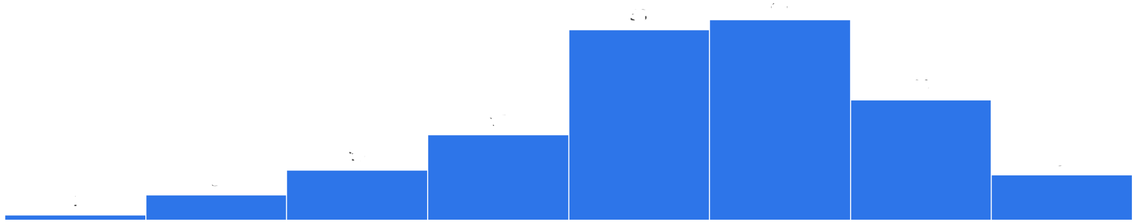
DS-GA 1003: Machine Learning

Lecture 7: Decision Trees & Random Forests

Slides adapted from material from David Rosenberg.

Logistics & Announcements

Midterm performance. Median right at 69.5, boost of +2.5. “Midterm grades” on Albert will be an “optimistic projection.”



Project proposal deadline. Due in a week, **March 31 11:59 PM!**

PS 3 extension. New due date is **March 31 11:59 PM**.

Partial “clobber” from final project. If project grade > midterm, **shift 3%** to project grade.

Classical ML to Modern ML shift. Sam’s last lecture next week; Nick taking over on April 7th.

Outline

Decision Tree Basics

Splitting Criterion

Decision Trees vs. Linear Models

Bootstrap Method

Ensemble Method: Bagging

Ensemble Method: Random Forests

Decision Trees

Departure from usual setup

Hypothesis class: $\mathscr{H} = \{$ decision trees of fixed size $\}$

Empirical risk minimization:

$$\min_{h \in \mathscr{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(h(x^{(i)}), y^{(i)})$$

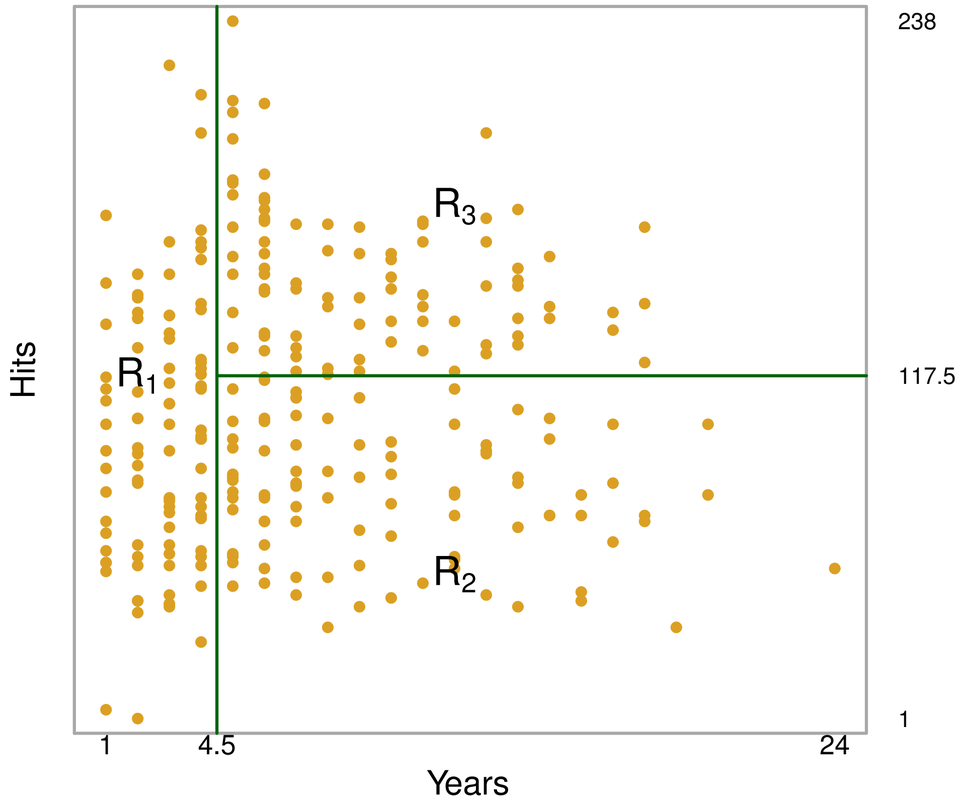
ERM is intractable for learning decision trees! Instead, to find decision trees, we will focus on a **greedy/top-down splitting algorithm**.

Keep in mind comparison to linear hypotheses: $\mathscr{H} = \{w^\top x + w_0 : w \in \mathbb{R}^d, w_0 \in \mathbb{R}\}$ today.

Adapts to all problems we've seen: $\mathscr{Y} = \mathbb{R}$ (regression) and $\mathscr{Y} = \{1, \ldots, K\}$ (multi-class classification).

Regression Trees

Example: Predicting Baseball Player Salaries



Adapted from *Introduction to Statistical Learning* (James, Witten, Hastie, Tibshirani), Ch. 8.

Classification Trees

Motivation for decision trees

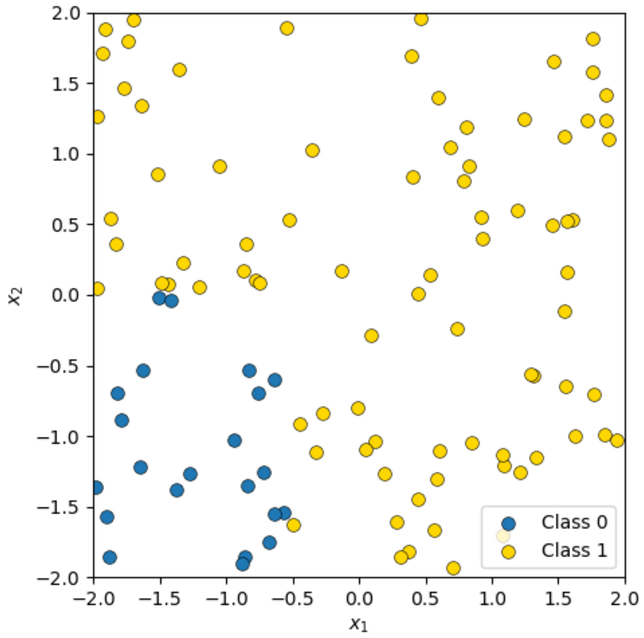
So far, we've seen two main classification methods (*SVM, Logistic regression*) using our familiar linear hypothesis class:

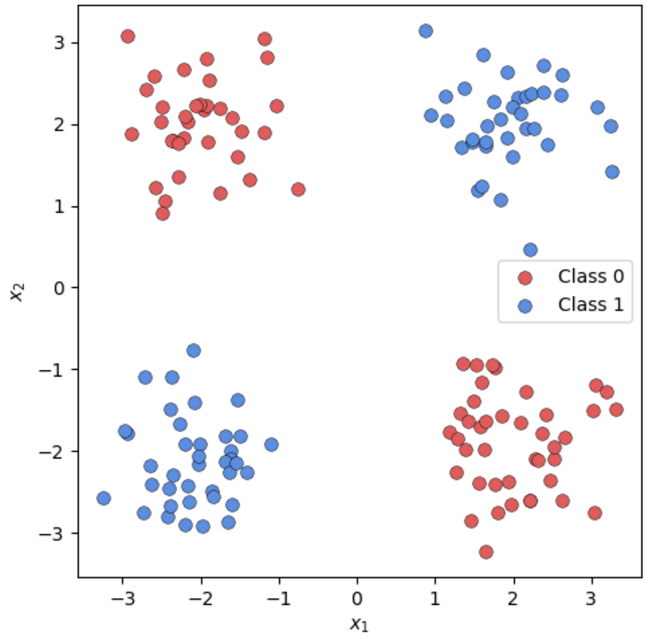
$$\mathscr{H} = \{w^\top x + w_0 : w \in \mathbb{R}^d, w_0 \in \mathbb{R}\}.$$

We've dealt with nonlinearities using kernels.

