Tree-based Methods[1]

Shaobo Li

University of Kansas

[1] Partially based on Hastie, et al. (2009) ESL, and James, et al. (2013) ISLR

Overview

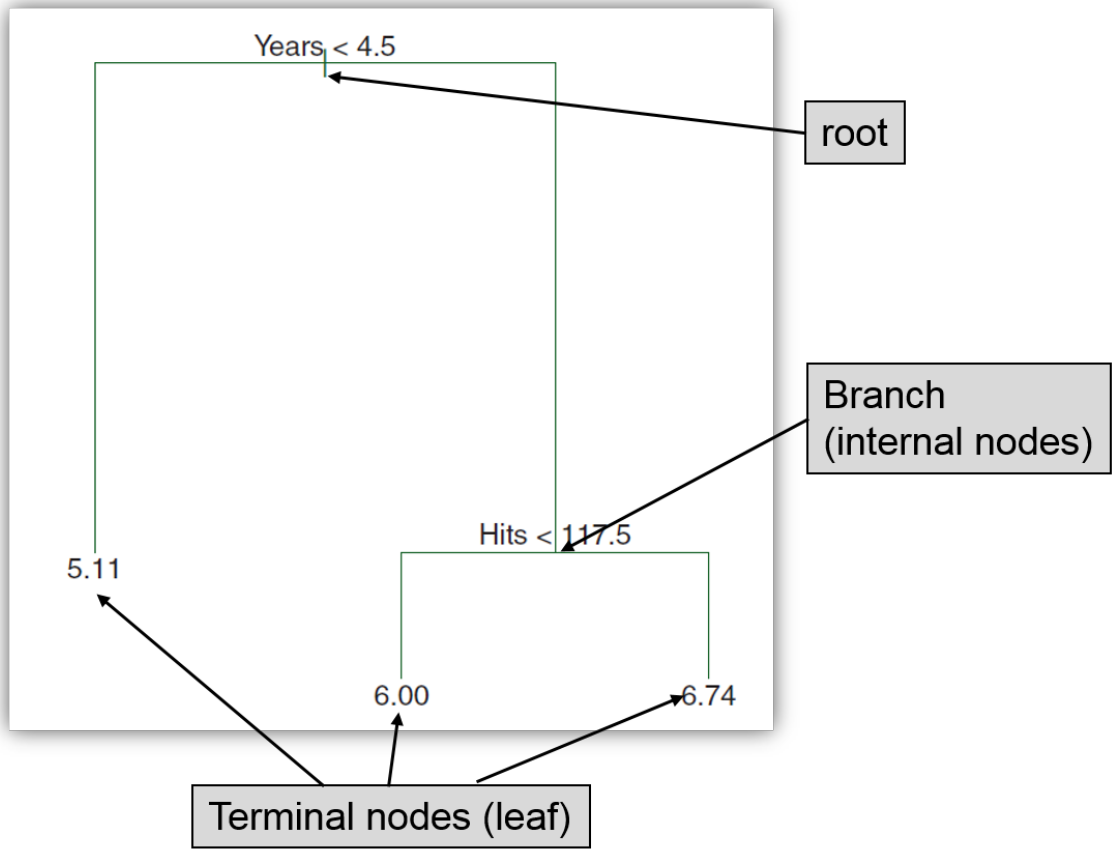
- Supervised learning
- Simple for interpretation
- Nonparametric method

We will talk about

- Classification and Regression Tree (CART)
- Bagging, Random Forest, and Boosting

A Simple Example of Regression Tree

- Predicting baseball player's log-salary using
  - X1: number of years he has played in the major leagues;
  - X2: number of hits he made in previous year



Prediction

- The values in terminal nodes
- Top-down
- At each splitting point, go left if TRUE

