

Machine Learning for Economists and Business Analysts

Trees and Forests

Anthony Strittmatter

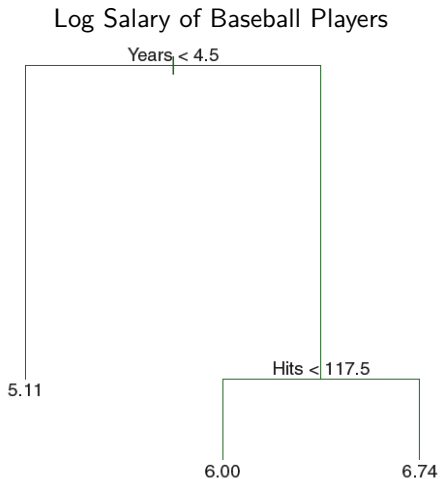
Literature

- ▶ James, Witten, Hastie, and Tibshirani (2013): "An Introduction to Statistical Learning", Springer, [download](#),
Chapters 4.1-4.2 and 8.
- ▶ Hastie, Tibshirani, and Friedman (2009): "Elements of Statistical Learning", 2nd ed., Springer, [download](#),
Chapter 9.2.

Tree

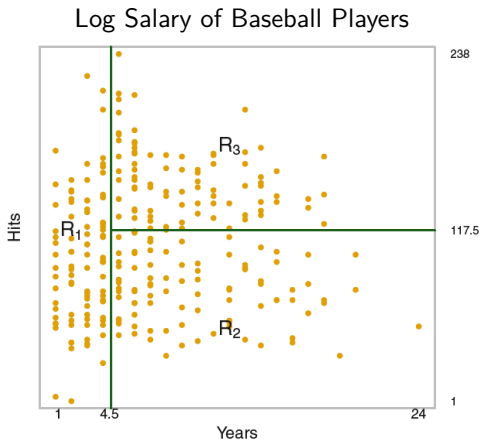
- ▶ Trees partition the sample into mutually exclusive groups l_j , which are called leaves.
- ▶ Let $\pi = \{l_1, \dots, l_{\#(\pi)}\}$ be the terminal leaves of a specific tree or sample partition.
- ▶ Let $l_j \equiv l_j(x, \pi)$ be the respective leaf (for $j = 1, \dots, \#(\pi)$).
- ▶ The leaf $l_j(x, \pi)$ of tree π is a function of the covariates X such that $x \in l_j$.
- ▶ Let $\#(\pi)$ be the number of terminal leaves in tree π .

Example: Shallow Tree



Source: James, Witten, Hastie, Tibshirani (2013)

Example: Shallow Tree (cont.)



Source: James, Witten, Hastie, Tibshirani (2013)

Recursive Partitioning

- ▶ Trees select the leaves with a top-down, greedy algorithm, which is called *recursive partitioning*.
- ▶ *Top-down* because we start with a root (tree without leaves) and successively add splits.
- ▶ *Greedy* because at each step of the tree building we add the split that improves the prediction power best (instead of looking ahead).

Tree Building Algorithm

- (1) Start with the entire sample (root).
- (2) For each predictor X_j and cut-point s define the pair of half-planes

$$I_1^{(j,s)} = \{X | X_j < s\} \text{ and } I_2^{(j,s)} = \{X | X_j \geq s\}.$$

- ▶ Calculate the mean outcomes \bar{Y}_1 and \bar{Y}_2 in each half-plane, respectively.
- ▶ Seek the covariate X_{j1}^* and the cut-point s_1^* that minimise

$$\sum_{i: X_i \in I_1^{(j,s)}} (Y_i - \bar{Y}_1)^2 + \sum_{i: X_i \in I_2^{(j,s)}} (Y_i - \bar{Y}_2)^2.$$

Tree Building Algorithm (cont.)