> **Main idea**: linearly separable in higher dimensional space.

How about *inherently* nonlinear methods for ML?





Decision Tree Structure

Binary Trees & Features

d features, n examples…

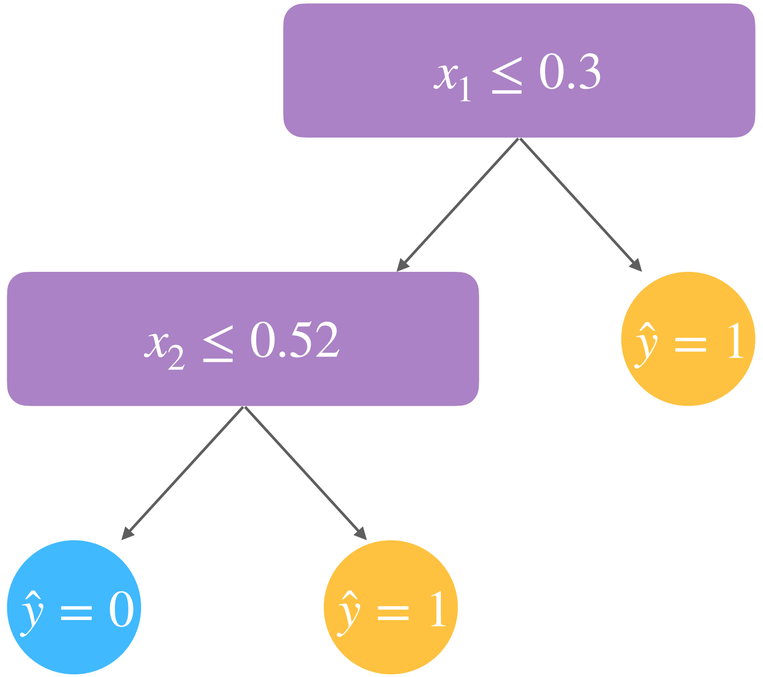
Focus on **binary trees**.

Nodes contain a subset of data points.

Data splits created by each node involve only a *single* feature $j \in [d]$.

Continuous features. Splits of form $x_j \leq t$.

Discrete features. Split on possible partitions into two sets.



Decision Tree Structure

Prediction

Leaf nodes associated with output from $\mathcal{Y}$.

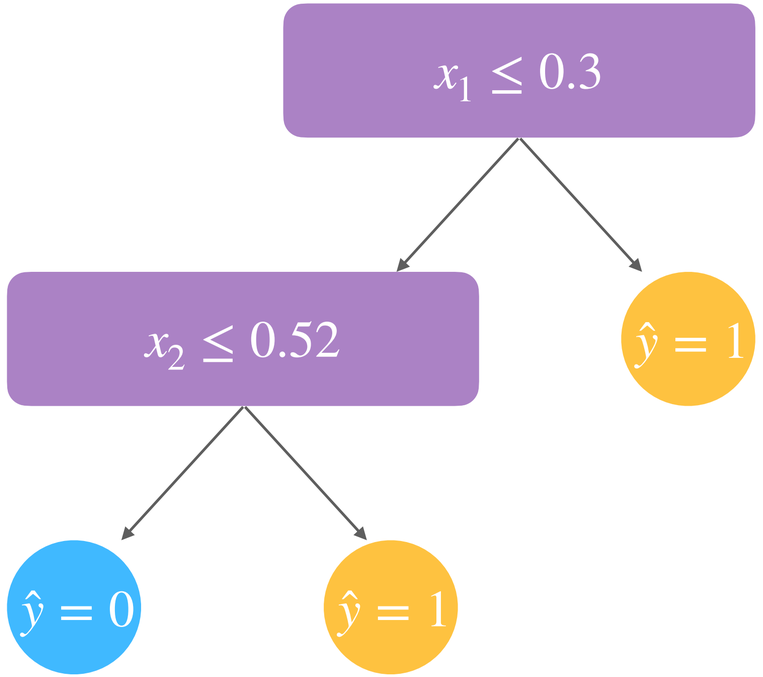
To compute the prediction on x:

Start at root node.

If current node is leaf node: return label.

Else if predicate on x is true: recurse on left child.

Else: recurse on right child.



Adapted from Daniel Hsu's lecture notes for COMS 4771 at Columbia University.

Constructing the tree

Regression Trees

We will focus on regression trees first and then discuss classification trees.

Goal. Find regions $R_1, \ldots, R_J$ that minimize $\sum_{j=1}^{J} \sum_{i \in R_j} (y^{(i)} - \hat{y}_{R_j})^2$ subject to complexity constraints.

Problem. Finding the *optimal* binary tree is computationally intractable.

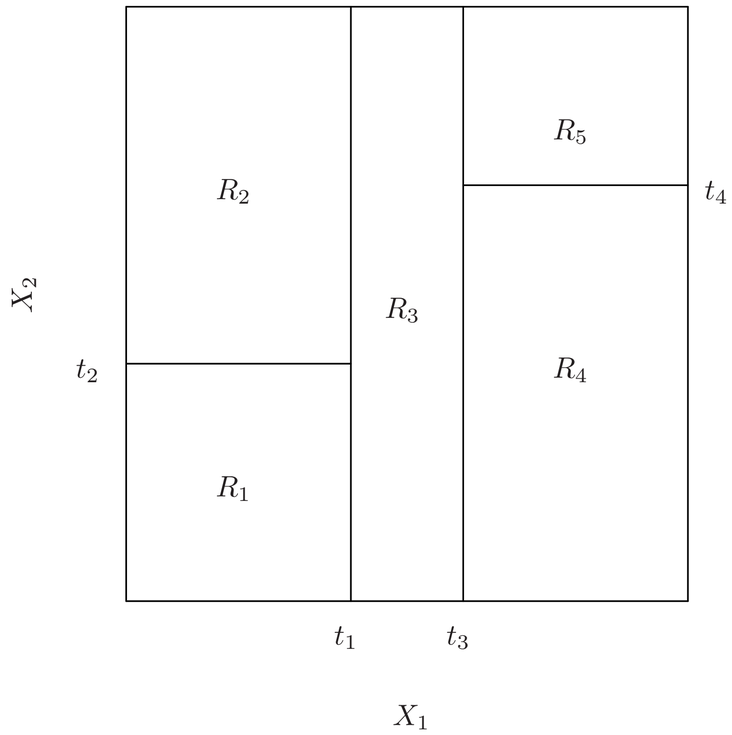
Solution. Greedy algorithm: start from root, repeat until a stopping criterion reached (e.g. max depth). Find the non-leaf node that results in "best" split. Greedy solution is "good enough."

