S&DS 265 / 565

Introductory Machine Learning

Trees and Forests

October 3

Yale

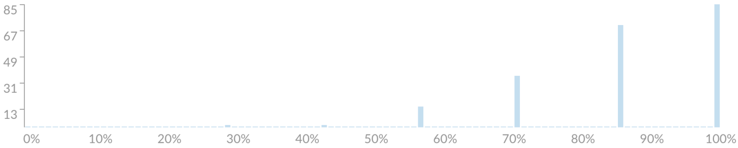
Reminders

- Assn 2 out; due Thursday at midnight
- Quiz 2 last week
- Midterm in class on Tuesday, October 17
- Questions?

Quiz Summary

Section Filter ▾ | Student Analysis | Item Analysis

Average Score	High Score	Low Score	Standard Deviation	Average Time
87%	100%	29%	0.99	13:54



Communication

Important: For any communication regarding assignments, late work, etc. that requires email:

- Email the instructor John Lafferty (john.lafferty@yale.edu) **and** course manager Joanne Chen (haoting.chen@yale.edu)
- State that this is for Introductory Machine Learning (S&DS 265)
- Any email to me alone may go unanswered

Classification and Regression Trees (CART)

Trees provide ways of modeling nonlinear relationships by carving out *rectangular regions* in the feature space.

Classification and Regression Trees (CART)

Trees provide ways of modeling nonlinear relationships by carving out *rectangular regions* in the feature space.

- Response variables can be categorical or quantitative

Classification and Regression Trees (CART)

Trees provide ways of modeling nonlinear relationships by carving out *rectangular regions* in the feature space.

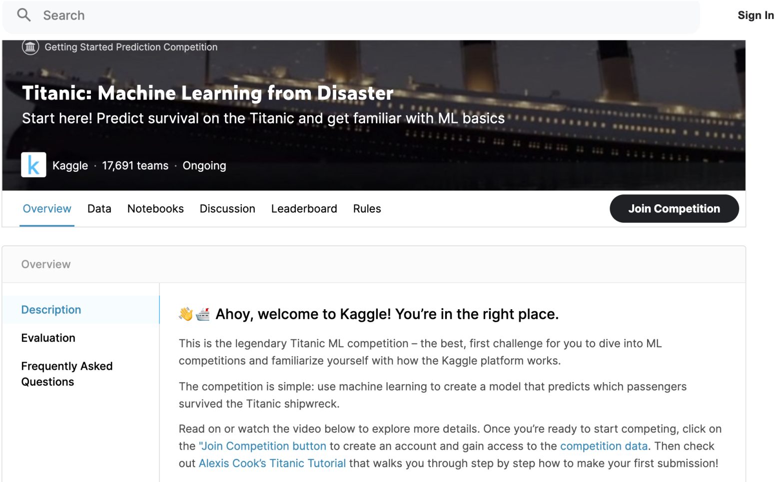
- Response variables can be categorical or quantitative
- Yields a set of interpretable decision rules

Classification and Regression Trees (CART)

Trees provide ways of modeling nonlinear relationships by carving out *rectangular regions* in the feature space.

- Response variables can be categorical or quantitative
- Yields a set of interpretable decision rules
- Predictive ability is mediocre, *but* can be improved by combining multiple trees (resampling, ensemble methods)

Titanic data



Titanic data

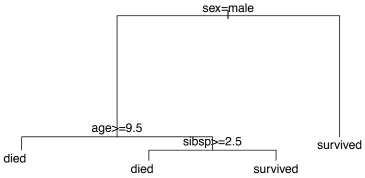
- **Survived**: Outcome of survival (0 = No; 1 = Yes)
- **Pclass**: Socio-economic class (1 = Upper class; 2 = Middle class; 3 = Lower class)
- **Name**: Name of passenger
- **Sex**: Sex of the passenger
- **Age**: Age of the passenger (Some entries contain `NaN`)
- **SibSp**: Number of siblings and spouses of the passenger aboard
- **Parch**: Number of parents and children of the passenger aboard
- **Ticket**: Ticket number of the passenger
- **Fare**: Fare paid by the passenger
- **Cabin** Cabin number of the passenger (Some entries contain `NaN`)
- **Embarked**: Port of embarkation of the passenger (C = Cherbourg; Q = Queenstown; S = Southampton)