Data Partition — Stratification

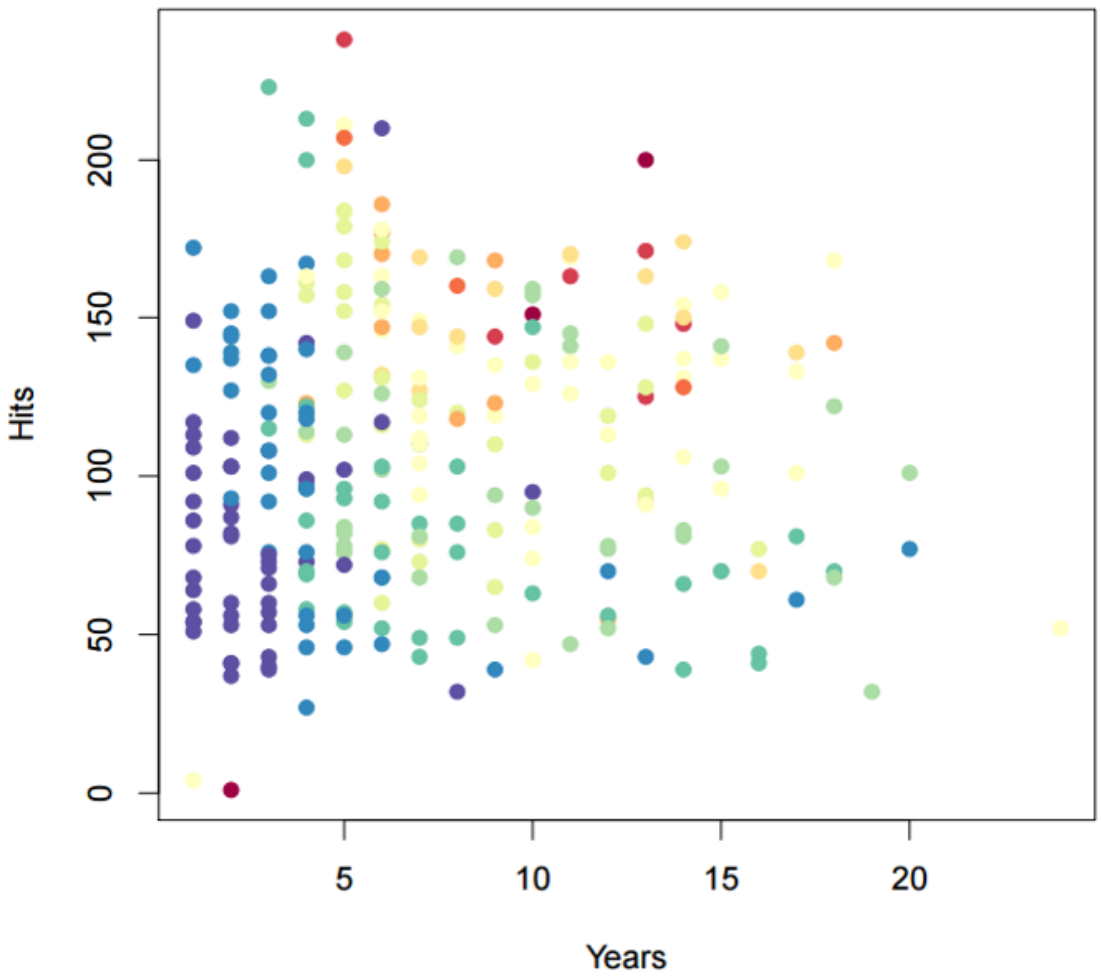
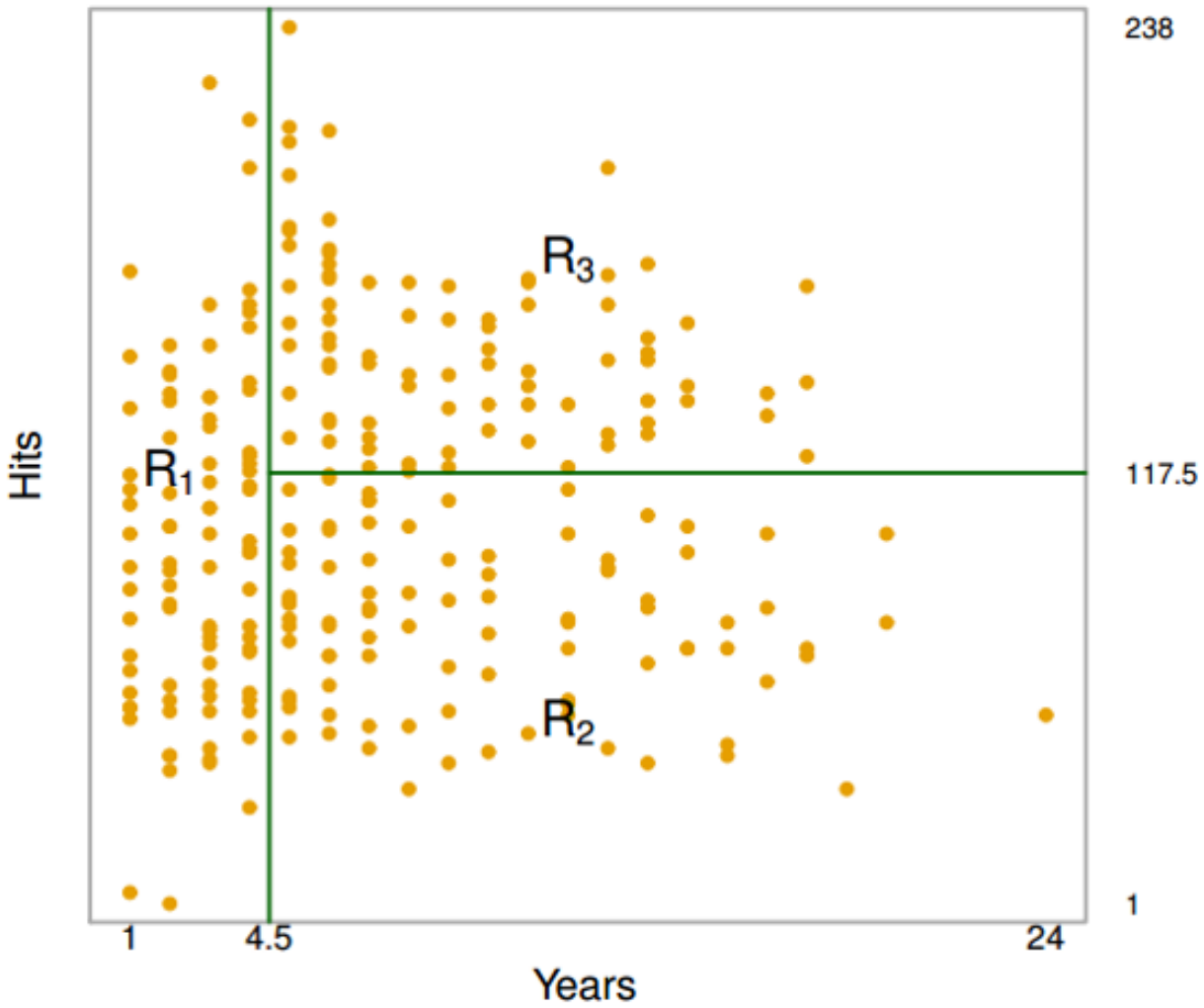


Figure: Salary is color-coded from low (blue, green) to high (yellow, red)

Data Partition — Stratification



A few technical question

- How to determine the optimal splitting point?
- What criteria should we use?
- When to stop growing the tree?

Optimal split (for a regression tree)

- For each split, we try to determine two regions R_1 and R_2 by finding the optimal splitting point $X_j = s$, that is

$$R_1 = \{\text{Data}|X_j \leq s\} \quad \text{and} \quad R_2 = \{\text{Data}|X_j > s\}$$

  This is done by scanning all possible splitting point for each X.
- Such optimal splitting point minimizes residual sum squares

$$\sum_{i \in R_1} (y_i - \bar{y}_{R_1})^2 + \sum_{i \in R_2} (y_i - \bar{y}_{R_2})^2$$

  where $\bar{y}_{R_1}$ and $\bar{y}_{R_2}$ are average of Y in each region.
- Repeat this process for each region until the decrease of RSS is not significant.