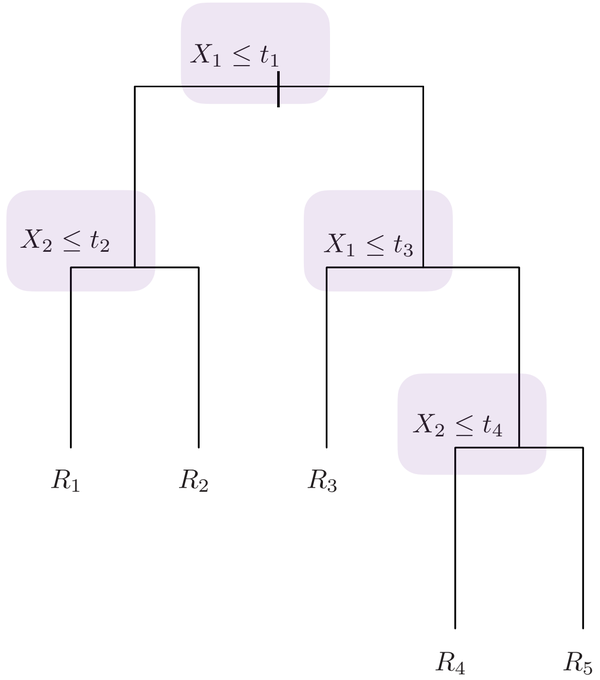
Split regions defined by previous non-leaf nodes.

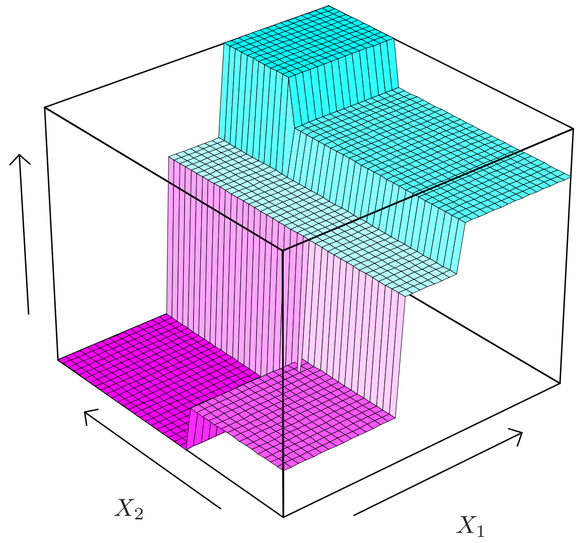
Prediction. Prediction is the mean value of a leaf node: $\hat{y}_{R_j} = \frac{1}{R_j} \sum_{i \in R_j} y^{(i)}$.

Prediction in regression trees

Three Views







Question: How do we choose where to split?

Adapted from *Introduction to Statistical Learning* (James, Witten, Hastie, Tibshirani), Ch. 8.

Outline

Decision Tree Basics

Splitting Criterion & Overfitting

Decision Trees vs. Linear Models

Bootstrap Method

Ensemble Method: Bagging

Ensemble Method: Random Forests

Greedy Algorithm

Step 1 Split

Goal: Find $R_1, \ldots, R_J$ that minimize

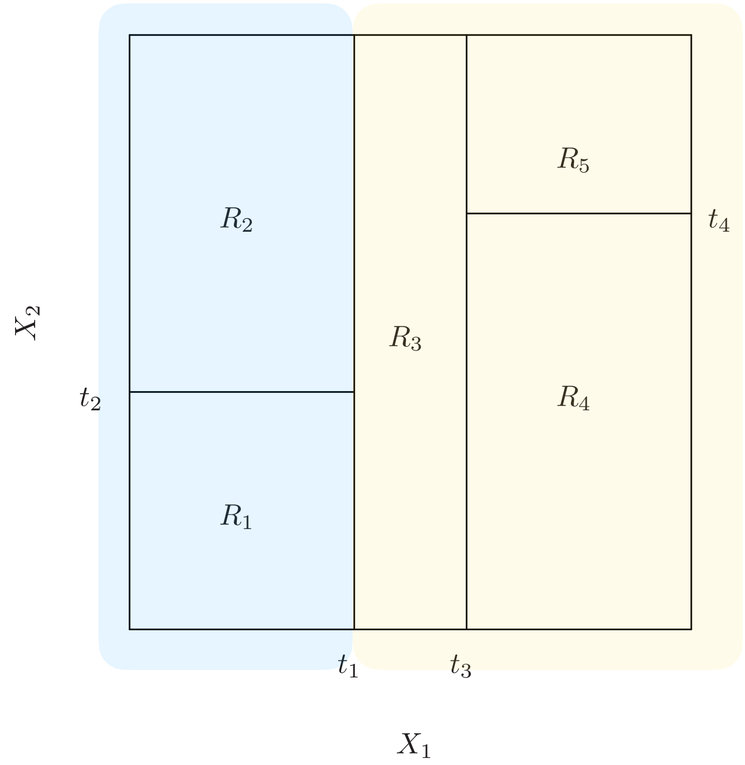
$$\sum_{j=1}^{J} \sum_{i \in R_j} (y^{(i)} - \hat{y}_{R_j})^2.$$

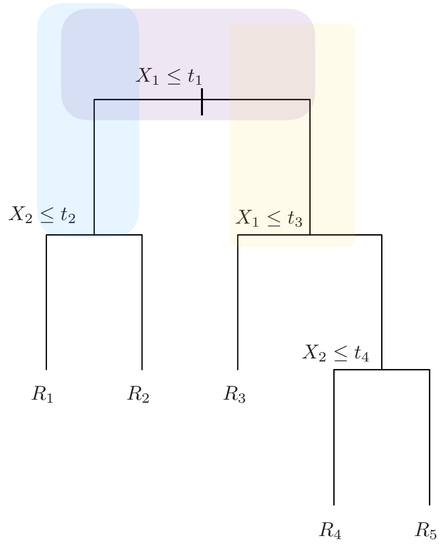
For all features $x_1, \ldots, x_d$ and all possible splitpoints t_1 for each $j \in [d]$:

Find $j \in [d]$ and t_1 minimizing:

$$\sum_{i:x^{(i)} \in R_1(j,t_1)} (y^{(i)} - \hat{y}_{R_1})^2 + \sum_{i:x^{(i)} \in R_2(j,t_1)} (y^{(i)} - \hat{y}_{R_2})^2$$

$$R_1(j, t_1) = \{x : x_j < t_1\}, R_2(j, t_1) = \{x : x_j \geq t_1\}.$$





Greedy Algorithm

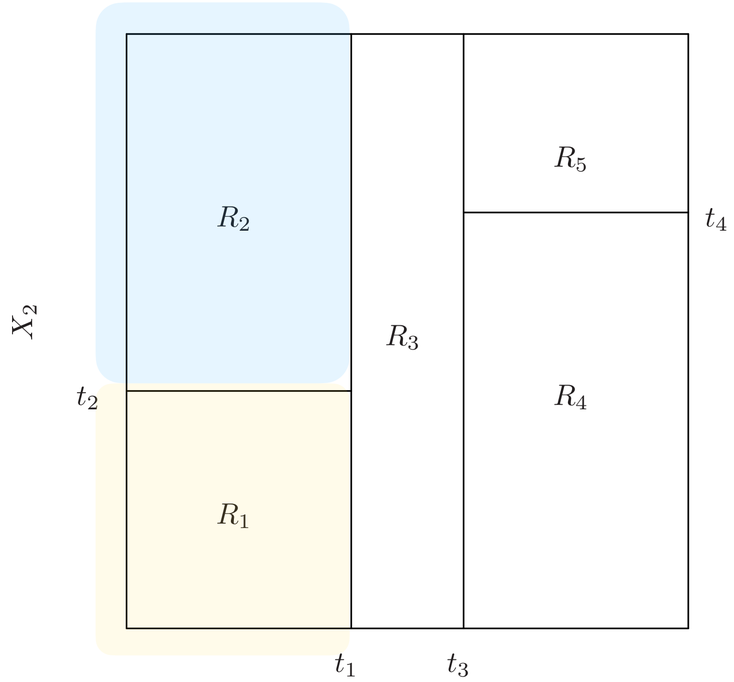
Step 2 split onward…

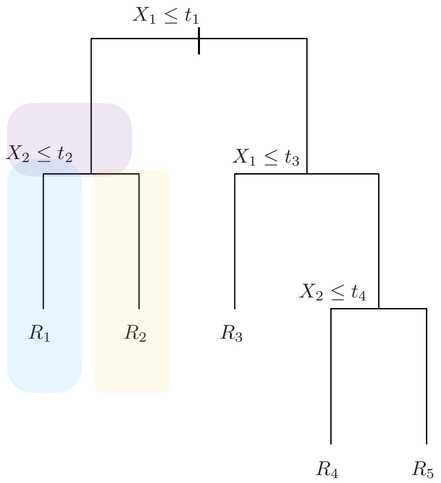
Goal: Find $R_1, \ldots, R_J$ that minimize

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

For all features $x_1, \ldots, x_d$ and all possible splitpoints s for each $j \in [d]$:

Repeat, finding j and s that minimize criterion only on examples **from one of the two previously identified regions**.





