



# Statistical Learning

## Lecture 6: Tree-based Methods

---

Yi He

January 25, 2023

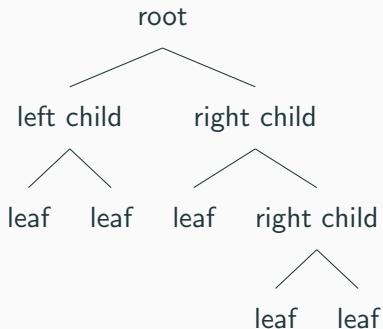
# Plan for Today

1. Introduction
2. Regression/Classification Tree
3. CART Algorithm
4. Bagging and Random Forest

# Introduction

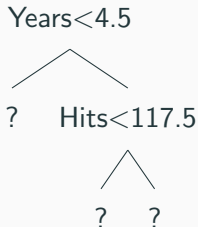
---

# Binary Tree



- root = top node
- terminal node (leaf): node with no child
- interior node: have left and right children
- depth of a node = length of path from the node to the root
- height of a tree = maximum depth
- Stump if height=1

# Decision Tree: Example



ISLR Figure 8.1: Hitter data for male baseball players, years = the number of years that he has played in the major leagues, Hits = number of hits that he made in the previous year.

- root/interior node = condition for a single feature
- TRUE: go left
- FALSE: go right
- Partition the data into regions at the terminal nodes
- Forecast/Estimate at the terminal nodes

# Regression/Classification Tree

---

# Partitioning the Feature Space

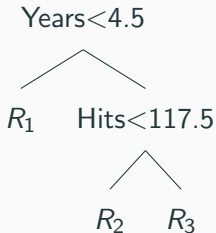
Tree-based model for regression function

$$f(x) = \sum_{m=1}^M c_m \mathbb{1}[x \in R_m] = \begin{cases} c_1 & x \in R_1, \\ \vdots & \\ c_M & x \in R_M \end{cases}$$

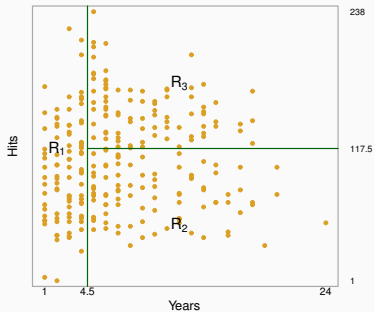
For every observation that falls into the region leaf  $R_m$ , our prediction is constant  $c_m$ .

- Looks like local regression, but the ‘neighbors’  $R_m$  do not depend on  $x$ .
- Looks like step regression, by using regions  $R_m$  instead of intervals for multivariate features
- ... but we will fit regions  $R_m$  to the data: next part

# Partitioning the Feature Space: Illustration



ISLR Figure 8.2



More exercises in Tute.



# Fitting a Tree Model

- Target  $y_i$  and features  $x_i \in \mathcal{X}$ ,  $i = 1, \dots, n$ .
- Suppose that the partition  $\{R_m : m = 1, \dots, M\}$  are *given*:

$$R_m \cap R_l = \emptyset, \quad m \neq l, \quad \bigcup_{m=1}^M R_m = \mathcal{X}.$$

Later on we will estimate them as well. For now we take them as known.

- We fit the parameters  $\{c_m\} \equiv \{c_m : m = 1, \dots, M\}$ . The mean square error function

$$\{\hat{c}_m\} = \underset{\{c_m\}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n \left( y_i - \sum_{m=1}^M c_m \mathbb{1}[x \in R_m] \right)^2.$$

Decompose the sum of squared errors

$$\begin{aligned} & \sum_{i=1}^n \left( y_i - \sum_{m=1}^M c_m \mathbb{1}[x \in R_m] \right)^2 \\ &= \sum_{m=1}^M \left\{ \sum_{i: x_i \in R_m} (y_i - c_m)^2 \right\} \equiv \sum_{m=1}^M L_m(c_m) \end{aligned}$$

