Bagging, Boosting and Random Forests

Section 8.2

Cathy Poliak, Ph.D.
cpoliak@central.uh.edu

Department of Mathematics
University of Houston

Decision Trees: Issues

Decision trees have the following disadvantages:

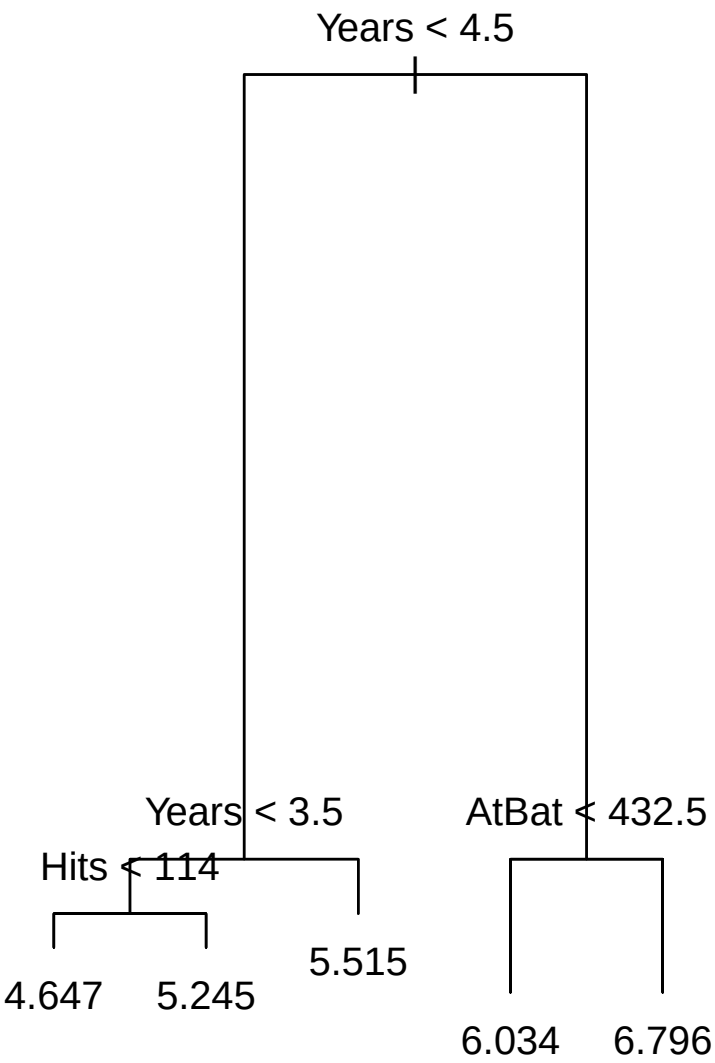
- They struggle with prediction accuracy.
- They suffer from high variance - different subsets of the same data could yield drastically different results.