Splitting Criterion

Dealing with infinitely many split points

If we enumerated all features and possible split points for each feature, there are possibly infinitely many split points.

But: suppose we consider splitting on feature x_j (where $j \in [d]$) and let $x_j^{(1)}, \ldots, x_j^{(n)}$ be the sorted values of the jth feature.

We only need to consider split points between two adjacent values, and any split point on the interval $(x_j^{(r)}, x_j^{(r+1)})$ will result in the same prediction.

Common to split halfway between two adjacent values (so $n - 1$ possible split points):

$$\frac{1}{2}(x_j^{(r)} + x_j^{(r+1)}) \text{ where } r = 1,\ldots,n-1.$$

Overfitting

For decision trees

What will happen if we keep on splitting on more and more regions?

Every data point will be in its own region – **overfitting**.

When should we stop splitting?

Limit total number of nodes.

Limit number of leaf nodes.

Limit tree depth.

Require min. number of points in leaves.

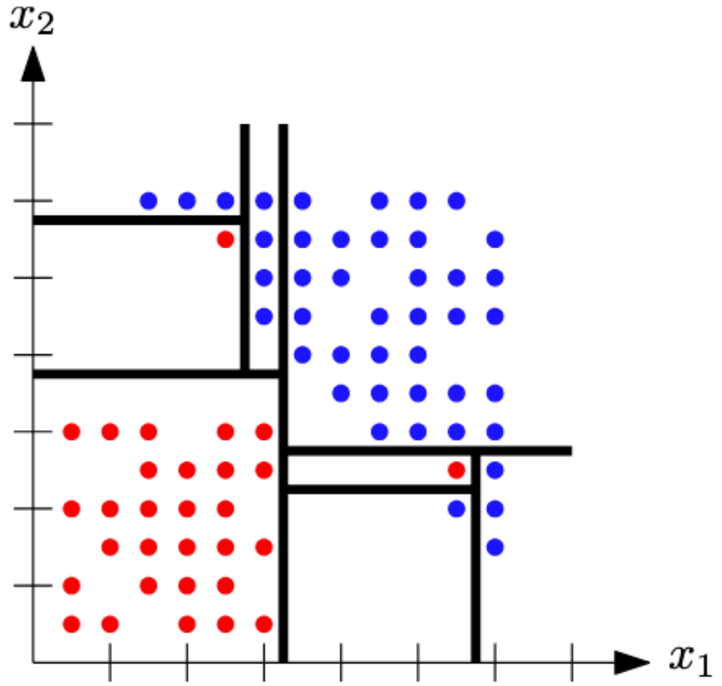


Figure from Daniel Hsu's lecture notes for COMS 4771 at Columbia University.

Overfitting

Controlling complexity

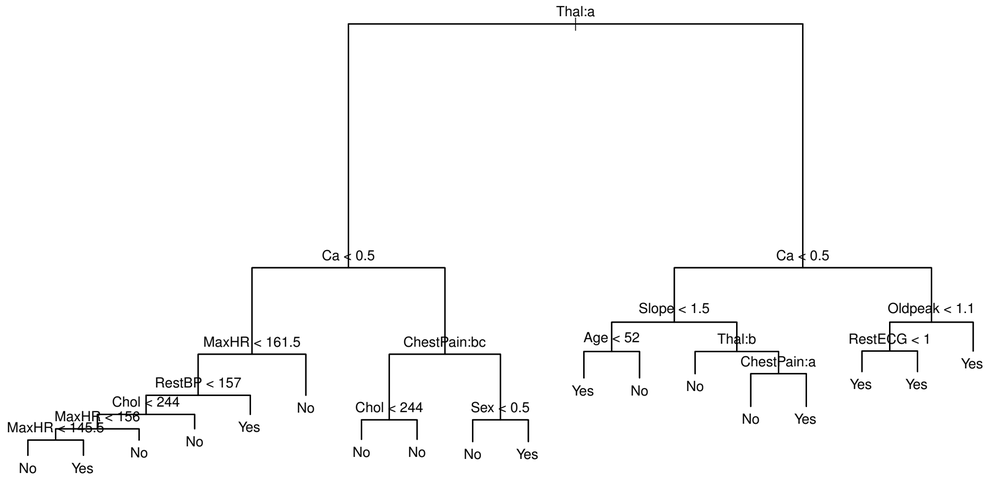
What will happen if we keep on splitting on more and more regions?

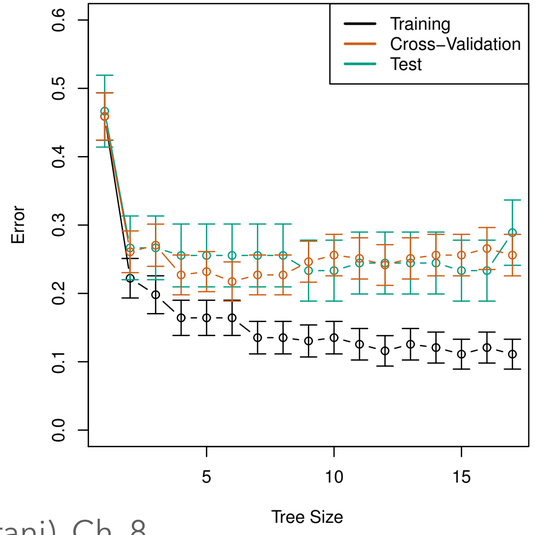
> Every data point will be in its own region – **overfitting**.

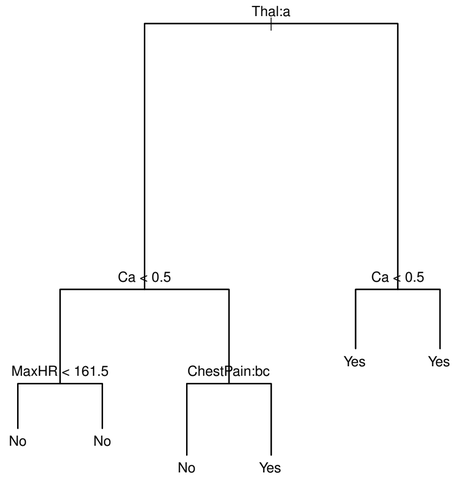
Need to control complexity of $\mathscr{H}$ via "regularization"!

$$\mathscr{H} = \{\text{depth-5 decision trees}\}$$

$$\mathscr{H} = \{\text{depth-3 decision trees}\}$$







Adapted from *Introduction to Statistical Learning* (James, Witten, Hastie, Tibshirani), Ch. 8.