(2) For each predictor X_j and cut-point s define the triple of half-planes

$$l_1^{(j,s)} = \{X | X_{j1}^* < s_1^*, X_j < s\}, l_2^{(j,s)} = \{X | X_{j1}^* < s_1^*, X_j \geq s\}, \text{ and} \\ l_3^{(j,s)} = \{X | X_{j1}^* \geq s_1^*\}$$

and

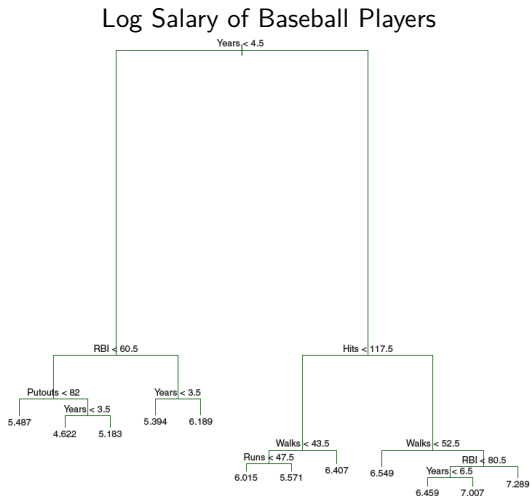
$$l_1^{(j,s)} = \{X | X_{j1}^* \geq s_1^*, X_j < s\}, l_2^{(j,s)} = \{X | X_{j1}^* \geq s_1^*, X_j \geq s\}, \text{ and} \\ l_3^{(j,s)} = \{X | X_{j1}^* < s_1^*\}.$$

- Calculate the mean outcomes \bar{Y}_1 , \bar{Y}_2 , and \bar{Y}_3 in each half-plane, respectively.
- Seek the covariate X_{j2}^* and the cut-point s_2^* that minimise

$$\sum_{i: X_i \in l_1^{(j,s)}} (Y_i - \bar{Y}_1)^2 + \sum_{i: X_i \in l_2^{(j,s)}} (Y_i - \bar{Y}_2)^2 + \sum_{i: X_i \in l_3^{(j,s)}} (Y_i - \bar{Y}_3)^2.$$

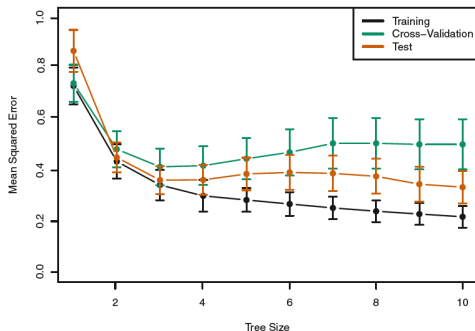
(3) Continue until some stopping rule is reached (e.g., max. tree size, min. terminal leave size, min. MSE gain) .

"Deep" Tree



Source: James, Witten, Hastie, Tibshirani (2013)

Selecting Optimal Tree Size



- Pruning the complexity of trees can improve out-of-sample prediction power.
- Select optimal tree size π with cross-validation.

Source: James, Witten, Hastie, Tibshirani (2013)

Complexity Pruning

- (A) Use recursive partitioning to grow the deep tree π_0 in the training data.
- (B) For each value of α a subtree $\pi \subseteq \pi_0$ minimises

$$\sum_{j=1}^{\#(\pi)} \sum_{i: X_i \in I_j} (Y_i - \bar{Y}_j)^2 + \alpha \#(\pi). \quad (1)$$

Obtain a sequence of best subtrees.

- (C) Use cross-validation to choose α . Partition the sample in k folds. For each fold:
 - (a) Repeat steps (A) and (B) excluding the k th-fold.
 - (b) Evaluate the MSE using equation (1) in the k th-fold.
 - (c) Average the MSE across the k folds for each value of α and select the α that minimises the average MSE.
- (D) Return to the subtree from (B) with the selected value of α .

