Ch 8.2.1, 8.2.2: Bagging and Random Forests

Lecture 17 - CMSE 381

Michigan State University
::
Dept of Computational Mathematics, Science & Engineering

Mon, March 25, 2024

Announcements

Last time:

- 8.1 Decision Trees

This lecture:

- 8.2.1 Bagging
- 8.2.2 Random forest

Announcements:

- Next lecture: Midterm-review

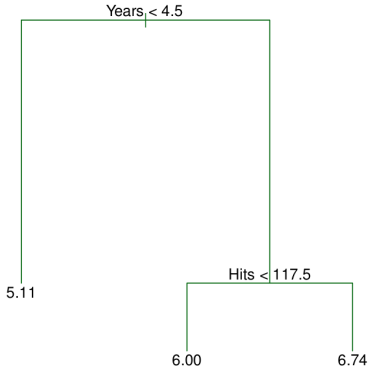
Section 1

Last time

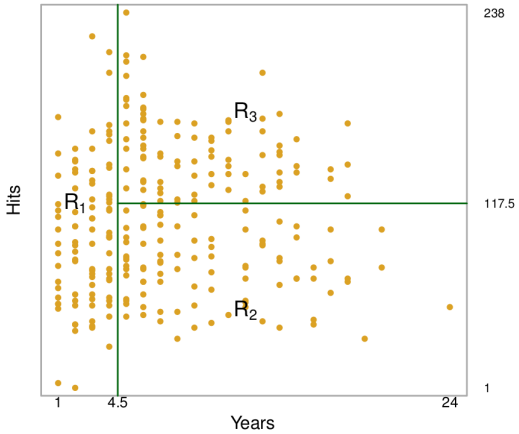
First decision tree example

	Hits	Years	LogSalary
1	81	14	6.163315
2	130	3	6.173786
3	141	11	6.214608
4	87	2	4.516339
5	169	11	6.620073
...	...	...	...
317	127	5	6.551080
318	136	12	6.774224
319	126	6	5.953243
320	144	8	6.866933
321	170	11	6.907755

- Top split assigns observations with Years¡4.5 to left branch
- Return mean response for players with that.
- Predictions:
  - mean log salary is 5.107, so returns $\exp(5.107) = \$165.174$ thousand dollars
  - $5.999 \Rightarrow \$402,834$
  - $6.740 \Rightarrow \$845,346$

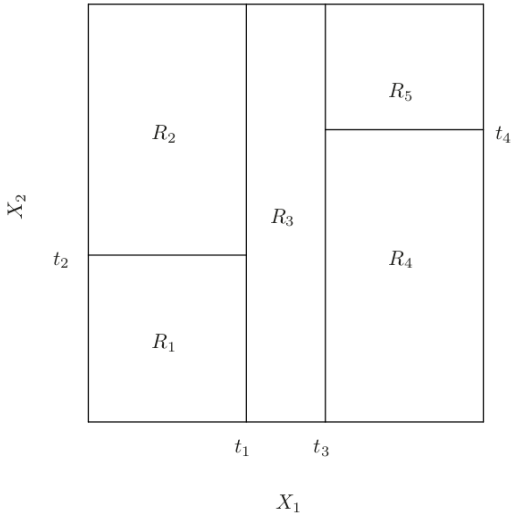


Viewing Regions Defined by Tree



How do we actually get the tree? Two steps

1. We divide the predictor space – that is, the set of possible values for $X_1, X_2, \cdots, X_p$ — into J distinct and non-overlapping regions, $R_1, R_2, \cdots, R_J$.
2. For every observation that falls into the region R_j, we make the same prediction = the mean of the response values for the training observations in R_j.



Recursive binary splitting

Goal:
Find boxes $R_1, \cdots, R_J$ that minimize

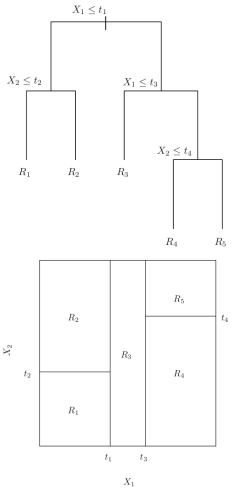
$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

$\hat{y}_{R_j}$ = mean response for training observations in jth box

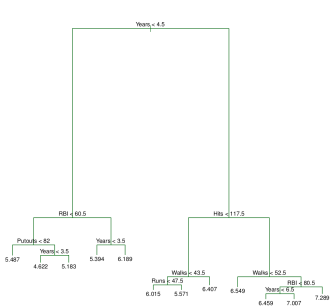
Pick s so that splitting into $\{X \mid X_j < s\}$ and $\{X \mid X_j \geq s\}$ results in largest possible reduction in RSS:

$$R_1(j, s) = \{X \mid X_j < s\}$$
$$R_2(j, s) = \{X \mid X_j \geq s\}$$

$$\sum_{i | x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i | x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