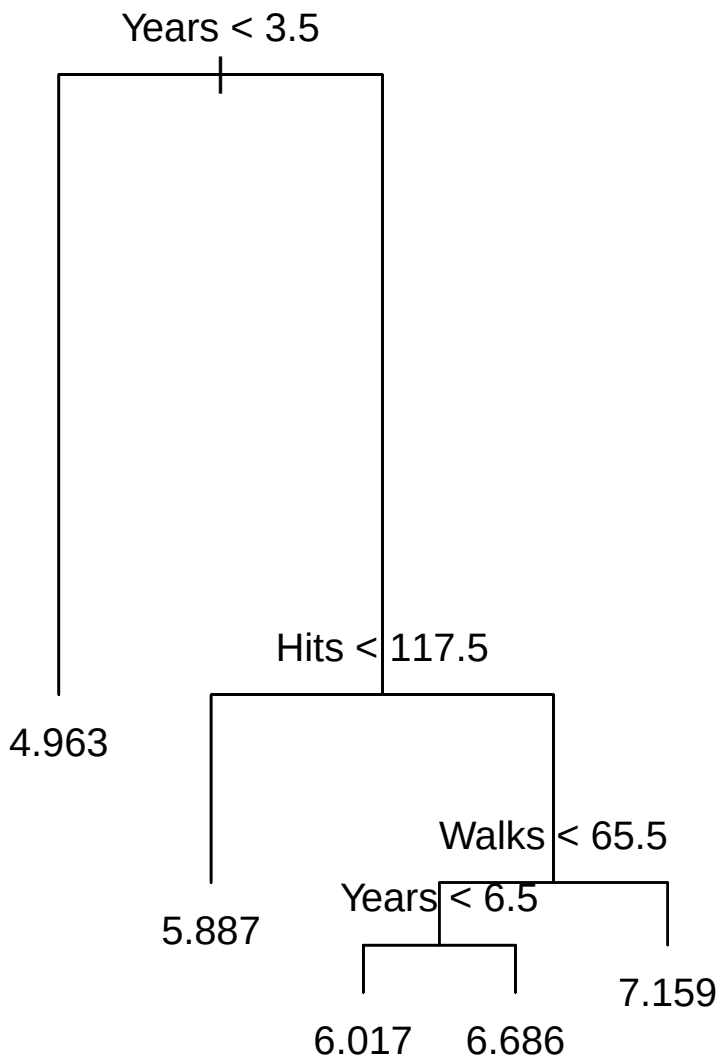
Example Recall the *Hitters* data set. Below we produced pruned regression trees that result from two different training subsets.

```
n = nrow(Hitters)
included.vars = names(Hitters[-c(3,8:15,20)])

par(mfrow = c(1,2))
for (i in 1:2) {
  train = sample(1:n, 0.5*n)
  tree.model = tree(log(Salary) ~.,
                    data = Hitters2[,included.vars],
                    subset = train)
  prune.model = prune.tree(tree.model,best = 5)
  plot(prune.model); text(prune.model,pretty = T)
}
par(mfrow = c(1,1))
```

Difference In Trees





Compared to Linear Models

For comparison, linear models have **low variance**

```
> for (i in 1:2) {
+   train = sample(1:n, .5*n)
+   lm.model = lm(log(Salary) ~.,
+                 data = Hitters2[,included.vars],
+                 subset = train)
+   print(lm.model$coeff)
+ }

(Intercept)         AtBat          Hits          Runs           RBI         Walks
4.3293120988 -0.0040777461  0.0178041401  0.0028982852  0.0014815773  0.0052671305
       Years       PutOuts       Assists        Errors
0.1093491642  0.0003759963  0.0006064904 -0.0058631340


(Intercept)         AtBat          Hits          Runs           RBI         Walks
4.0518530539 -0.0013953663  0.0088648110  0.0050624823  0.0012224423  0.0045370919
       Years       PutOuts       Assists        Errors
0.1032572113  0.0006584508  0.0006051528 -0.0074061893
```

Recall Bootsrap Method

- Resample from the original data - either directly or via a fitted model - to create data sets, from which the variability of the quantities of interest can be assessed with out long-winded and error-prone analytical calculations.
- This approach involves repeating the original data analysis procedure with many replicate sets of data.
- The central goal is to obtain reliable standard errors, confidence intervals, and other measures of uncertainty for a wide range of problems.
- This approach can be applied in simple problems to check the adequacy of standard measures of uncertainty, to relax assumptions, and to give quick approximate solutions.
- The basic idea of bootstrap is to make inference about an estimate (such as the sample mean or sample coefficients $\hat{\beta}_j$) for a population parameter θ (such as the population mean or coefficients β_j) on sample data.

Recall General Approach to Statistical Learning

- Let Y be the response (dependent variable).
- Let $X = (X_1, X_2, \ldots, X_p)$ be p different predictors (independent) variables.
- We assume there is some sort of relationship between X and Y, which can be written in the general form
$$Y = f(X) + \epsilon$$
- Statistical leaning refers to a set of approaches for estimating f.
- A way to reduce the variance and increase the prediction accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set and average the resulting predictions.
- That is we could calculate $\hat{f}^1(x), \hat{f}^2(x), \ldots, \hat{f}^B(x)$ using B separate training sets and average them in order to obtain a single low-variance statistical learning model, given by
$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

Bagging

Since we do not have access to multiple training sets, we can bootstrap, by taking repeated samples from the (single) training data set.