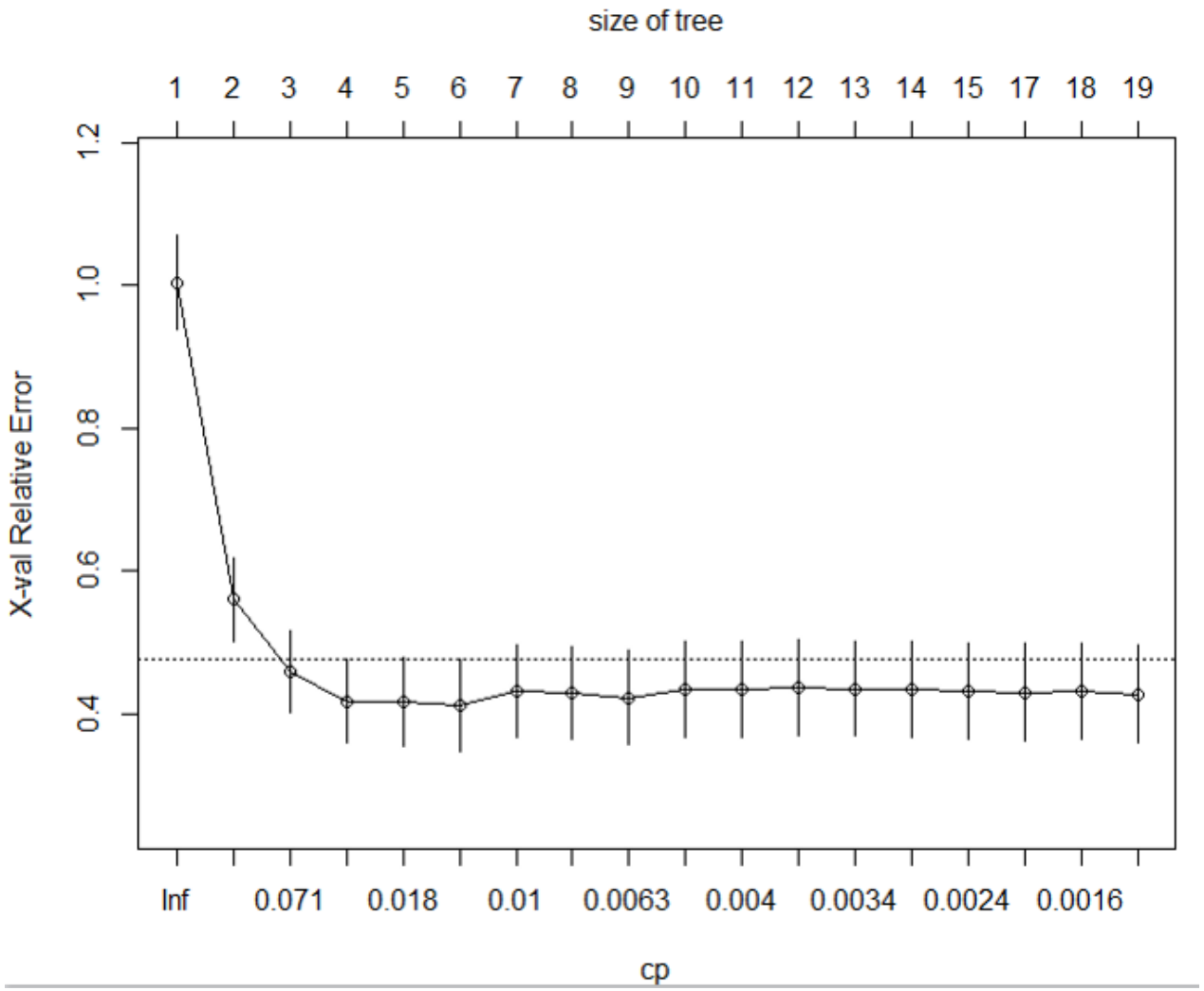
Stopping Strategy — Cost-Complexity Pruning

- To avoid overfitting!
- Suppose T_0 is a very large tree (overfitted tree). Denote T as a subtree of T_0, and $|T|$ as the size of that tree (number of terminal nodes). We minimize the cost complexity criterion

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

  where RSS_m is residual sum squares in region (terminal) m.
- Larger α penalizes the tree size ($|T|$) more
  - Recall how do we use AIC/BIC for variable selection
  - Every α corresponds to a unique optimal tree T_α.
- We use cross-validation to choose the optimal α.

