.5 **years** experience and *more than* 117.5 **hits**

Regression Trees: Example

From the previous decision tree, we identified three regions:

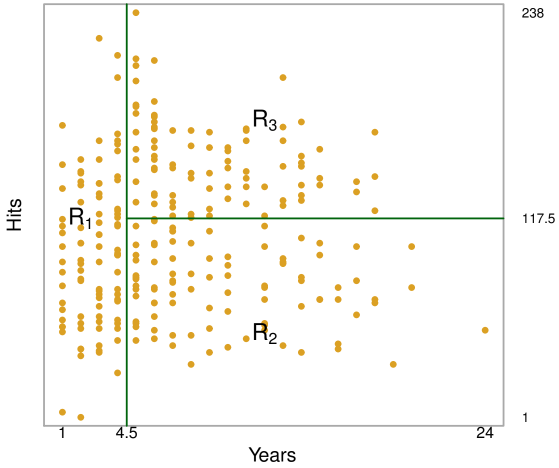
- $R_1 = \{X \mid \text{Years} < 4.5\}$
- $R_2 = \{X \mid \text{Years} \geq 4.5, \text{Hits} < 117.5\}$
- $R_3 = \{X \mid \text{Years} \geq 4.5, \text{Hits} \geq 117.5\}$

Prediction:

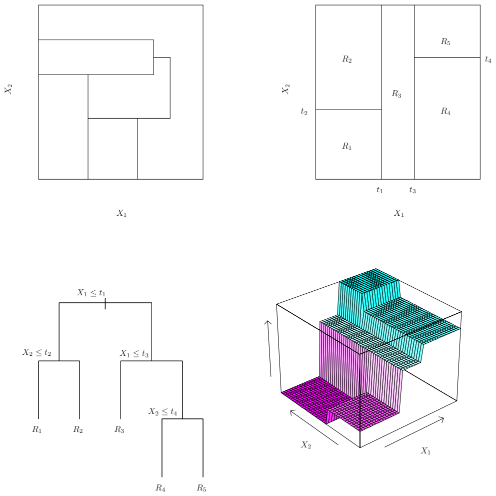
- Identify i such that $\mathbf{x}_o \in R_i$
- Predict $\hat{y}_o = \text{Ave}(y_j \mid \mathbf{x}_j \in R_i)$
- **Note**: we predict the *same* value for all new data that is contained in the same region!

Regression Trees: Example

Note: Our identified regions partition the predictor space.



Decision Trees in Higher Dimensions



Regression Trees

General Procedure:

- Divide the predictor space into J distinct and non-overlapping regions $R_1, \ldots, R_J$
- For every observation that falls into region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

Questions:

- How do we choose our regions?
- How do we choose J?

Choosing the Regions

Overall Aim: Identify the "boxes" $R_1, \dots, R_J$ that minimize the residual sum of squares:

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.

Unfortunate Point: It is computationally infeasible to consider every partition of J boxes / regions. Instead, we use a greedy and recursive approach called binary splitting

Binary Splitting

A greedy, top-down approach:

1. Select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X \mid X_j < s\}$ and $\{X \mid X_j \geq s\}$ leads to the greatest possible reduction in RSS. That is, for any j and s consider the half-planes:

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

   Then, we seek the value of j and s that minimize the equation:

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

2. Repeat the above procedure until a pre-specified stopping criterion is met.

How do we choose a stopping criterion?

- Binary splitting will perform well on the training data; however, it is prone to overfitting. How do we choose a simpler tree that performs well?
- We could require that each RSS reduction exceeds a high threshold; however, this doesn't work well in practice as often a "meaningless" cut is often followed by a much better split.
- Instead, we build a large decision tree T_o and then prune the tree to find the "best" subtree T via weakest link pruning.

Weakest Link Pruning

Rather than considering every possible subtree, we consider a sequence of trees indexed by a tuning parameter $\alpha \geq 0$.

For each α, there is a subtree $T \subseteq T_o$ that minimizes:

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

where $|T|$ = the number of terminal nodes in T.