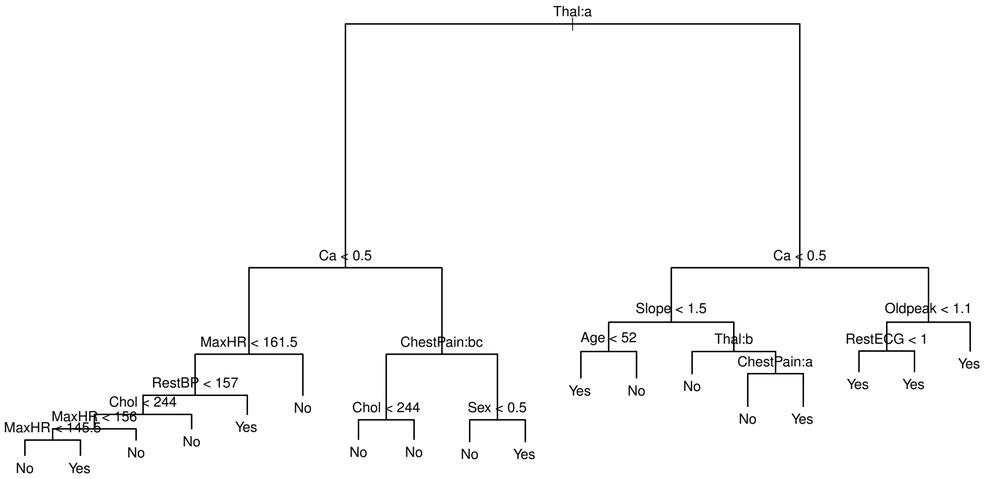
Overfitting

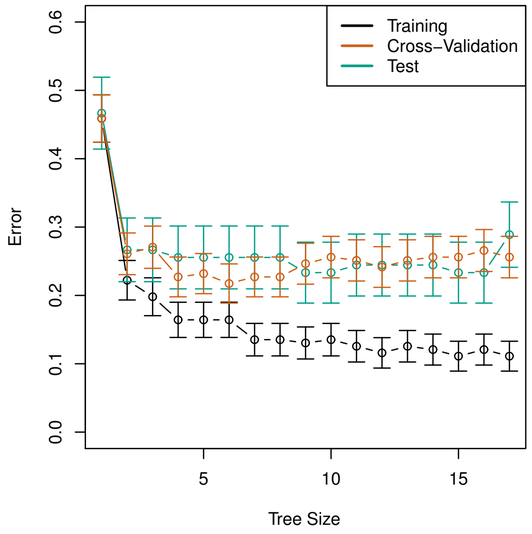
Pruning

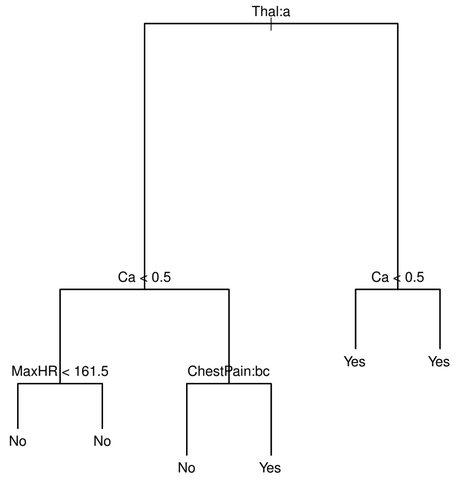
Pruning (approach in CART algorithm of Breiman et al. 1984):

1. Build a really big tree (e.g. until all regions have ≤ 5 points).

2. *Prune* the tree back greedily, potentially all the way to root, until validation performance starts decreasing.







Splitting Criterion

For classification

For classification trees: predict the **majority label** in each region.

Which of the following splits are better?

Split 1: R_1 : $8+$ examples and $2-$ examples R_2 : $2+$ examples and $8-$ examples.

Split 2: R_1 : $6+$ examples and $4-$ examples R_2 : $4+$ examples and $6-$ examples.

Intuition: Want to produce *pure* nodes, i.e. nodes where most instances have the same class.

Classification Trees

Misclassification Error in a Node

Decision trees let us easily handle **multiclass classification:** $\mathcal{Y} = \{1,2,\ldots,K\}$.

Let node m represent region R_m with N_m examples.

Denote the proportion of observations in R_m with class k by:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{i:x^{(i)} \in R_m} \mathbf{1}\{y^{(i)} = k\}.$$

Predict the majority class in node m:

$$k(m) = \operatorname*{argmax}_k \hat{p}_{mk}$$

Classification Trees

Node Impurity Measures

Three popular measures of node impurity for leaf node m:

1. Misclassification error: $1 - \hat{p}_{mk(m)}$
2. Gini index: $\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
3. Entropy: $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

In practice: Gini index and Entropy are numerically similar to each other, and both work better in practice than misclassification error.

Classification Trees

Splitting Criterion: Node Impurity

Consider a potential split that produces nodes R_1 and R_2:

If we have N_1 points in R_1 and N_2 points in R_2…

Let $Q(R_1)$ and $Q(R_2)$ be the node impurity measures for each node.

Greedy/top-down algorithm again iterates through $j \in [d]$ and splitpoints to find split minimizing the *weighted average of node impurities:*

$$\frac{N_1 Q(R_1) + N_2 Q(R_2)}{N_1 + N_2}$$ (and repeat!)

Outline

Decision Tree Basics

Splitting Criterion & Overfitting

Decision Trees vs. Linear Models

Bootstrap Method

Ensemble Method: Bagging

Ensemble Method: Random Forests

Decision Tree Overview

Interpretability

Trees are often easier to visualize and explain than other classifiers (even linear regression).

Smaller trees are interpretable; larger trees maybe not so much.

Decision Tree Overview

vs. Linear Models

Trees may have to work hard to capture linear decision boundaries…

But can easily capture certain nonlinear ones.