Pruning a Tree in R

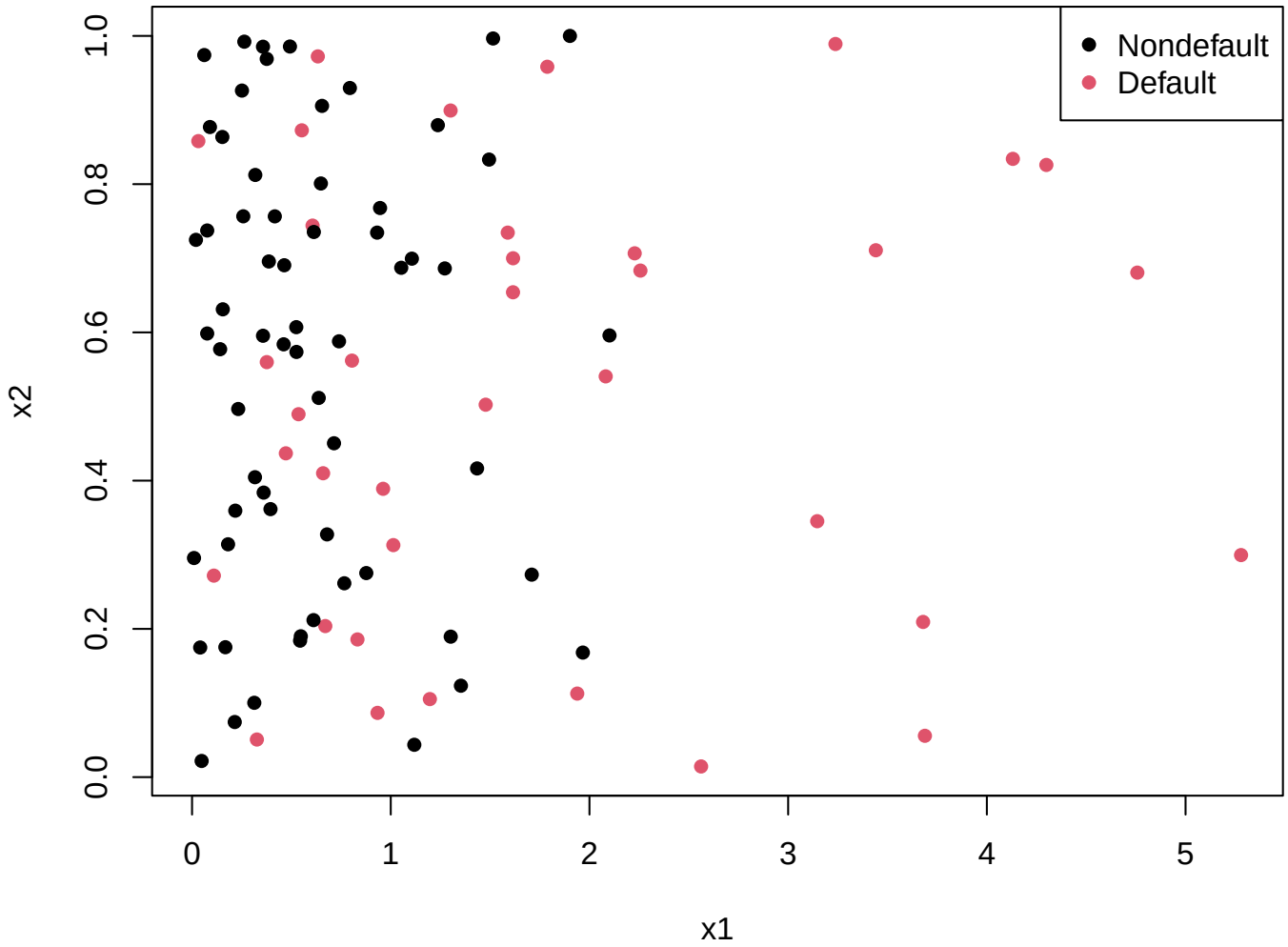


- cp is the complexity parameter α we have discussed.

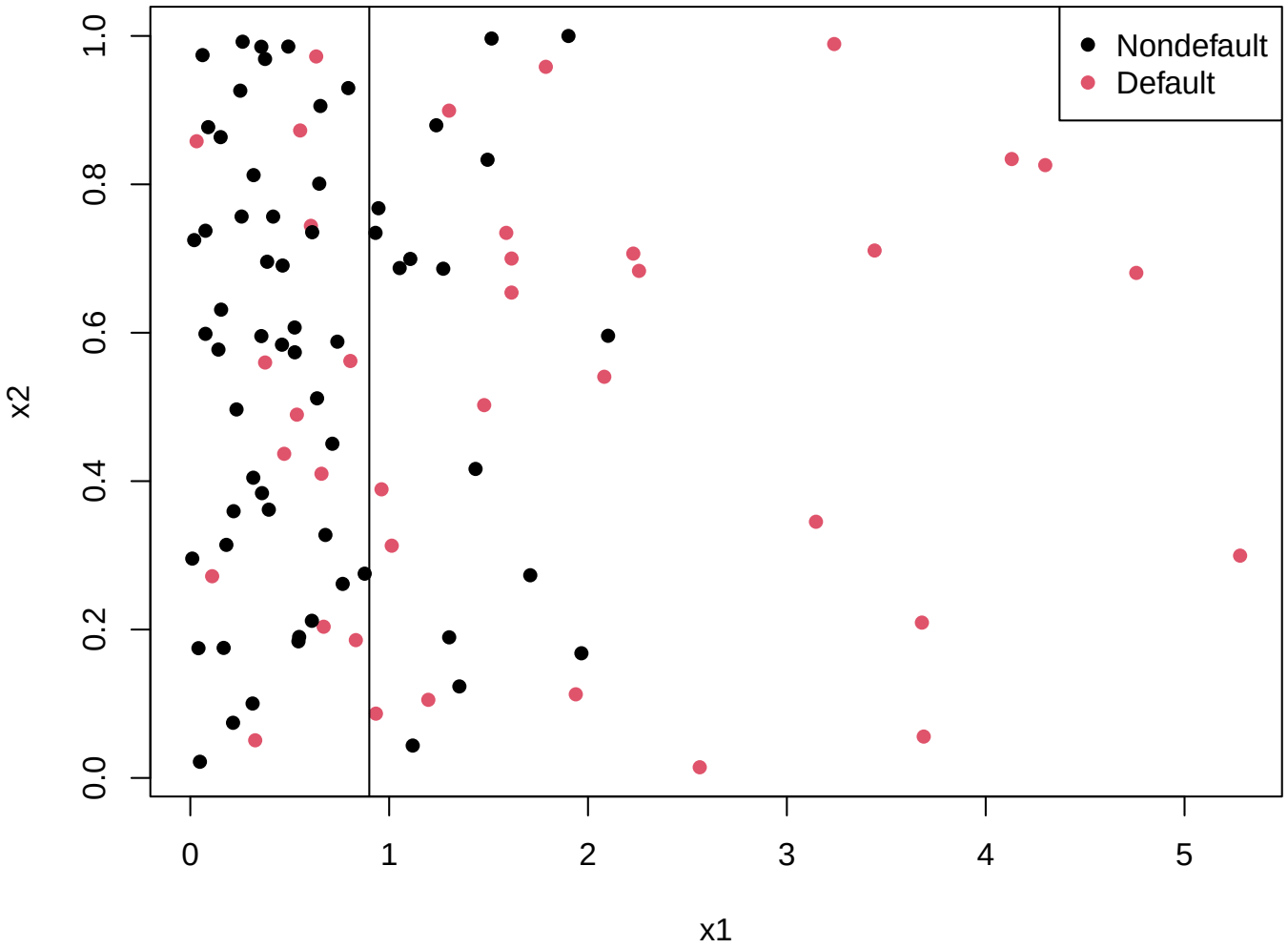
Classification Tree

- Outcome is categorical variable
- The same idea as for regression tree
- The only difference is the splitting criteria
- What should we use for the criteria?