Properties:

- α acts a penalty for the number of terminal nodes in a tree, namely α penalizes the complexity of a tree
- $\alpha = 0$: $T = T_o$

Building a Regression Tree

Algorithm

1. Use recursive binary splitting to grow a large tree T_o, stopping when each terminal node has fewer than some minimum number of observations
2. Apply weakest link pruning to T_o to obtain a sequence of best subtrees, as a function of α
3. Use k-fold cross-validation to choose α.
4. Return the subtree from Step 2 with the best α

Back to the Hitters Data

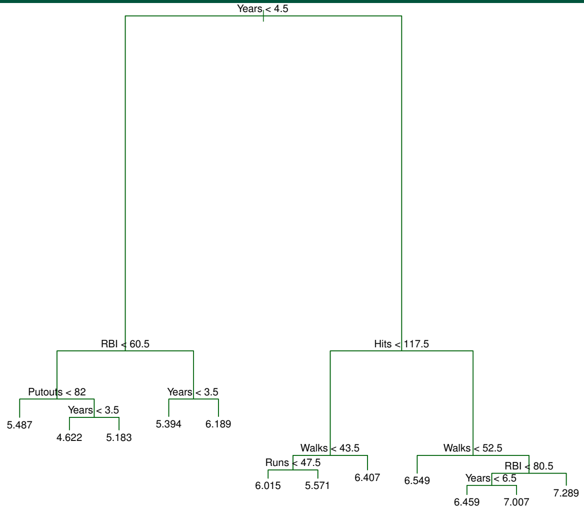


Figure: The unpruned tree that results from binary splitting.

Pruning The Hitters Data Tree

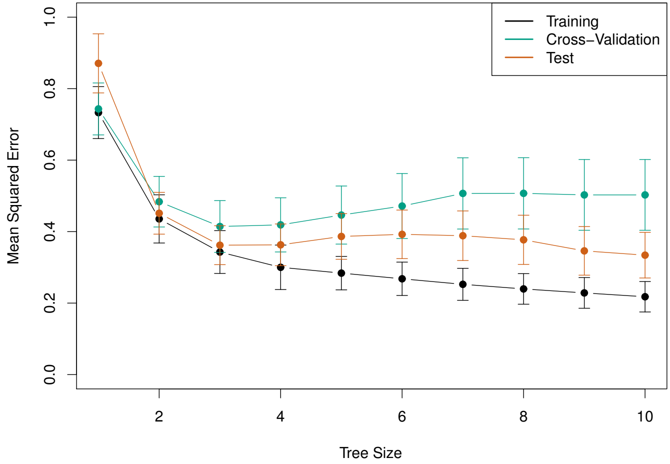


Figure: The best subtree has 3 terminal nodes.

Regression Trees vs. Standard Regression

Linear Regression: The model takes form

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

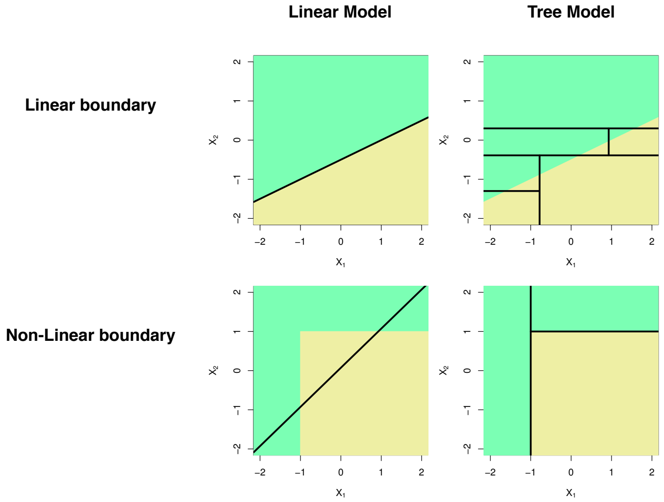
Regression Trees: The model takes form

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

where $c_j = \text{Ave}(y_i \mid \mathbf{x}_i \in R_j)$, and J = number of regions.

Which works better? As always, it depends!