Trees

Trees

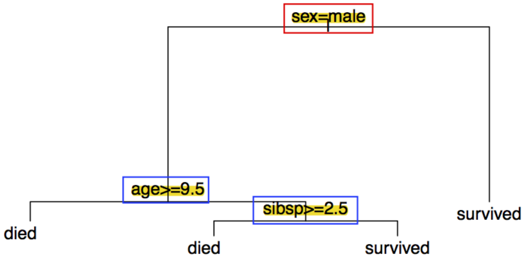
Trees

Modeling Titanic survival:



Tree terminology

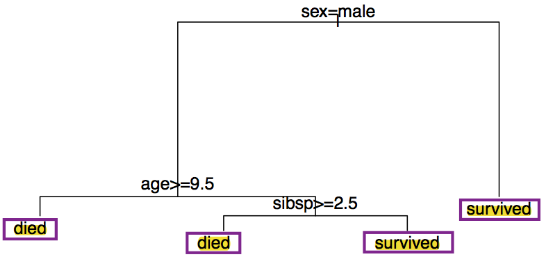
Internal nodes are points where the predictor space is split.



The internal node at the top is the **root** of the tree.

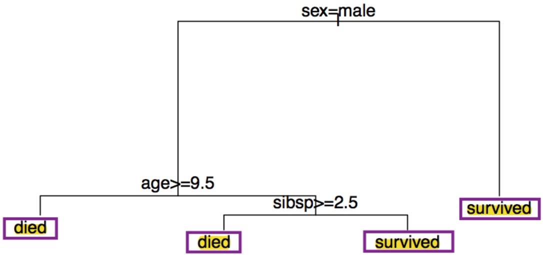
Tree terminology

Terminal nodes (or **leaves**) are the ends of the tree where no further splitting occurs.



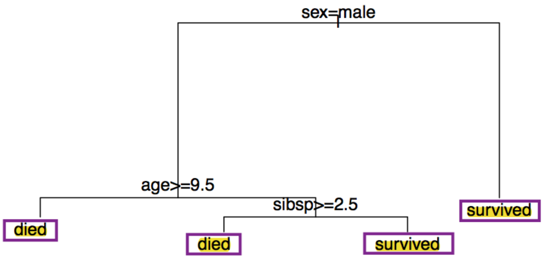
Tree terminology

Terminal nodes (or **leaves**) are the ends of the tree where no further splitting occurs.



Denote these J regions as $R_1, \ldots, R_J$.

Tree terminology



- $R_1 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i \geq 9.5\}$
- $R_2 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i < 9.5 \cap \text{sibsp}_i \geq 2.5\}$
- $R_3 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i < 9.5 \cap \text{sibsp}_i < 2.5\}$
- $R_4 = \{i : \text{sex}_i \neq \text{male}\}$

Let’s go to the Titanic demo

Bias-variance

- Nodes are split by greedily choosing the best question (greatest reduction in error)
- As tree is grown deeper, bias decreases
- But the variance increases
- How to choose the right size of tree?

Stopping criterion

Once we stop, we relabel the terminal nodes to be $R_1, \ldots, R_J$ and compute $\bar{y}_{R_j}$ (means within each region) to serve as $\widehat{y}$ values.

But when do we stop?

Stopping criterion

Once we stop, we relabel the terminal nodes to be $R_1, \ldots, R_J$ and compute $\bar{y}_{R_j}$ (means within each region) to serve as $\widehat{y}$ values.

But when do we stop?