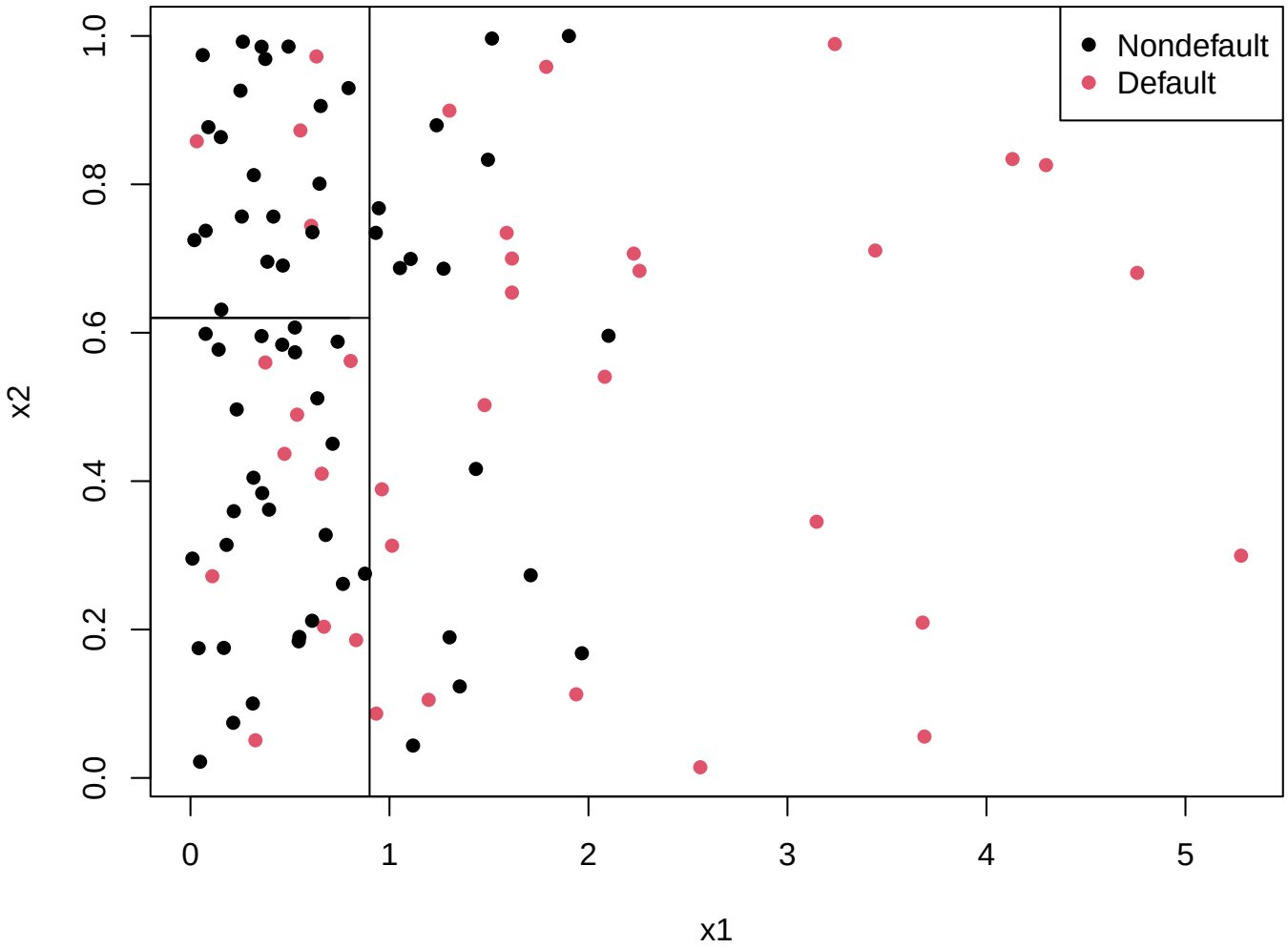
Illustration



Illustration



Illustration



Splitting Criteria for Classification Tree

- Misclassification rate
- Gini index

$$\text{Gini index} = \sum_{k \neq k'} \hat{p}_k \hat{p}_{k'} = \sum_{k=1}^{K} \hat{p}_k (1 - \hat{p}_k)$$

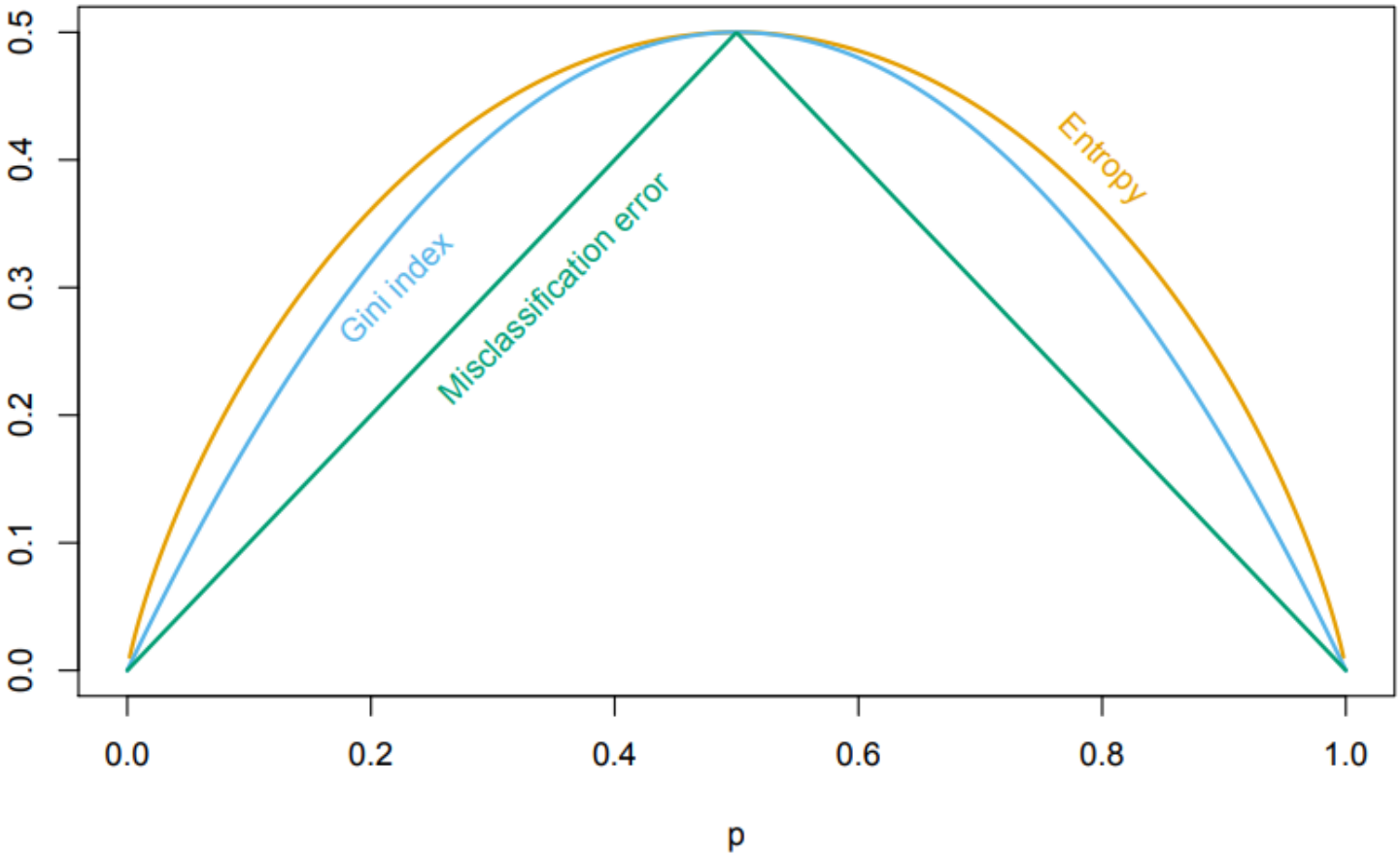
where $\hat{p}_k$ is the proportion of k category observation in one node.