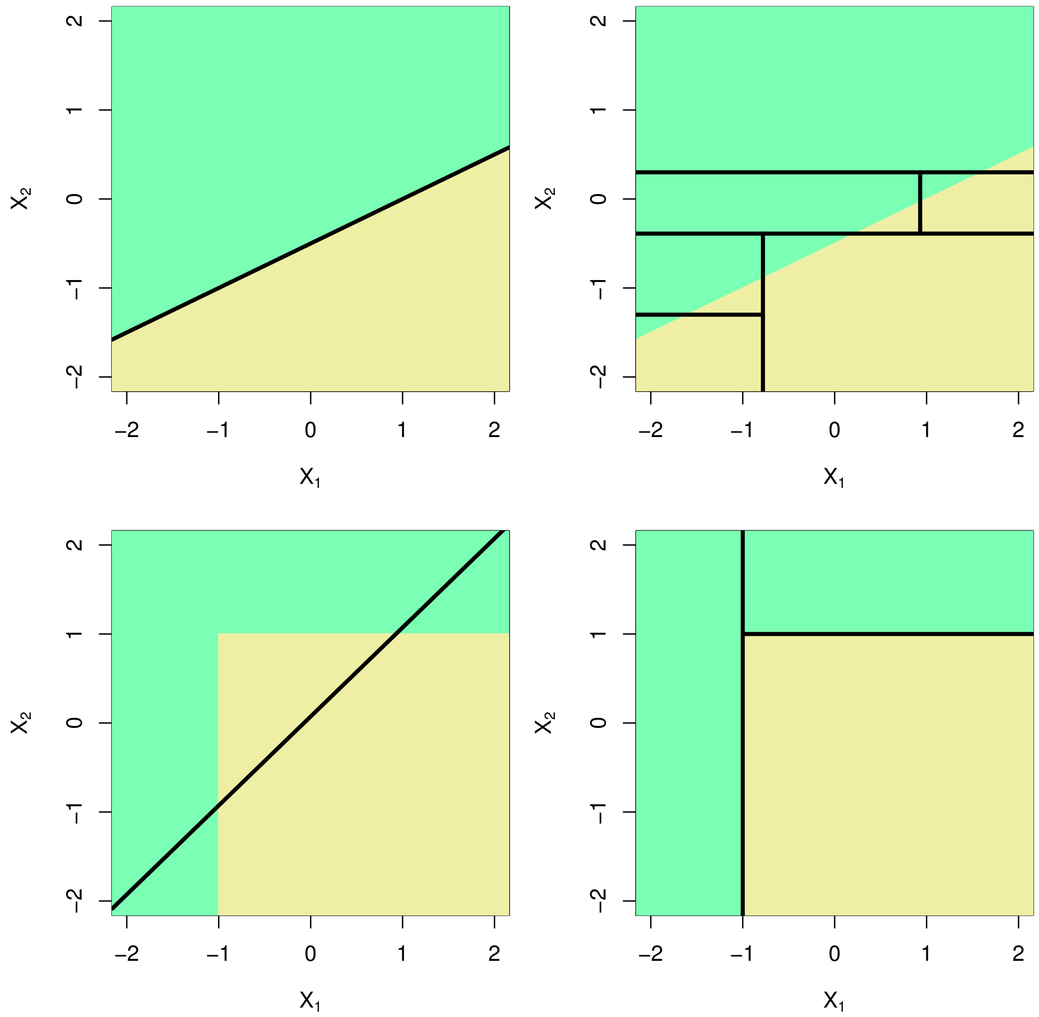


Figure from *Introduction to Statistical Learning* (James, Witten, Hastie, Tibshirani), Ch. 8.

Decision Tree Overview

Summary

Inherently nonlinear: decision boundary that results from splitting may end up being quite complicated.

Non-metric: do not rely on the geometry of the space (no inner products/distances).

Non-parametric: make no assumptions about distribution of data.

Additional pros:

Interpretable and simple to understand.

Cons:

Struggle to capture linear decision boundaries.

Have high variance and tend to **overfit**. Sensitive to small changes in the training data.

Outline

Decision Tree Basics

Splitting Criterion & Overfitting

Decision Trees vs. Linear Models

Bootstrap Method

Ensemble Method: Bagging

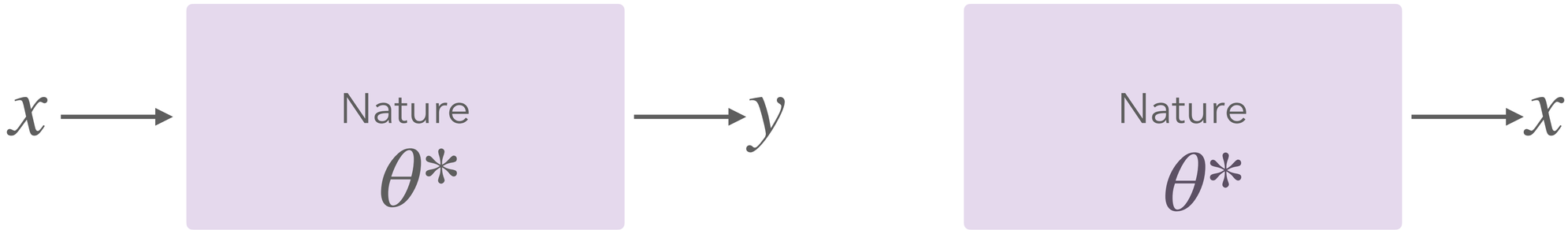
Ensemble Method: Random Forests

Statistical Estimator

Intuition

A (statistical) estimator is a “best guess” at some (unknown) quantity of interest (the estimand) using observed data.

The quantity doesn’t have to be a single number; it could be, for example, a fixed vector, matrix, or function.



Statistical Estimator

Definition

Let $x_1, \ldots, x_n$ be n i.i.d. random variables drawn from some distribution P with parameter θ.

An estimator $\hat{\theta}_n$ of some fixed, unknown parameter θ is some function of $x_1, \ldots, x_n$:

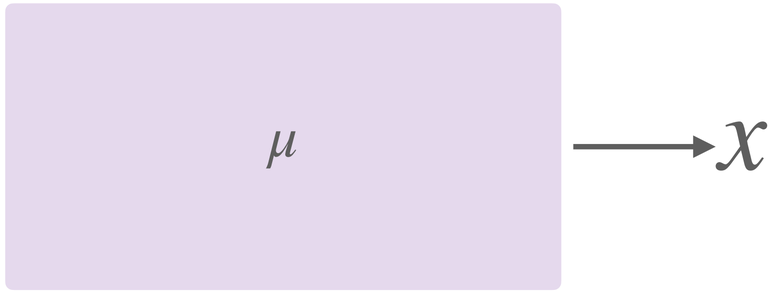
$$\hat{\theta}_n = g(x_1, \ldots, x_n).$$

Importantly: statistical estimators are functions of RVs, so they are *themselves* RVs!

Examples: sample mean, sample variance, histogram, etc.

Statistical Estimator

Example: Mean Estimator for Coins



Example. Let x_i be a random variable denoting the outcome of a single fair coin toss, with $x_i = 0$ for tails and $x_i = 1$ for heads. Clearly, $\mu := \mathbb{E}[x_i] = 1/2$.

Suppose we independently toss n coins, obtaining i.i.d. RVs $x_1, \ldots, x_n$.

Estimand: $\theta = \mu$.

Estimator: $\hat{\theta}_n = \bar{x}_n := \dfrac{1}{n} \sum_{i=1}^{n} x_i$.

Notice that $\hat{\theta}_n$ is random – different realizations of $x_1, \ldots, x_n$ should change $\hat{\theta}_n$.

Bias and Variance

Review: Properties of Estimators

$$\hat{\theta}_n = g(x_1, \ldots, x_n)$$

$\hat{\theta}_n$ itself has a probability distribution, called the sampling distribution.

Some parameters of the sampling distribution we are interested in:

Bias. $\text{Bias}(\hat{\theta}_n) := \mathbb{E}[\hat{\theta}_n] - \theta$

Variance. $\text{Var}(\hat{\theta}_n) := \mathbb{E}[\hat{\theta}_n^2] - \mathbb{E}^2[\hat{\theta}_n]$

Why do we care about variance, if the estimator is unbiased?

$\hat{\theta}(x_1, \ldots, x_n) = x_1$ is an unbiased estimator for the mean of a Gaussian, but is on average further away from μ than the sample mean.

Variance of an average

Motivation for averaging predictors

Denote $D = (x^{(1)}, \ldots, x^{(n)})$ and let $\hat{\theta}(D)$ be an unbiased estimator with variance σ^2.

$$\mathbb{E}[\hat{\theta}] = \theta \text{ and } \mathrm{Var}(\hat{\theta}) = \sigma^2.$$

So far, we have used a single statistic $\hat{\theta} = \hat{\theta}(D)$ to estimate θ.

Consider a new estimator that takes the average of i.i.d. $\hat{\theta}_1, \ldots, \hat{\theta}_k$ where $\hat{\theta}_j = \hat{\theta}(D^{(j)})$.

Assumes we have datasets $D^{(1)}, \ldots, D^{(j)}$.

$$\mathbb{E}\left[\frac{1}{k}\sum_{j=1}^{k}\hat{\theta}_j\right] \text{ (still unbiased) and } \mathrm{Var}\left[\frac{1}{k}\sum_{j=1}^{k}\hat{\theta}_j\right] = \frac{\sigma^2}{k} \text{ (reduced variance!)}$$

Average of Hypotheses

Average Hypothesis Function

Suppose we have B independent training sets, all drawn from the same distribution P.