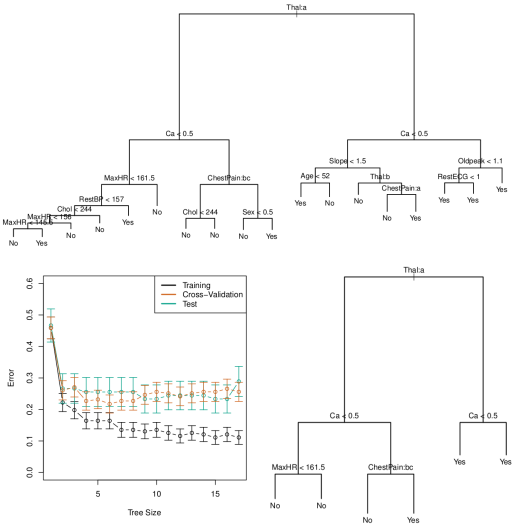


Pruning



- Big trees leave you open to potential overfitting
- One option is maximum deapth
- Another is Weakest link pruning

Classification version



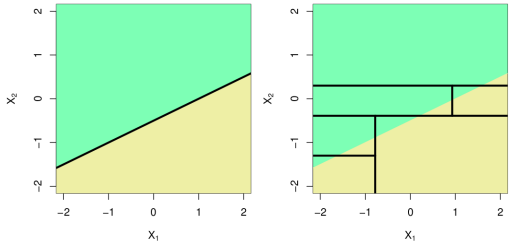
Evaluating the splits:

- $\hat{p}_{mk}$ = proportion of training observations in R_m from the kth class
- Error: $E = 1 - \max_k(\hat{p}_{mk})$
- Gini index:

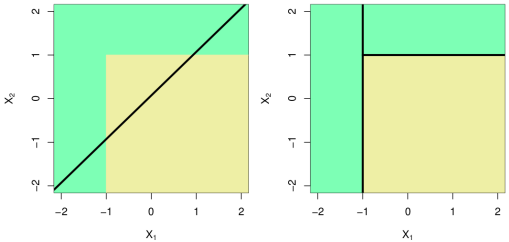
$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Linear models vs trees

- Trying to predict green vs yellow



Obviously linear does better here



Not going to beat this case though

Pros/Cons

Pros:

- Trees are very easy to explain to people. Often easier to explain than linear regression!
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Trees can easily handle qualitative predictors without the need to create dummy variables.

Cons:

- Not as accurate as other methods of classification and regression
- Pruning can be time-consuming
- Not robust: small change in data can cause large change in estimated tree
- Fix..... aggregate many decision trees

Section 2

8.2.1 Bagging

Recall: The bootstrap

Want to do (but can't):
Build separate models from independent training sets, and average resulting predictions:

- $\hat{f}^1(x), \cdots, \hat{f}^B(x)$ for B independent training sets
- Return the average

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

- Decreases variance, but not practical

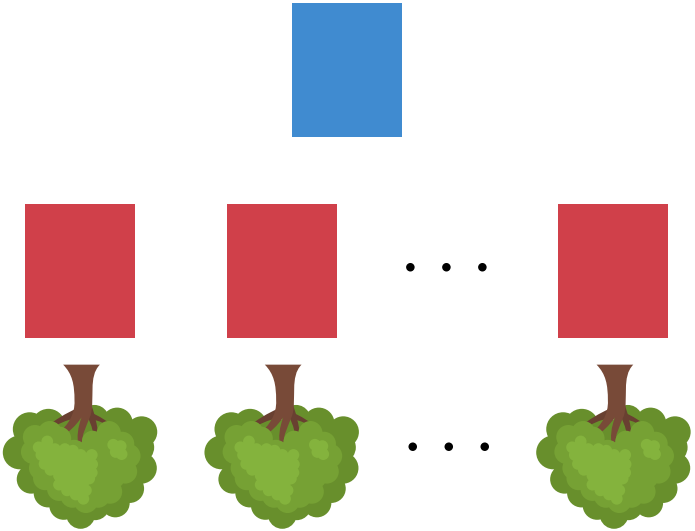
Boostrap modification:

- Work with fixed data set
- Take B samples from this data set (with replacement)
- Train method on bth sample to get $\hat{f}^{*b}(x)$
- Return average of predictions (regression)

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

or majority vote (classification)

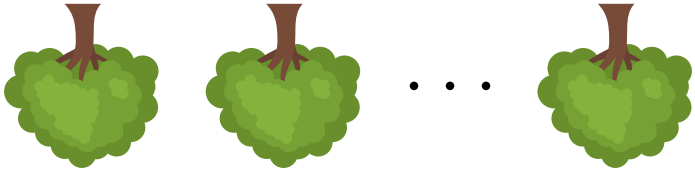
Tree version



- *Blue is original data set* $X : n \times p$
- *Red are the subsampled data sets*
- *Build a tree from each*
- *Get prediction from each, return the average*