- Entropy

$$\text{Entropy} = -\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k$$

For each split, gini index (or entropy) is calculated for two regions, and the weighted average is used as the criterion to be *minimized*.

Illustration



Example

- Suppose a node contains binary outcome (0 or 1) with each 400 observations.
- Split strategy 1: (300, 100) and (100, 300)
- Split strategy 2: (400, 200) and (0, 200)
- What is the misclassification rate, Gini index, and Entropy for each split strategy?

Categorical Predictor

- If a categorical predictor has q possible values, how many possible splits?
- Computationally infeasible for large q.
- Solution for binary outcome:
  - Sort the q classes according to the proportion of "1" in each of the q classes.
  - Split the ordered variable as if it is numeric variable.
  - This split is the optimal in terms of Gini index and Entropy.
- But we should avoid categorical variable which has too many categories. Why?

Asymmetric Cost

- Similar idea as we discussed in last lecture
- Define a loss matrix with different weights in misclassified cells.

	Pred=1	**Pred=0**
True=1	0	w_0
True=0	w_1	0

- Loan application example (1=default): $w_1 < w_0$
- The weights are incorporated during each split

Bagging

- Limitation of a single tree: high variance, unstable
  - Change of points in the sample would lead to different split.
  - Current split heavily relies on previous splits.
- Bagging — Bootstrap aggregation

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

  - Leo Breiman (1996)
  - Fit many trees on bootstrap samples, and then take average.
  - Variance can be significantly reduced
- For classification tree, how to take average?

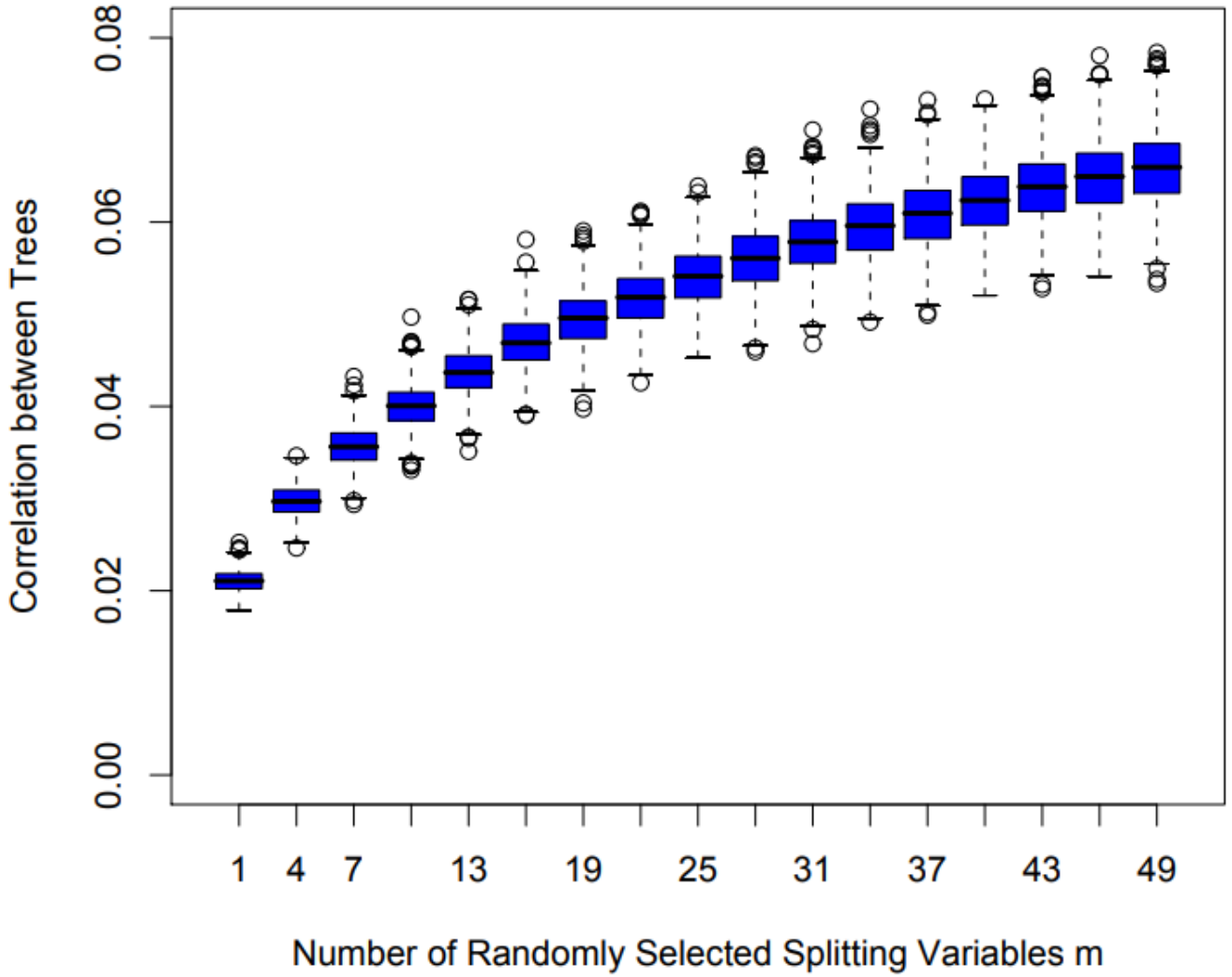
Out-of-bag (OOB) Prediction

- Similar to cross-validation
- A type of testing error — to prevent overfitting
- How OOB is computed:
  For each observation i:
  1. Identify all bootstrap samples that did not include i.
  2. Predict i's observation using all the trees based on those bootstrap samples.
  3. Average (for regression) or majority vote (for classification) across those trees to get the OOB prediction for observation i.