Regression Trees vs. Standard Regression



Classification Trees

Setting:

- Training data: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ where $y_i \in \{1, \ldots, K\}$
- Test data: observations of form $(\mathbf{x}_o, y_o)$

Classification Trees: Classify $\mathbf{x}_o$ using

$$\phi(\mathbf{x}_o) = \sum_{j=1}^{J} c_j \mathbb{I}(X \in R_j)$$

where R_j is the jth region and

$$c_j = \text{Mode}(y_i \mid \mathbf{x}_i \in R_j)$$

Choosing R_j

We can again choose R_j using binary splitting; however, how do we measure the "mixing" of a region j?

Let $\hat{p}_{jk}$ = the proportion of training observations in the jth region that are from the kth class. There are **three** common measures to assess the mixture of a region:

- Classification error rate:

$$E = 1 - \max_k(\hat{p}_{jk}) \in [0, 1]$$

Choosing R_j

- Gini index:

$$G = \sum_{k=1}^{K} \hat{p}_{jk}(1 - \hat{p}_{jk}) = \text{total variance across the } m \text{ classes}$$

- Cross-entropy:

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

In each case, we want to choose regions R_j such that these measures are *minimized*.

Choosing R_j

Notes about purity metrics:

- When building a classification tree, one typically uses either the Gini index or the Cross-entropy, since they are more sensitive to *node purity*.
- When *pruning* a tree, the classification error rate is used whenever prediction is the goal.

Example: Heart Disease Classification

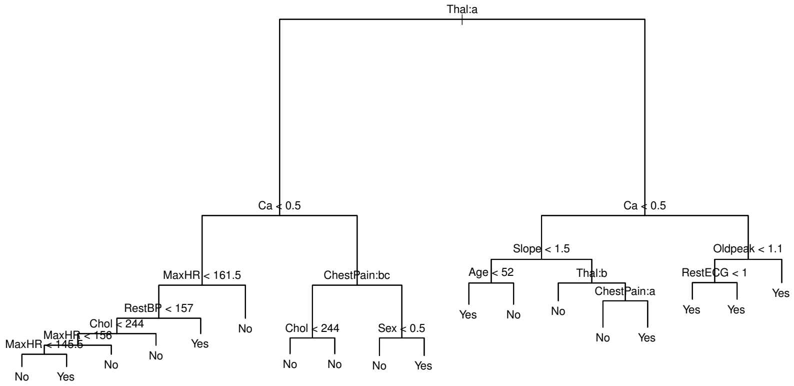
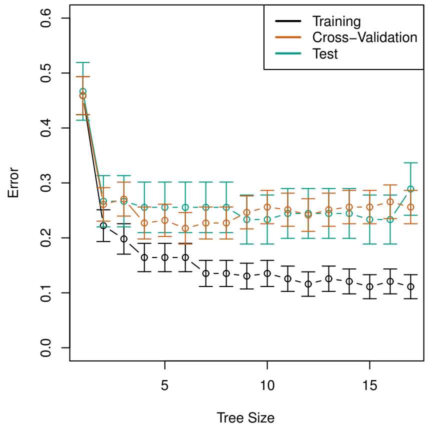


Figure: The unpruned classification tree. Note that the response is binary.

Example: Heart Disease Classification



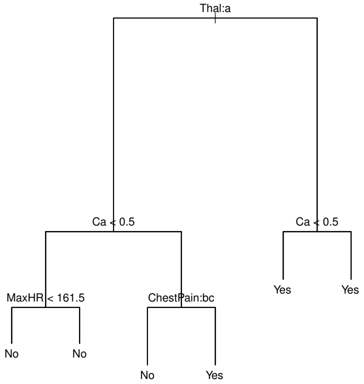


Figure: Classification errors and the pruned tree with minimal cross-validation error.

Pros and Cons of Decision Trees

- **Pro:** Very easy to explain
- **Pro:** Can be displayed graphically; easily interpreted
- **Pro:** Easily handle qualitative predictors without the need to create ”dummy” variables
- **Con:** Do not generally have the same level of predictive accuracy as regression and other classification methods since decision trees have high variance. But don’t worry, we’re not done with trees just yet...