For each  $m$ , minimizing the univariate error function

$$L_m(c) = \sum_{i: x_i \in R_m} (y_i - c)^2,$$

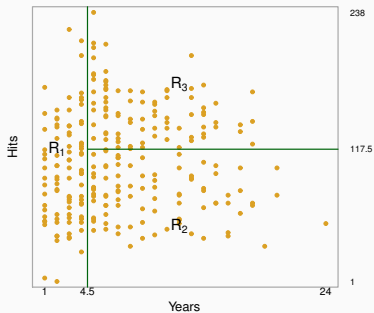
yields that (see Tute1, Q2a):

$$\hat{c}_m = \frac{1}{|\{i : x_i \in R_m\}|} \sum_{i: x_i \in R_m} y_i,$$

the *sample mean of the response values* for the training observations in the terminal node  $m$ . Here  $|\mathcal{I}|$  means the number of elements in index set  $\mathcal{I}$ .



ISLR Figure 8.2



# Regression Versus Classification Trees

- For classification problem with binary target  $Y \in \{0, 1\}$  and features  $X \in \mathcal{X}$
- Recall from Lecture 4: the regression function is the posterior probability

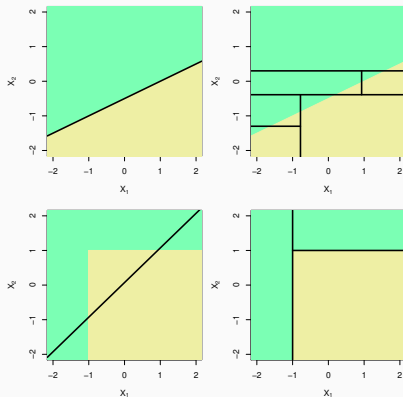
$$\mu(x) = \mathbb{E}[Y|X = x] = \mathbb{P}(Y = 1|X = x).$$

and with  $\mu(x)$  we can construct the Bayes classifier.

- We can use the similar prediction method for regression and classification problems when the partition  $\{R_m\}$  is given.
- However, we shall use *different* methods to construct the subregions  $R_m$  for regression and classification problems: next part.

# Tree versus Linear models

- Linear model  $h(x) = \beta_0 + x^T \beta$
- Tree model  $h(x) = \sum_{m=1}^M \tilde{c}_m \mathbb{1}(x \in R_m)$



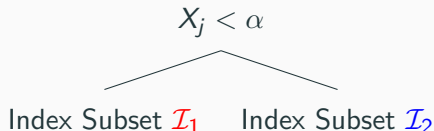
ISLR Figure 8.7: Two-class Classification Problem

# CART Algorithm

---

# Splitting the Data Set

For each candidate feature  $X_j$  and splitting threshold  $\alpha$



- Denote  $\mathcal{I} \subset \{1, \dots, n\}$  as the index set *before* the split.
- $\mathcal{I}_1 = \{i \in \mathcal{I} : x_{ij} < \alpha\}$ ,  $\mathcal{I}_2 = \{i \in \mathcal{I} : x_{ij} \geq \alpha\}$
- In sample, we often choose  $\alpha$  such that  $\{i \in \mathcal{I} : x_{ij} = \alpha\} = \emptyset$  to avoid boundary issues and thus  $\mathcal{I}_2 = \{i \in \mathcal{I} : x_{ij} > \alpha\}$ .
- Accordingly we split the target values  $\mathcal{Y} = \{y_i : i \in \mathcal{I}\}$  into two subsets  $\mathcal{Y}_1 = \{y_i : i \in \mathcal{I}_1\}$  and  $\mathcal{Y}_2 = \{y_i : i \in \mathcal{I}_2\}$ .