Random Forests

- Improvement of Bagging
  - The trees in Bagging are correlated to each other
  - It may add variance and prediction error
- Goal: decorrelate a series of trees while maintaining the strength of ensemble.
- How RF works: randomly choose m predictors as candidate splitting variables for each split
- See the inventor's (Leo Breiman) website for more details.

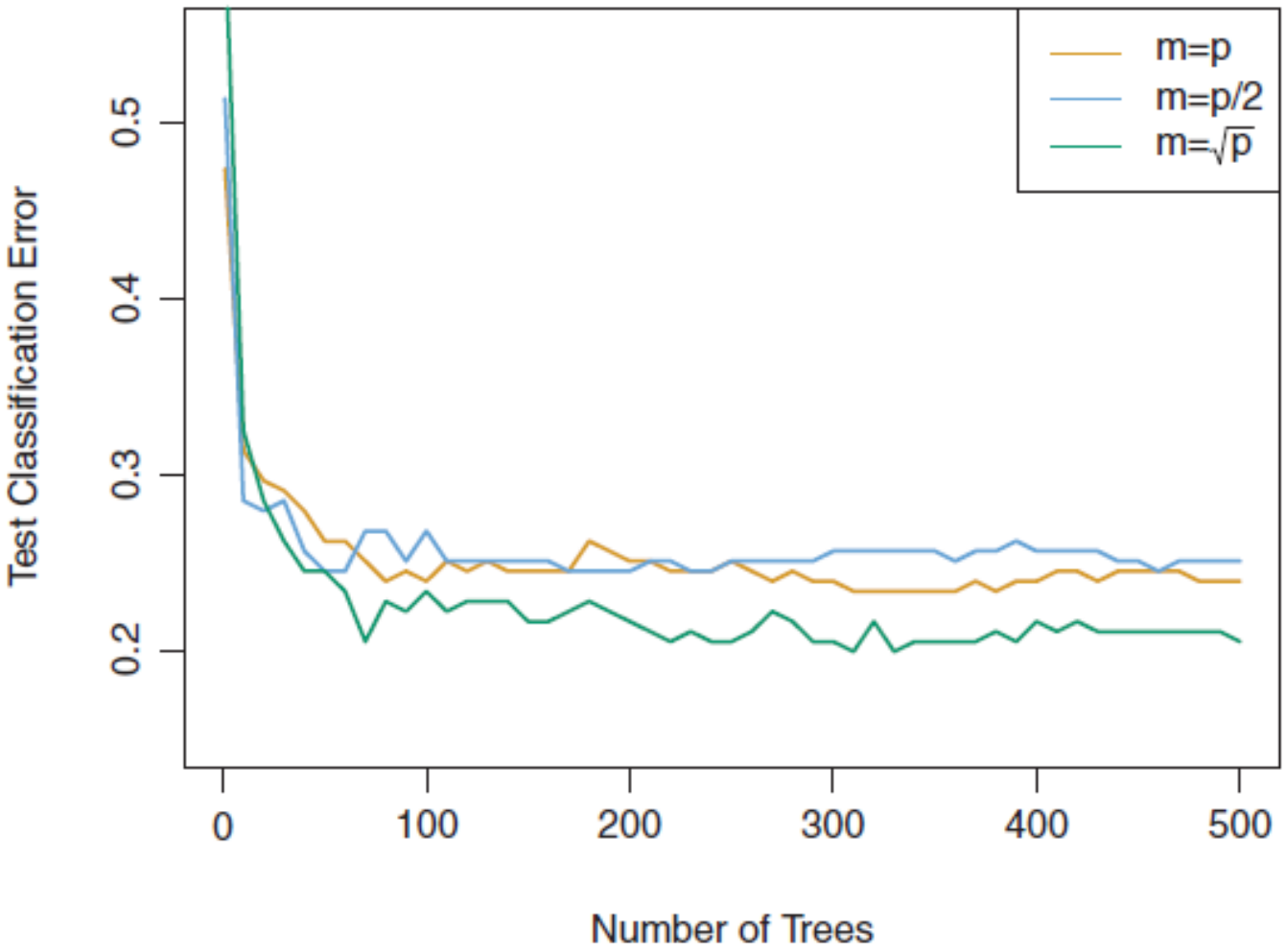
Illustration of De-correlation



Choice of m

- Smaller m decreases the correlation, but also decreases strength
- Larger m increases the correlation, but also increases strength
- Use OOB error to determine the optimal m
- By default, $m \approx \sqrt{p}$ for classification and $p/3$ for regression

Change of Testing Error across m



Gradient Boosting

- Ensemble method
- Based on small trees (weak learner)
- Additive – adding up all small trees
- An *illustration*
- Pros:
  - Powerful algorithm, high prediction accuracy
  - Small variance (advantage of ensemble methods)
- Cons:
  - Computational expensive for large data
  - Lack of interpretability
  - May overfit data

Boosting Regression Tree

- Additive: sequentially grow the tree
- Combination of many small trees
- Boosting for regression tree
  1. Set $\hat{f}(x) = 0$, and $r_i = y_i$
  2. For $b = 1, \ldots, B$, repeat:
     - Fit a small tree $\hat{f}^{(b)}(x)$ with d splits based on the training sample
     - Update residual $r_i \leftarrow r_i - \lambda \hat{f}^{(b)}(x)$, and treat this residual as new response, where λ is shrinkage parameter
     - Update the tree $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^{(b)}(x)$
  3. Output the boosted regression tree

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^{(b)}(x)$$

Gradient Boosting Machine (GBM)

- A more generalized version.
- User supplied differentiable loss function (e.g. squared loss)

$$\sum_{i=1}^{n} L(y_i, f(x_i)) = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

- The negative **gradient** for each $L(y_i, f(x_i))$ is essentially residual or pseudo residual.
- Then fit a new regression tree to approximate the gradient (fit the residual).
- For binary classification, the loss can be binomial deviance (equivalent to cross entropy, see ESL p346), but the weak learner in each iteration is still a regression tree that approximate the gradient of the deviance $(y_i - p_i)$.