Prediction

- For the selected tree π^* use the estimation sample to predict

$$\hat{Y}_i = \frac{1}{\sum_{j=1}^{\#(\pi^*)} \sum_{i=1}^N 1\{X_i \in I_j(x, \pi^*)\}} \sum_{j=1}^{\#(\pi^*)} \sum_{i=1}^N 1\{X_i \in I_j(x, \pi^*)\} \cdot Y_i.$$

- This is equivalent to the linear regression

$$\min_{\beta} \sum_{i=1}^N \left(Y_i - \sum_{j=1}^{\#(\pi^*)} 1\{X_i \in I_j(x, \pi^*)\} \beta_j \right)^2.$$

Advantages and Disadvantages of Trees

Advantages:

- ▶ Shallow trees are very easy to understand.
- ▶ Shallow trees can be displayed graphically in a nice way.
- ▶ Trees automatically handle interactions between covariates.
- ▶ It is not necessary to transform covariates as long as they have an order.

Disadvantages:

- ▶ Often trees are unstable and have a high variance.

Bootstrap Sampling

- ▶ We observe a sample $\{Y_i, X_i\}_{i=1}^N$ with size N
- ▶ Bootstrap algorithm:
 1. Draw randomly N observations with replacement from the original sample
 2. Estimate \hat{Y}_i^b in the “bootstrapped” sample b with a tree
 3. Repeat 1. and 2. B times
- ▶ We obtain B estimates $\hat{Y}_i^1, \hat{Y}_i^2, \dots, \hat{Y}_i^B$

Subsampling

- ▶ We observe a sample $\{Y_i, X_i\}_{i=1}^N$ with size N
- ▶ Subsampling algorithm:
 1. Draw randomly $M < N$ observations without replacement from the original sample
 2. Estimate \hat{Y}_i^s in the subsample s with a tree
 3. Repeat 1. and 2. S times
- ▶ We obtain S estimates $\hat{Y}_i^1, \hat{Y}_i^2, \dots, \hat{Y}_i^S$

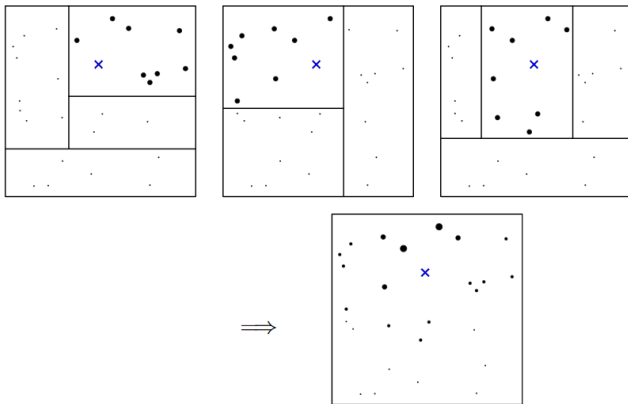
Random Forests

- ▶ Build G deep trees π_g on different subsets of the data (subsampling or bootstrapping) and covariates.
 - decorrelated trees
- ▶ These trees are overfitted in the sample and will have a high out-of-sample variance.
- ▶ To overcome this problem, we aggregate the trees

$$\hat{Y}_i^{RF} = \frac{1}{G} \sum_{g=1}^G \hat{Y}_i^{\pi_g}.$$

- ▶ This procedure is often called bootstrap-aggregation ("bagging").
- ▶ We lose interpretability but gain prediction power compared to (shallow) trees.
- ▶ Tuning parameters: Forest size, subsample selection, covariate selection, tree size, honest inference, etc.