# Impurity Function

- We wish the target values to be 'pure' *within* each subset of target values  $\mathcal{Y}_1 = \{y_i : i \in \mathcal{I}_1\}$  and  $\mathcal{Y}_2 = \{y_i : i \in \mathcal{I}_2\}$ .
- We find the best split that minimizes the *impurity* criterion

$$\frac{|\mathcal{I}_1|}{|\mathcal{I}|} \cdot \ell(\mathcal{Y}_1) + \frac{|\mathcal{I}_2|}{|\mathcal{I}|} \cdot \ell(\mathcal{Y}_2)$$

where  $\ell(\mathcal{Y}_m)$  measures the *impurity* of each subset  $\mathcal{Y}_m$ .

- For regression trees, we measure impurity through sample variance:

$$\ell(\mathcal{Y}_m) = \frac{1}{|\mathcal{I}_m|} \sum_{i \in \mathcal{I}_m} (y_i - \bar{y}_m)^2, \quad \bar{y}_m = \frac{1}{|\mathcal{I}_m|} \sum_{i \in \mathcal{I}_m} y_i$$

- For two-class classification problem:

$$\ell(\mathcal{Y}_m) = \psi(p_{m1}, p_{m0}), \quad p_{mk} = \frac{1}{|\mathcal{I}_m|} \sum_{i \in \mathcal{I}_m} \mathbb{1}[y_i = k], \quad k = 0, 1,$$

where  $\psi$  is an impurity function to be specified.



- The *impurity* criterion

$$\begin{aligned} & \frac{|\mathcal{I}_1|}{|\mathcal{I}|} \cdot \ell(\mathcal{Y}_1) + \frac{|\mathcal{I}_2|}{|\mathcal{I}|} \cdot \ell(\mathcal{Y}_2) \\ &= \frac{|\mathcal{I}_1|}{|\mathcal{I}|} \cdot \frac{1}{|\mathcal{I}_1|} \sum_{i \in \mathcal{I}_1} (y_i - \bar{y}_1)^2 + \frac{|\mathcal{I}_2|}{|\mathcal{I}|} \frac{1}{|\mathcal{I}_2|} \sum_{i \in \mathcal{I}_2} (y_i - \bar{y}_2)^2 \end{aligned}$$

- It is equivalent to minimize the sum of squared errors

$$\sum_{i \in \mathcal{I}_1} (y_i - \bar{y}_1)^2 + \sum_{i \in \mathcal{I}_2} (y_i - \bar{y}_2)^2,$$

where  $\bar{y}_1$  and  $\bar{y}_2$  are the subset average target values (i.e. the least-squares estimator for this candidate split).

- Choose the feature  $X_j$  and threshold  $\alpha$  that minimizes sum of squared errors.

The impurity function

$$\ell(\mathcal{Y}_m) = \psi(p_{m1}, p_{m0}), \quad p_{mk} = \frac{1}{|\mathcal{I}_m|} \sum_{i \in \mathcal{I}_m} \mathbb{1}[y_i = k], \quad k = 0, 1,$$

needs to satisfy the following properties

- $\psi\left(\frac{1}{2}, \frac{1}{2}\right) \geq \psi(p, 1 - p)$ : lower is better
- $\psi(0, 1) = \psi(1, 0) = 0$ : the ideal separation
- $\psi(p, 1 - p)$  increases in  $p$  on  $[0, 1/2]$  and decreases in  $[1/2, 1]$

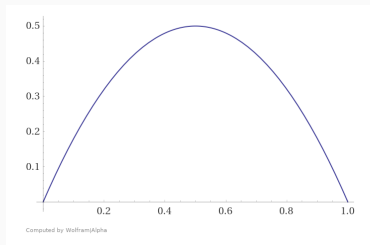
# Impurity Functions

- Gini function

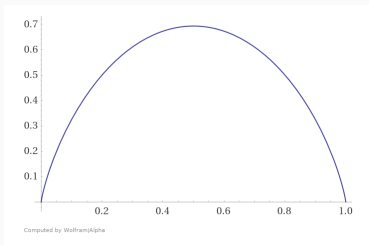
$$\psi(p, 1 - p) = p(1 - p) + (1 - p)p = 2p(1 - p),$$

- Entropy

$$\psi(p, 1 - p) = -p \log p - (1 - p) \log(1 - p).$$



(a) Gini  $\psi(p, 1 - p)$



(b) Entropy  $\psi(p, 1 - p)$

# Growing a Tree

- Recursive binary splitting: repeat the splitting processes at each node until *a stopping criterion* is reached
- for instance, may continue until no region contains more than 5 observations.
- or until the a maximum depth value is reached,...
- Without a stopping criterion, CART algorithm tends to isolate all observations!