The algorithm (ESL p387)

Algorithm 10.4 *Gradient Boosting for K-class Classification.*

1. Initialize $f_{k0}(x) = 0,\ k = 1, 2, \ldots, K$.
2. For m=1 to M:
   (a) Set
   $$p_k(x) = \frac{e^{f_k(x)}}{\sum_{\ell=1}^{K} e^{f_\ell(x)}},\ k = 1, 2, \ldots, K.$$
   (b) For $k = 1$ to K:
      i. Compute $r_{ikm} = y_{ik} - p_k(x_i),\ i = 1, 2, \ldots, N$.
      ii. Fit a regression tree to the targets $r_{ikm},\ i = 1, 2, \ldots, N$, giving terminal regions $R_{jkm},\ j = 1, 2, \ldots, J_m$.
      iii. Compute
      $$\gamma_{jkm} = \frac{K-1}{K} \frac{\sum_{x_i \in R_{jkm}} r_{ikm}}{\sum_{x_i \in R_{jkm}} |r_{ikm}|(1 - |r_{ikm}|)},\ j = 1, 2, \ldots, J_m.$$
      iv. Update $f_{km}(x) = f_{k,m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jkm} I(x \in R_{jkm})$.
3. Output $\hat{f}_k(x) = f_{kM}(x),\ k = 1, 2, \ldots, K$.

AdaBoost — Boosting for Classification

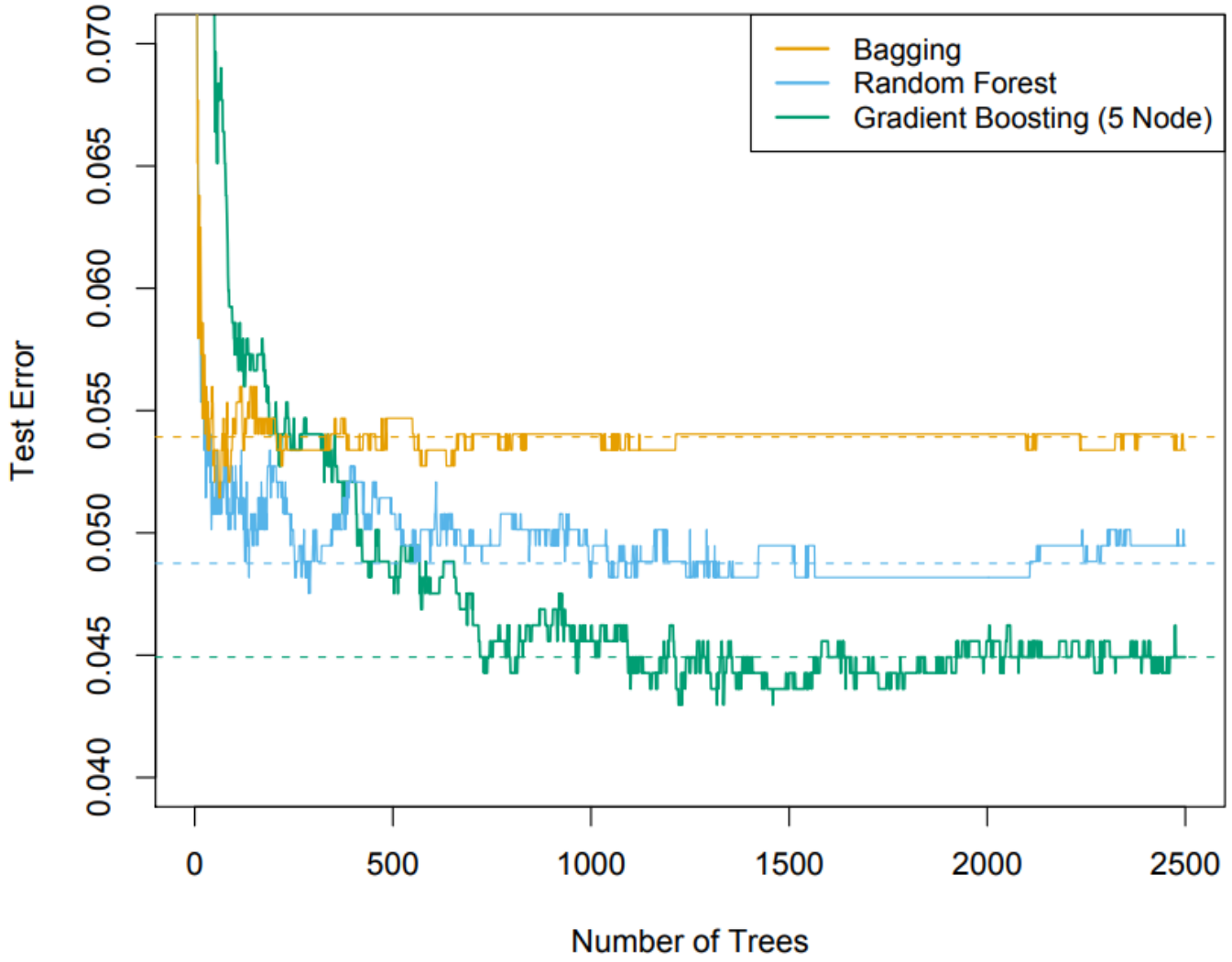
- Originally designed for classification $Y = \{-1, 1\}$
- Algorithm:
  1. Assign equal weights to all observations $w_i = 1/N$
  2. For $b = 1, \ldots, B$, repeat:
     - Fit a classifier $G_b(x)$ to the training sample using weights w_i
     - Compute weighted misclassification error

$$\text{err}_b = \frac{\sum_{i=1}^{N} w_i \mathbb{I}(y_i \neq G_b(x_i))}{\sum_{i=1}^{N} w_i}$$

     - Compute $\alpha_b = \log((1 - \text{err}_b)/\text{err}_b)$
     - Update weights $w_i \leftarrow w_i \exp\{\alpha_b \mathbb{I}(y_i \neq G_b(x_i))\}$, and normalize such that the sum equals to 1.
  3. Output $G(x) = \text{sign}\{\sum_{b=1}^{B} \alpha_b G_b(x)\}$.

Here is the StatQuest video that may help you to understand the idea.

Peformance Comparison



XGBoost – Extreme Gradient Boosting

- Scalable tree boosting
- Recently developed by Tianqi Chen in 2016.
- One of the best supervised learning algorithms (Machine Learning Challenge Winning Solutions)

General idea

We omit the mathematical details here. For more detail, please see the official tutorial, and original paper.

- Similar to traditional gradient tree boosting.
- Added regularization which prevents overfitting.
- Optimized the utility of computing power.