Some possibilities:

- number of observations in a node has reached a minimum

Stopping criterion

Once we stop, we relabel the terminal nodes to be $R_1, \ldots, R_J$ and compute $\bar{y}_{R_j}$ (means within each region) to serve as $\widehat{y}$ values.

But when do we stop?

Some possibilities:

- number of observations in a node has reached a minimum
- depth of tree has reached a maximum

Stopping criterion

Once we stop, we relabel the terminal nodes to be $R_1, \ldots, R_J$ and compute $\bar{y}_{R_j}$ (means within each region) to serve as $\widehat{y}$ values.

But when do we stop?

Some possibilities:

- number of observations in a node has reached a minimum
- depth of tree has reached a maximum
- grow until no further splits can reduce RSS by some amount

Stopping criterion

Once we stop, we relabel the terminal nodes to be $R_1, \ldots, R_J$ and compute $\bar{y}_{R_j}$ (means within each region) to serve as $\widehat{y}$ values.

But when do we stop?

Some possibilities:

- number of observations in a node has reached a minimum
- depth of tree has reached a maximum
- grow until no further splits can reduce RSS by some amount

Stopping criterion

Once we stop, we relabel the terminal nodes to be $R_1, \ldots, R_J$ and compute $\bar{y}_{R_j}$ (means within each region) to serve as $\widehat{y}$ values.

But when do we stop?

Some possibilities:

- number of observations in a node has reached a minimum
- depth of tree has reached a maximum
- grow until no further splits can reduce RSS by some amount

Many options – resulting in tuning parameters that are hard to deal with.

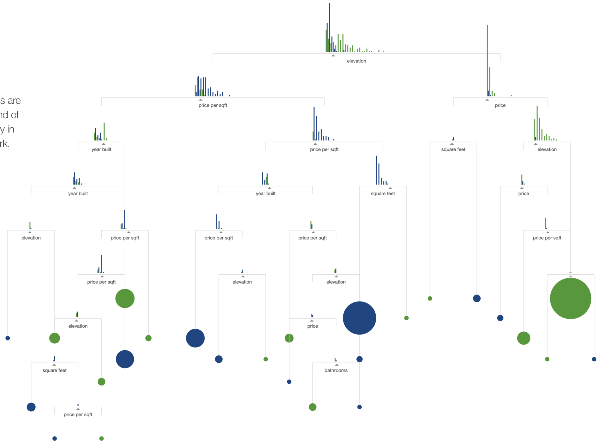
Tree pruning

Another way to get around the overfitting problem is to grow a large tree and then **prune** it back.

Typically, pruning involves looking at subtrees of the fully-grown tree, and comparing how well the subtrees perform.

Tree pruning

You could even continue to add branches until the tree's predictions are **100% accurate**, so that at the end of every branch, the homes are purely in San Francisco or purely in New York.



Tree pruning

How do we prune?

- cross validation
- cost-complexity pruning

Cost-complexity pruning

$$\text{Minimize:} \qquad \text{Loss}(T) + \lambda \{\text{\# of nodes in T}\}$$
$$= \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \widehat{y}_{R_m})^2 + \lambda |T|$$

λ is a tuning parameter that controls for the complexity of the model.

Cost-complexity pruning

$$\text{Minimize:} \qquad \text{Loss}(T) + \lambda \{\text{\# of nodes in T}\}$$
$$= \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \widehat{y}_{R_m})^2 + \lambda |T|$$

λ is a tuning parameter that controls for the complexity of the model.