Random Forest: Weighted Representation



Source: [Athey, Tibshirani, Wager \(2018\)](#)

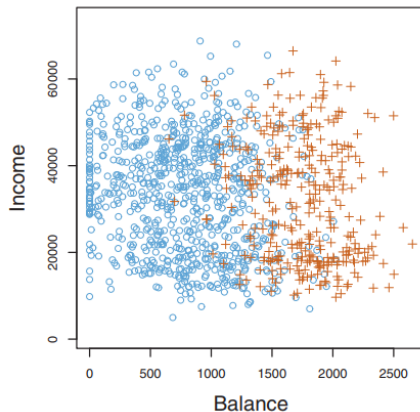
Classification

Outcome variable is categorical:

- ▶ School degrees (ordered)
- ▶ Occupations (unordered)
- ▶ Credit default (binary dummy)

→ In this lecture we consider binary dummies as outcome variable.

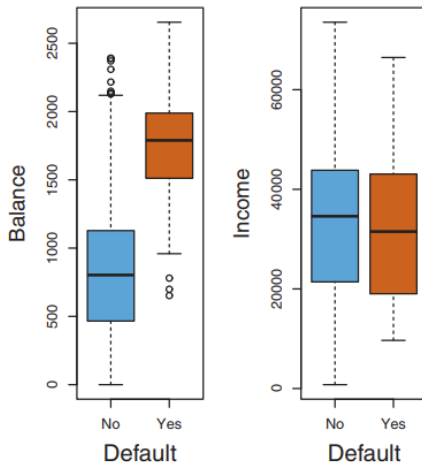
Example: Credit Default Risk



→ defaulted (orange crosses) and not defaulted (blue circles)

Source: James, Witten, Hastie, Tibshirani (2013)

Example: Credit Default Risk



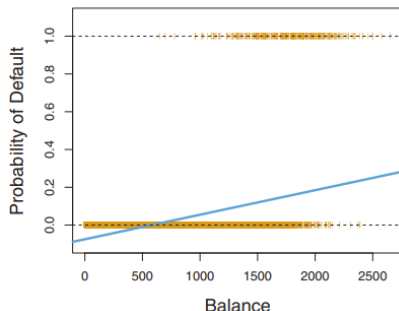
→ defaulted (orange) and not defaulted (blue)

Source: James, Witten, Hastie, Tibshirani (2013)

Why not linear Regression?

Possible disadvantages of linear regression when outcome is binary:

- Predictions off-support:



- Linearity assumption violated by construction.
- Heteroskedasticity → there exist more efficient estimators.

Why are Trees Suited for Binary Outcomes?

Advantages:

- ▶ Trees allow for non-linearities.
- ▶ Group averages cannot lie outside of support $[0, 1]$.

Disadvantages:

- ▶ MSE is not optimal to select optimal complexity of “classification” trees.

| Y | Estimator 1 | | Estimator 2 | |
|-----|-----------------------|-------------------------------|-----------------------|-------------------------------|
| | $\widehat{Pr}(Y = 1)$ | $(Y - \widehat{Pr}(Y = 1))^2$ | $\widehat{Pr}(Y = 1)$ | $(Y - \widehat{Pr}(Y = 1))^2$ |
| 1 | 0.51 | 0.2401 | 0.49 | 0.2601 |
| 0 | 0.49 | 0.2401 | 0.01 | 0.0001 |
| 1 | 0.51 | 0.2401 | 0.99 | 0.0001 |
| MSE | | 0.2401 | | 0.0868 |

Alternatives to MSE

Node Purity:

- ▶ Gini-Index:

$$G = \sum_{j=1}^{\#(\pi)} \hat{p}_j(1 - \hat{p}_j),$$

with $\hat{p}_j = \widehat{Pr}(Y = 1|l_j)$ and l_j indicating the terminal leaves for $j = 1, \dots, \#(\pi)$.

- ▶ Cross-Entropy:

$$D = - \sum_{j=1}^{\#(\pi)} \hat{p}_j \log(\hat{p}_j).$$

- Both node purity measures are minimized when each terminal leave contains only observations with $Y = 1$ or $Y = 0$.
- We use one node purity measure for complexity pruning (α -pruning).

Classification Tree