Prediction on new data point

New data point goes in, run through each tree, average outputs



Bagging vs Bootstrap

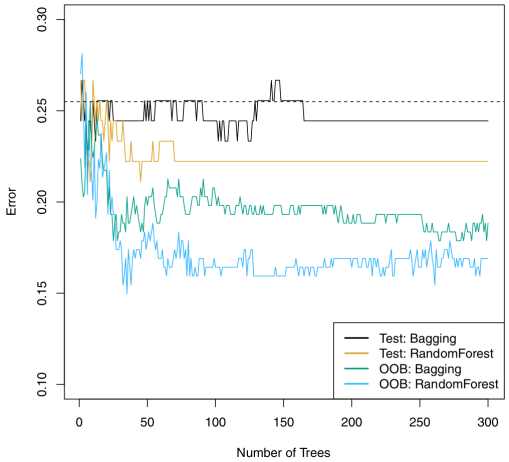
Bootstrap

- Getting subset of data by subsampling
- Better name: Bootstrap sample

Bagging

- Taking the predicted values and averaging them
- Better name: Bootstrap aggregation
- Note: bagging can be done on many different kinds of models. We just happen to be doing it here on trees

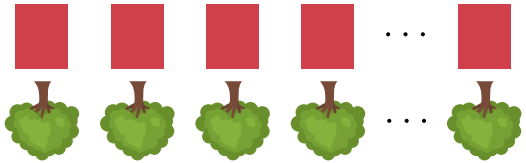
Example: Heart classification data



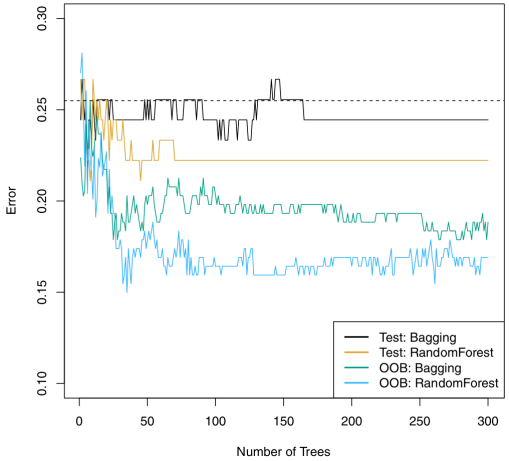
- Example is done for classification
- Ignore everything but black lines
- Dashed is test error from single classification tree
- Solid black is from bagged version
- Note that you just need B big enough for error to stabilize, so about 100 is good

Out of Bag Error Estimation

- On average, bootstrap sample uses about 2/3 of the data *See Ch 5, Ex. 2*
- Remaining observations not used are called *out-of-bag* (OOB) observations
- For each observation, run through all the trees where it wasn't used for building
- Return the average (or majority vote) of those as test prediction



Error using OOB



- Green line has OOB error
- Book says the fact that the OOB value is so much lower is just random chance?

Section 3

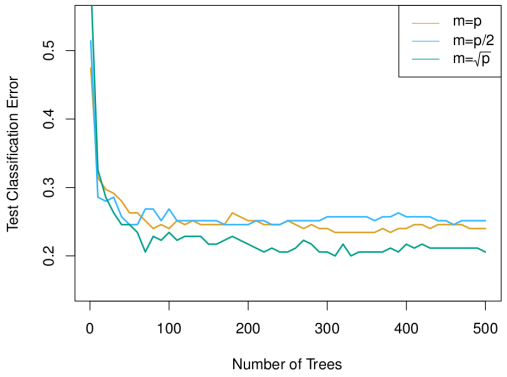
Random Forests

The idea

- Goal is to decorrelate the bagged trees:
  - If there is a strong predictor, the first split of most trees will be the same
  - Most or all trees will be highly correlated
  - Averaging highly correlated quantities doesn't decrease variance as much as uncorrelated

- The random forrest fix:
  - Each time a split is considered, only use a random subset of m the predictors
  - Fresh sample taken every time
  - Typically $m \approx \sqrt{p}$
  - On average, $(p - m)/p$ of splits won't consider strong predictor
  - $m = p$ gives back bagging

Example on gene expression



- Helpful for large p
- Predict one of 14 types of cancer (15 total levels), use 500 genes with most variance
- Random split to train and test set, three values of m
- Error rate of single tree: 45.7%
- Null rate 75.4% (classify each into the dominant overall class, here the healthy class)
- 400 trees enough for good performance
- $m = \sqrt{p}$ small improvement over bagging ($m = p$) in this case

Summary

- Bagging: trees grown independently on random samples. Trees tend to be similar to each other, can result in getting caught in local optima
- Random forest: trees independently on samples, but split is done using random subset of features