- $\lambda = 0$ implies the full tree

Cost-complexity pruning

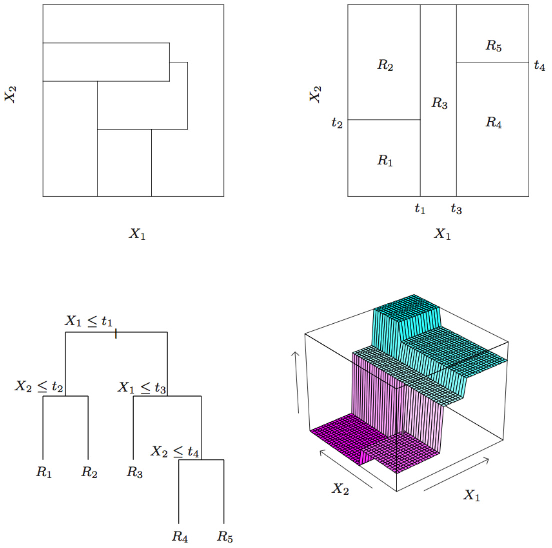
$$\text{Minimize:} \qquad \text{Loss}(T) + \lambda \{\text{\# of nodes in T}\}$$
$$= \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \widehat{y}_{R_m})^2 + \lambda |T|$$

λ is a tuning parameter that controls for the complexity of the model.

- $\lambda = 0$ implies the full tree
- Larger λ implies higher penalty for complexity of model

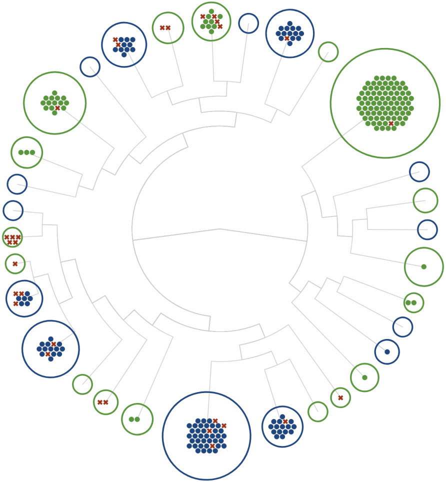
Tree pruning

1. Grow a big tree on a *training set*.
2. Obtain a nested set of subtrees $T_L \subset \cdots \subset T_2 \subset T_1 \subset T_0$ corresponding to a sequence of λ values.
3. Use (leave-one-out or k-fold) cross-validation to identify the subtree/λ that does best.



X_2
X_1
R_5
R_2
t_4
R_3
R_4
t_2
R_1
t_1
t_3
$X_1 \le t_1$
$X_2 \le t_2$
$X_1 \le t_3$
$X_2 \le t_4$
R_1
R_2
R_3
R_4
R_5
X_2
X_1

Beautiful demo `http://www.r2d3.us/`



This demo gives a nice description of bias-variance tradeoff for trees. It's well worth a look!

Trees vs. other methods

Decision trees are similar in spirit to k-nearest neighbors.

Trees vs. other methods

Decision trees are similar in spirit to k-nearest neighbors.

- Both produce simple predictions (averages/maximally occurring) based on “neighborhoods” in the predictor space.

Trees vs. other methods

Decision trees are similar in spirit to k-nearest neighbors.

- Both produce simple predictions (averages/maximally occurring) based on “neighborhoods” in the predictor space.
- Neighborhoods chosen very differently

Trees vs. other methods

Recall that linear regression fits models of the form

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

Trees vs. other methods

Recall that linear regression fits models of the form

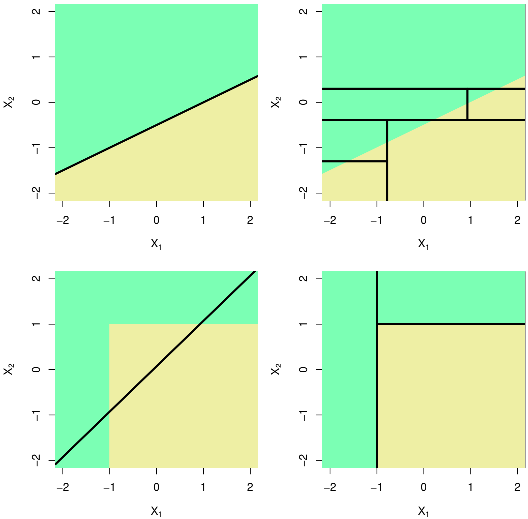
$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

Regression trees are like fitting linear regression models with a bunch of indicators

$$f(X) = \sum_{j=1}^{J} \beta_j \mathbb{1}\left\{X \in R_j\right\}$$

Trees vs. other methods

Are trees always better than linear methods?