Our learning algorithm gives us B hypotheses $\hat{h}_1(x), \ldots, \hat{h}_B(x)$.

Define the average hypothesis as:

$$\hat{h}_{avg} := \frac{1}{B} \sum_{b=1}^{B} \hat{h}_b.$$

Average of Hypotheses

Averaging reduces variance

$$\hat{h}_{avg} := \frac{1}{B}\sum_{b=1}^{B} \hat{h}_b.$$

The average prediction for x is:

$$\hat{h}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{h}_b(x)$$ (which has the same expected value as $\hat{h}_b(x)$).

But $\hat{h}_{avg}(x)$ has smaller variance: $\mathrm{Var}(\hat{h}_{avg}(x)) = \frac{1}{B}\mathrm{Var}(\hat{h}_1(x))$!

Problem: In practice, we don't have B independent training sets!

Bootstrap

Bootstrap sample

Problem: In practice, we don't have B independent training sets!

How do we simulate multiple samples when we only have one?

A bootstrap sample from $D_n = (x^{(1)}, \ldots, x^{(n)})$ is a sample of size n drawn *with replacement* from D_n. Some elements of D_n will show up multiple times; some none at all.

> Each $x^{(i)}$ has probability $(1 - 1/n)^n$ of *not* being included in bootstrap sample.
>
> For large n, $(1 - 1/n)^n \approx 1/e \approx .368$.
>
> So we expect ≈63.2% of elements of D_n will show up at least once.

Bootstrap

Bootstrap method

Problem: In practice, we don't have B independent training sets!

A bootstrap method simulates B independent samples from P by taking B bootstrap samples from the sample D_n.

Given original data D_n, compute bootstrap samples $D_n^1, \ldots, D_n^B$.

For each bootstrap sample, compute some function:

$$\phi(D_n^1) \ldots, \phi(D_n^B).$$

Use these values as though $D_n^1, \ldots, D_n^B$ were i.i.d. samples from P.

This ends up being very close to what we'd get with independent samples from P!

Bootstrap

Independent vs. bootstrap samples

Point estimator $\hat{\theta} = \hat{\theta}(D_{100})$ for sample of size $n = 100$ for a synthetic example where we know P.

Histograms based on:

1000 independent samples of size $n = 100$ (left).

1000 bootstrap samples of size $n = 100$ (right).

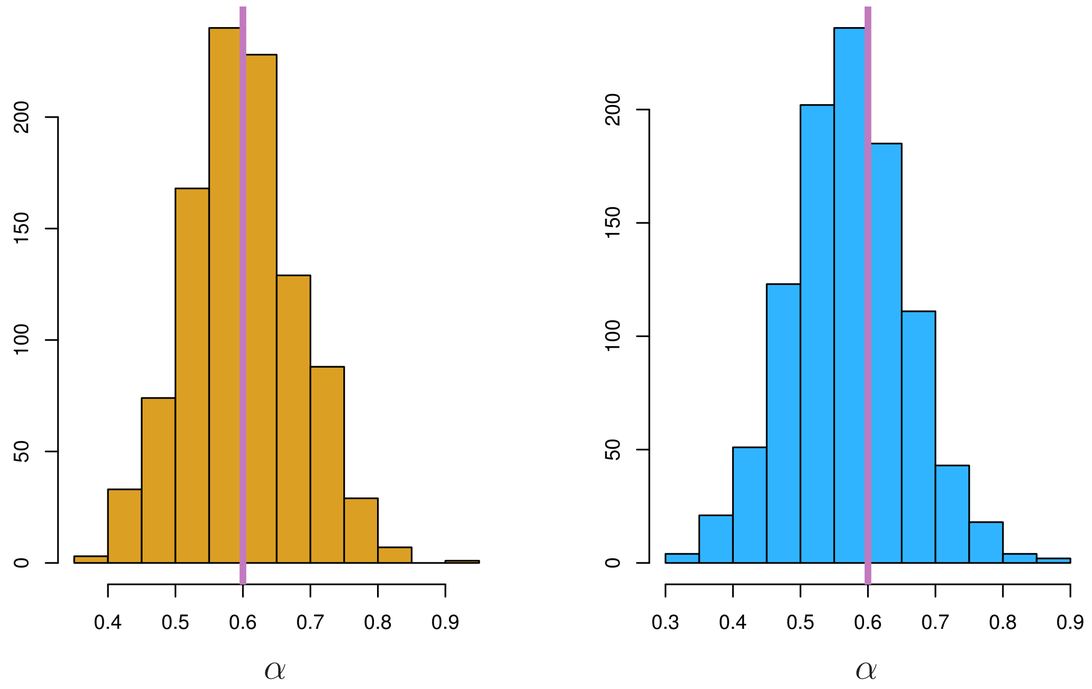


Figure from *Introduction to Statistical Learning* (James, Witten, Hastie, Tibshirani), Ch. 5.

Ensemble Methods

Key Ideas

Ensemble methods combine multiple weaker models into a single, more powerful model.

Averaging i.i.d. estimates reduces variance without changing bias.

We can use bootstrap to simulate multiple data samples and average them.

Parallel ensembling: models are built independently (e.g. bagging).

Sequential ensembling: models are built sequentially (e.g. boosting).

Outline

Decision Tree Basics

Splitting Criterion & Overfitting

Decision Trees vs. Linear Models

Bootstrap Method

Ensemble Method: Bagging

Ensemble Method: Random Forests

Bagging

Bootstrap Aggregation

Draw B bootstrap samples $D^1, \ldots D^B$ from original data D.

Let $\hat{h}_1, \ldots, \hat{h}_B$ be the hypotheses resulting from training on $D^1, \ldots, D^B$ respectively.

The bagged prediction function/hypothesis is some combination of these:

$$\hat{h}_{avg}(x) = \text{combine}(\hat{h}_1(x), \ldots, \hat{h}_B(x)).$$

For regression: combine with **averaging**.

For classification: combine with **majority vote**.

Bagging

Specifically for decision trees

$$\hat{h}_{avg}(x) = \text{combine}(\hat{h}_1(x), \ldots, \hat{h}_B(x)).$$

For regression: combine with **averaging**.

For classification: combine with **majority vote**.

This is a general method for variance reduction, but particularly useful for decision trees.

Increasing the number of trees we use in bagging does *not* lead to overfitting.

Is there a downside compared to having a single decision tree?

Many trees makes the bagged predictor much less interpretable.

Bagging

Example: Classification Trees

$\mathcal{X} = \mathbb{R}^5$ and $\mathcal{Y} = \{-1, +1\}$, with $n = 30$.

Each bootstrap tree is quite different: different splitting variable at root!

High variance: small perturbations of training data lead to high degree of model variability.

Bagging helps most when base learners are relatively unbiased but have high variance (which is the case for decision trees).