Reducing Variance via Averaging

Problem: Decision trees suffer from high variance

- Let $X_1, \ldots, X_n$ be independent random variables where

$$\text{Var}(X_i) = \sigma^2$$

- Important (and straightforward) fact:

$$\text{Var}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{\sigma^2}{n}$$

Key Take-away: The average of *n* random variables has *n*-fold smaller variance than the variance of each random variable, taken separately. Can we use this to reduce the variance of trees?

Bagging

Bootstrap **agg**regat**ing**

Problem: Decision trees suffer from high variance

Idea:

- Split the training data into multiple data sets
- Create a decision tree for each set
- Average the results to reduce variance

Potential Issue: Eventually, we'd run out of data in the training set. Instead, we repeatedly sample (bootstrap) from the training data.

Bagging

Algorithm

- **Given:** Training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, and test observation $\mathbf{x}_o$
- Bootstrap B separate training sets (with replacement)
- Create a decision tree for each of the B samples $T_1, \ldots, T_B$ and train a model for $i = 1, \ldots, B$ on each tree
  - Regression: $f_i(x)$
  - Classification $\phi_i(x)$
- Aggregate the models:
  - Regression: $f_{bag}(x) = \frac{1}{B}\sum_{i=1}^{B} f_i(x)$
  - Classification: $\phi_{bag}(x) = \text{Mode}(\phi_1(x), \ldots, \phi_B(x))$

Bagging

- When bagging, one typically grows "deep" trees for each sample $i = 1, \ldots, B$ to ensure low bias. Of course, these trees will then have high variance, but averaging reduces this large variance
- Instead, one chooses an upper limit on the size of each tree *a priori*
- The resulting trees $T_1, \ldots, T_B$ are likely highly correlated due to sampling with replacement from original training data. As a result, we cannot reduce the variance as much as we'd like to.

Question: Can we somehow reduce the correlation between these tree samples?

Random Forests

Idea: Remove correlation issues that result from Bagging

Means: For each bootstrapped sample $i = 1, \ldots, B$, only allow a random selection of $m < p$ predictors for building tree T_i

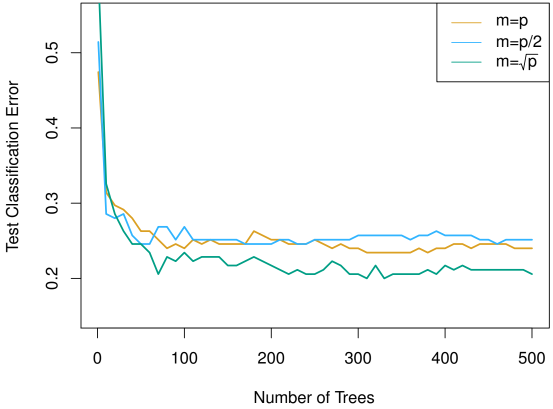
Result: Redundancies caused by ”major players” in predictors are lost, and the trees $T_1, \ldots, T_B$ become ”more independent.”

Random Forests

Algorithm

- **Given:** Training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, and test observation $\mathbf{x}_o$
- Bootstrap B separate training sets (with replacement)
- For $i = 1, \ldots, B$, choose $m < p$ predictors at random. Create a decision tree for each of the B samples $T_1, \ldots, T_B$ and train a model for $i = 1, \ldots, B$ on each tree
- Aggregate the models:
    - Regression: $f_{bag}(x) = \frac{1}{B}\sum_{i=1}^{B} f_i(x)$
    - Classification: $\phi_{bag}(x) = \text{Mode}(\phi_1(x), \ldots, \phi_B(x))$

Choice of m



In practice, $m = \sqrt{p}$ is often used.

Summary of Bagging and Random Forests

- Low bias: trees are typically grown ”deep”
- Low variability: averaging many samples decreases variance
- Random forests strongly reduce correlation among trees and allow for even further decreases in variance
- Both Bagging and Random Forests increase predictive power at the expense of interpretability: decision trees are easy to interpret, but combinations are difficult!

Next Up

- Implementation of tree-based methods in R
- Unsupervised Learning
  - Hierarchical clustering
  - k-means
  - Spectral clustering