Summary so far

- Trees give interpretable, nonlinear prediction rules
- Deep trees have low bias, high variance
- Shallow trees have high bias, low variance
- Deep trees are pruned back using cross-validation to find best bias/variance tradeoff.

Random Forests

- Grow many trees and average their predictions
- Trees are grown deep, to have low bias, but high variance
- To “decorrelate” the trees and reduce variance, each tree is
  - grown on a bootstrap sample of the data
  - grown with random subsets of the predictors at each split
- Tree growing can be done in parallel

Leo Breiman—“Keep it simple”

Random Forests Algorithm

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample Z^* of size n from the training data
   (b) Grow a random-forest tree T_b to the bootstrapped data, recursively repeating following steps, until minimum node size reached:
      i. Select m variables at random from the p variables
      ii. Pick the best variable/split-point among the m
      iii. Split the node into two children nodes
2. Output the ensemble of trees $\{T_b\}_{b=1}^{B}$.

Random Forests Algorithm

To make a prediction at a new point x:

Regression: Average $\widehat{f}^B_{rf}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$

Classification: Majority vote of the individual trees

Out of bag (OOB) prediction

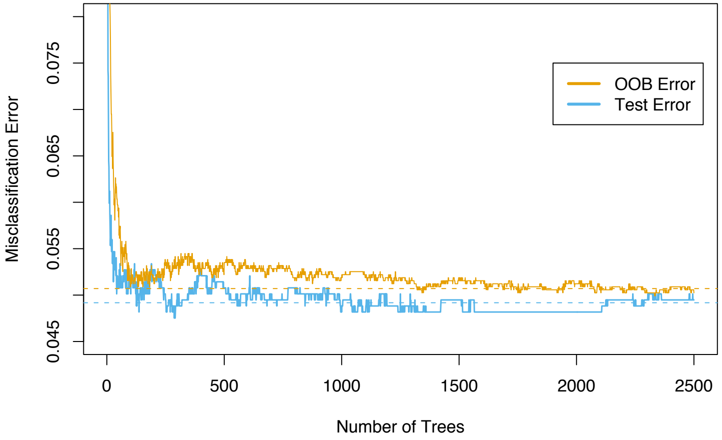
A bootstrap sample, or "bag" of data, is a n data points sampled with replacement from the original set of n data points. Will contain repetitions.

Each tree is grown on such a sample, which contains about $\frac{2}{3}$ of the original data (with repetitions). The remaining $\frac{1}{3}$ can be used as validation data

- For each observation $z_i = (x_i, y_i)$, construct its random forest predictor by averaging only those trees corresponding to bootstrap samples in which z_i did not appear
- Thus, cross-validation can be performed "along the way"

Chance a sample x_i does not appear in a bootstrap sample is $\left(1 - \frac{1}{n}\right)^n \longrightarrow \frac{1}{e} \approx 0.37$

Out of bag (OOB) prediction



Performance on email spam task

Single tree: 8.7%

Random forest: 5.1%

(standard error of the estimates is $\approx 0.6\%$)

Variable importance

Random forests improve upon the predictive ability of trees, but sacrifices the interpretability of the single tree model

Variable importance

Random forests improve upon the predictive ability of trees, but sacrifices the interpretability of the single tree model