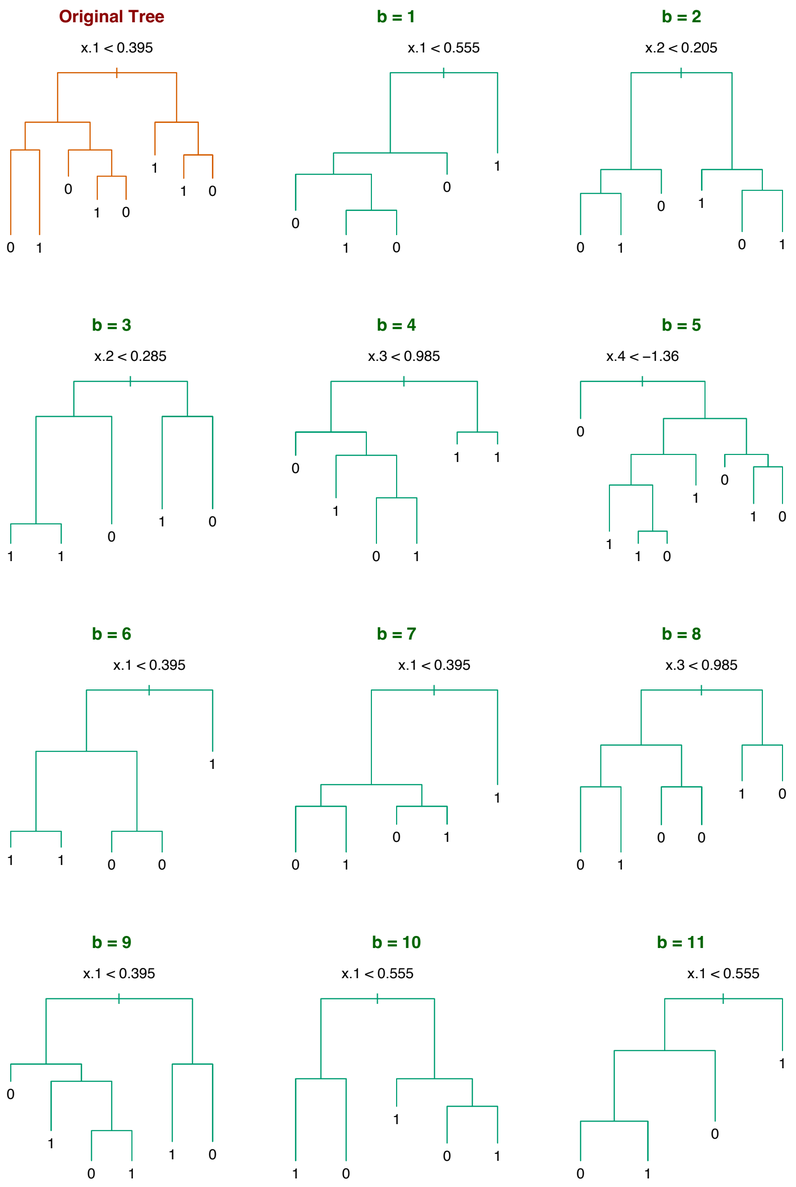


Figure from *Elements of Statistical Learning* (Hastie, Tibshirani, Friedman), Ch. 8.

Outline

Decision Tree Basics

Splitting Criterion & Overfitting

Decision Trees vs. Linear Models

Bootstrap Method

Ensemble Method: Bagging

Ensemble Method: Random Forests

Correlated Hypotheses

Motivation for random forests

$$\mathbb{E}\left[\frac{1}{B}\sum_{b=1}^{B}\hat{\theta}_b\right] \text{ (still unbiased) and } \mathrm{Var}\left[\frac{1}{B}\sum_{b=1}^{B}\hat{\theta}_b\right] = \frac{\sigma^2}{B} \text{ (reduced variance!)}$$

What is $\hat{\theta}$'s are correlated?

For large n, the covariance terms in dominate, limiting the benefits of averaging.

Bootstrap samples are *not* truly independent samples from P (though they are independent samples from the training set).

Need some way to reduce the dependence between the $\hat{\theta}_b$'s!

Random Forests

Main Idea

Use bagged decision trees, but modify the tree-growing procedure to reduce dependence between trees!

Build a collection of trees independently (in parallel), as in bagging.

When constructing each tree node, restrict choice of splitting variable to randomly chosen subset of features of size m.

Prevents situation where all trees dominated by same small number of strong features (and are therefore too similar to each other).

Typically, $m \approx \sqrt{d}$ or can be chosen via cross-validation (if $m = d$ this is just bagging).

Random Forests

Effect of m

Build a collection of trees independently (in parallel), as in bagging.

When constructing each tree node, restrict choice of splitting variable to randomly chosen subset of features of size m.

> Prevents situation where all trees dominated by same small number of strong features (and are therefore too similar to each other).
>
> Typically, $m \approx \sqrt{d}$ or can be chosen via cross-validation (if $m = d$ this is just bagging).

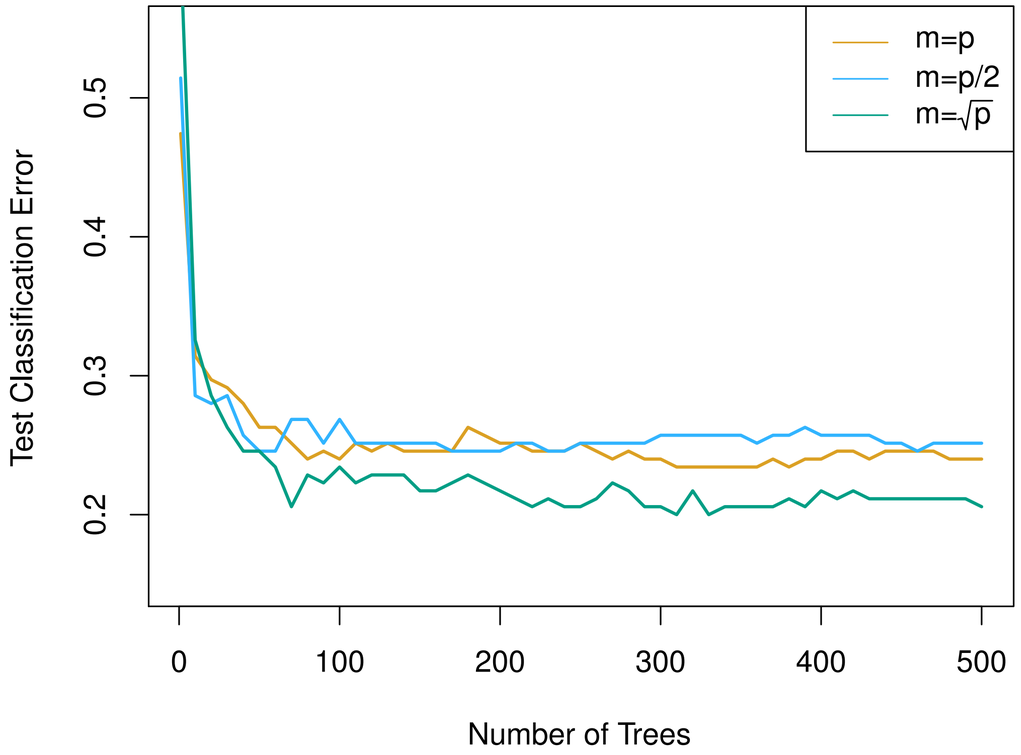


Figure from *Introduction to Statistical Learning* (James, Witten, Hastie, Tibshirani), Ch. 8.

Ensemble Methods

Summary so far

Usual approach: build very deep trees — low bias but **high variance**.

- A change in the training set can greatly impact the output hypothesis.

Ensembling many models may reduce variance.

- Motivation: mean of i.i.d. estimates has smaller variance than a single estimate.

Bootstrap simulates many data samples from one fixed dataset $\Longrightarrow$ bagging.

- But bootstrap samples and the induced models are correlated.

Ensembling works better with a diverse set of hypotheses $\Longrightarrow$ random forest.

- Randomly select subset of features for each decision tree.

Outline

Decision Tree Basics

Splitting Criterion & Overfitting

Decision Trees vs. Linear Models

Bootstrap Method

Ensemble Method: Bagging

Ensemble Method: Random Forests