Overview

- Intuition for Trees
- Regression Trees
- Classification Trees
Idea of Trees: Regression Trees
Continuous response

Binary Tree

\[
Y = \begin{cases} 
1.2 & X \leq 1 \\
1.4 & X > 1 \\
0.7 & X > 0.3 \\
1.9 & X \leq 0.3 \\
1.3 & X > 2 \\
0.2 & X \leq 2 \\
0.3 & X > 2 
\end{cases}
\]
Idea of Trees: Classification Tree
Discrete response

Survived in Titanic?

800/200

No

Sex=F

150/50

No

Age <35

3/17

Yes

Age ≥35

147/33

No

Sex=M

650/150

No

Age <27

70/130

Yes

Age ≥27

580/20

No

Missclassification rate:
- Total: (3+33+70+20) / 1000 = 0.126
- “Yes”-class: 53/200 = 0.26
- “No”-class: 73/800 = 0.09
Intuition of Trees: Recursive Partitioning

For simplicity: Restrict to recursive binary splits
Fighting overfitting: Cost-complexity pruning

Overfitting: Fitting the training data **perfectly** might not be good for predicting future data

For trees:
1. Fit a very detailed model
2. Prune it using a complexity penalty to optimize cross-validation performance
Building Regression Trees 1/2

- Assume given partition of space $R_1, \ldots, R_M$

Tree model:  
$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

- Goal is to minimize sum of squared residuals:  
$$\sum (y_i - f(x_i))^2$$

- Solution: Average of data points in every region  
$$\hat{c}_m = \text{ave}(y_i | x_i \in R_m)$$
Finding the best binary partition is computationally infeasible

Use greedy approach: For variable $j$ and split point $s$ define the two generated regions:

$$R_1(j, s) = \{X | X_j \leq s\} \quad \text{and} \quad R_2(j, s) = \{X | X_j > s\}.$$ 

Choose splitting variable $j$ and split point $s$ that solve:

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

inner minimization is solved by

$$\hat{c}_1 = \text{ave}(y_i | x_i \in R_1(j, s)) \quad \text{and} \quad \hat{c}_2 = \text{ave}(y_i | x_i \in R_2(j, s))$$

Repeat splitting process on each of the two resulting regions
Pruning Regression Trees

- Stop splitting when some minimal node size (= nmb. of samples per node) is reached (e.g. 5)
- Then, cut back the tree again (“pruning”) to optimize the cost-complexity criterion:

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

Goodness of fit

Complexity

- Tuning parameter \( \alpha \) is chosen by cross-validation
Classification Trees

- Regression Tree:
  Quality of split measured by “Squared error”

- Classification Tree:
  Quality of split measured by general “Impurity measure”
Classification Trees: Impurity Measures

- Proportion of class \( k \) observations in node \( m \):
  \[
  \hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)
  \]

- Define majority class in node \( m \): \( k(m) \)

- Common impurity measures \( Q_m(T) \):
  
  **Misclassification error:**
  \[
  \frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}
  \]

  **Gini index:**
  \[
  \sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})
  \]

  **Cross-entropy or deviance:**
  \[
  - \sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.
  \]

- For just two classes:
Example: Gini Index

Side effects after treatment? 100 persons, 50 with and 50 without side effects: 50 / 50 (No / Yes)

Split on sex

50 / 50

M

30 / 40
Gini = 0.49

F

20 / 10
Gini = 0.44

Total Gini = 0.49 + 0.44 = 0.93

Split on age

50 / 50

old

10 / 50
Gini = 0.27

young

40 / 0
Gini = 0

Total Gini = 0.27 + 0 = 0.27

0.27 < 0.93, therefore: Choose split on age
Classification Trees: Impurity Measures

- Usually:
  - Gini Index used for building
  - Misclassification error used for pruning
Example: Pruning using Misclass. Error (MCE)

\[ C_\alpha(T) = \sum_{m=1}^{\mid T \mid} N_m Q_m(T) + \alpha \mid T \mid \]

\[
\begin{align*}
C_\alpha(T) &= 50 \times 0 + 10 \times 0 + 40 \times 0 + 0.5 \times 3 = 1.5 \\
C_\alpha(T) &= 60 \times 0.167 + 40 \times 0 + 0.5 \times 2 = 11.0
\end{align*}
\]

Smaller \( C_\alpha(T) \), therefore don’t prune
Trees in R

- Function “rpart” (recursive partitioning) in package “rpart” together with “print”, “plot”, “text”

- **Function “rpart” automatically prunes using optimal \( \alpha \) based on 10-fold CV**

- Functions “plotcp” and “printcp” for cost-complexity information

- Function “prune” for manual pruning
Concepts to know

- Trees as recursive partitionings
- Concept of cost-complexity pruning
- Impurity measures
R functions to know