The stopping rule is crucial for bias variance tradeoff.

- Stopping early:  $\downarrow$  variance,  $\uparrow$  bias
- Stopping late:  $\uparrow$  variance,  $\downarrow$  bias

# Bagging and Random Forest

---

- Tree is unstable: a small change in the data can cause a large change in the final estimate.
- Unstable estimator suffers in large predictive variances
- Aggregate many decision trees to **smooth** the estimated regression function and reduce estimation variance
- Regression: average the target estimates over tree
- Classification: average the posterior probabilities estimates over trees
- But... how to generate many trees, from one data set?

# Bagging

- Bagging = Bootstrap aggregating
- Resample  $n^*$  data points randomly *with replacement* from the training database

$$S = \{y_i, x_i : i = 1, \dots, n\}$$

- Repeat  $B$  times to extract the *bootstrapped* training sets

$$S^{*b} = \{y_i^{*b}, x_i^{*b} : i = 1, \dots, n^*\}, \quad b = 1, \dots, B$$

- Aggregate the estimates

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(x)$$

where  $\hat{f}^{*b}(x)$  is the estimate using the *bootstrapped* training set  $S^{*b}$

- Bagging uses  $n^* = n$ .

- Subagging = subsample aggregating
- Subsample aggregating: resample  $n^* < n$  times *without replacement* for each bootstrap data set
- Subagging could have substantial computational advantages since the original predictor is only evaluated many times for  $n^*$  instead of  $n$  data points
- Fraction subagging uses  $n^* = \lceil n\alpha \rceil$ ,  $\alpha \in (0, 1)$
- Small order subagging:  $n^*/n \approx 0$  but sufficiently large  $n^*$ .
- For small order subagging, re-sampling with replacement often works similarly as that without replacement



# Real-life Example: Ozone Data Set

- $n = 330$  maximum daily ozone in the Los Angeles area
- $p = 8$  meteorological predictor variables
- $B = 25$  trees

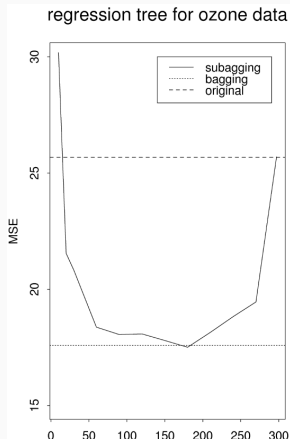


Figure 10 in Bühlmann and Yu (2002, AoS): MSE over 50 training-test-set random divisions

# Correlations Between Trees

- Bootstrapping does not generate new data!
- Aggregation smooths the estimated regression function and improves stability, but does *not* contain more information beyond your original training set.
- I **won't** accept the following explanations from the textbook in exam as the trees are **not** independent:

Recall that given a set of  $n$  independent observations  $Z_1, \dots, Z_n$ , each with variance  $\sigma^2$ , the variance of the mean  $\bar{Z}$  of the observations is given by  $\sigma^2/n$ . In other words, *averaging a set of observations reduces variance*.

- Since the bootstrapped training sets  $S^{*b}$  all depend on the same random sample  $S$ , the tree estimators  $\hat{f}^{*b}(x) = \hat{f}^{*b}(x; S)$  are correlated.
- If the correlation is too high, law of large numbers is not working well and aggregation makes little improvements.

# Random Forest

- A *random* sample of  $m$  predictors is chosen from the full set of  $p$  predictors for each split
- ... to *decorrelate* the tree estimators  $\hat{f}^{*b}$
- In bagging we use all predictors with  $m = p$
- For classification you may take  $m \approx \sqrt{p}$ :