A good tool for interpreting a forest is **variable importance**

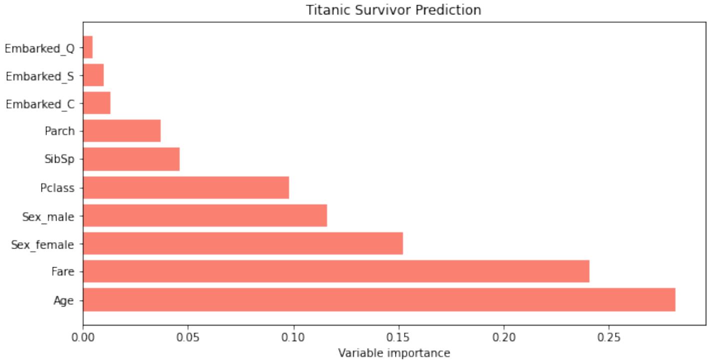
Variable importance

Random forests improve upon the predictive ability of trees, but sacrifices the interpretability of the single tree model

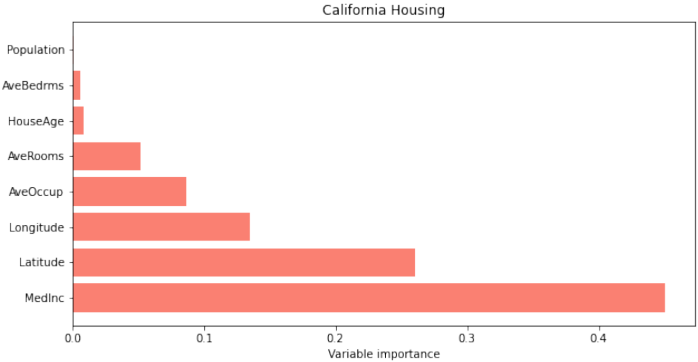
A good tool for interpreting a forest is **variable importance**

Variable importance is the amount that the RSS (or other loss) is reduced due to splits over a given predictor, averaged over all trees in the forest

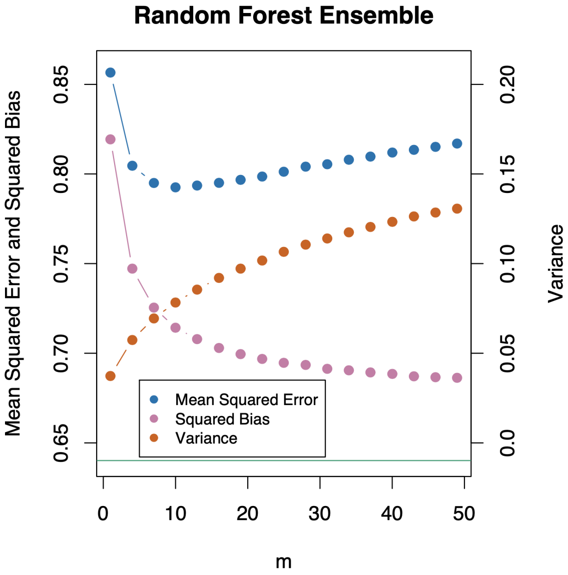
Variable importance



Variable importance



Random forest MSE



Understanding the bias-variance tradeoff for random forests

- Each tree has low (squared) bias, because it is deep
- The bootstrap sample and random subset of questions allowed at a given split result in “diversity” among of the trees
- This diversity translates to decorrelation, and tends to reduce the variance
- As we increase the number *m* in the random subset of questions allowed at each split, the complexity increases—variance goes up, squared bias goes down

Let’s go to the notebook

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:
```
titanic_train = pd.read_csv('https://raw.githubusercontent.com/minsuk-heo/kaggle-titanic/master/input/train.csv')
titanic_test = pd.read_csv('https://raw.githubusercontent.com/minsuk-heo/kaggle-titanic/master/input/test.csv')
titanic_train
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

Summary

- Trees give interpretable, nonlinear prediction rules
- Deep trees have low bias, high variance
- Random forests are a way of combining trees
- Want: Different trees should capture different aspects of the data
- How: Grow each tree on a random (bootstrap) sample of the data and choosing from a random set of questions at each split