1. We generate B different bootstrapped training data sets.
2. We train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$.
3. Average all the predictions to obtain

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

This is called **bagging**.

Bootstrap Aggregating

Bagging for Decision Trees

- Bagging can improve predictions for many regression methods, it is particularly useful for decision trees.
- To apply bagging to regression trees:
  1. Construct B regression trees using B bootstrapped training sets.
  2. Average the resulting predictions.
- These trees are grown deep, and are not pruned. Thus each individual tree has high variance but low bias.
- Averaging these B trees reduces the variance.
- To apply bagging to classification:
  1. We record the class predicted by each of the B trees
  2. Take a *majority vote*: the overall prediction is the most commonly occurring class among the B predictions.

Bagging in R

Type and run the following in R

```
#install.packages('randomForest')
library(randomForest)
set.seed(1)
p = ncol(Hitters2) - 4 #No. of predictors
B = 100 #No. of bootstrap trees
bag.model.p = randomForest(log(Salary) ~. -NewLeague-League-Division,
                           data = Hitters2,
                           mtry = p)
bag.model.p #To get an outline of bagging results.
> bag.model.p$predicted
-Alan Ashby       -Alvin Davis      -Andre Dawson  -Andres Galarraga
6.111872           6.627010          6.790049           4.676994
```

Lab Questions

Type and run the following in R

```
hitters.test = Hitters2[-train,"Salary"]
yhat = predict(prune.model,newdata = Hitters2[-train,included.vars])
mean((yhat - log(hitters.test))^2)
```

1. What is the name of this result?

   a) Miss classification
   b) Mean of squared residuals
   c) Predicted salary
   d) Predicted hitters

2. Using the bagging method from previous slide what did we get as this value?

   a) 0.195
   b) 0.763
   c) 0.394
   d) 0.060

Example for Classification Tree

We will use the *Heart* data. Recall trees.

```
#Example of Classification Tree
#Using Heart data
Heart = na.omit(Heart); Heart$X = NULL
Heart$AHD = as.factor(Heart$AHD) #needs to be categorical for the tree
Heart$ChestPain = as.factor(Heart$ChestPain) #originally imported as "character"
Heart$Thal = as.factor(Heart$Thal) #originally imported as "character"
train = sample(1:nrow(Heart),nrow(Heart)/2+0.5)
tree.heart = tree(AHD ~ ., Heart,subset =  train)
Heart.test = Heart[-train,]
tree.pred = predict(tree.heart,Heart.test,type = "class")
(conf.matrix = table(tree.pred,Heart.test$AHD))
(conf.matrix[1,2]+conf.matrix[2,1])/sum(conf.matrix)
```

3. What is the name of this result?

a) Test error rate
b) Test correction
c) Predicted values
d) Training values

Bagging

Type and run the following in R.

```
Heart = na.omit(Heart); Heart$X = NULL
n = nrow(Heart); p = ncol(Heart)-1
B = 1000
Heart$AHD = as.factor(Heart$AHD)
bag.model = randomForest(AHD ~., data = Heart,
                         ntree = B,
                         mtry = p,
                         importance = TRUE)
bag.model
```

4. What is the estimate of the error rate?

a) 20%
b) 27%
c) 17%
d) 24%

Estimating the Test Error

- On average, each bagged tree makes use of around two-thirds of the B observations.
- The remaining one-third of the observations not used to fit a given bagged tree are referred to as the **out-of-bag** (OOB) observations.
- We can predict the response for the ith observation using each of the trees in which that observation was OOB.
- We predict the response for the ith observation using each of the trees in which that observation was OOB. This yields around $B/3$ predictions for the ith observation.
- We can then
  - Average these predicted responses if regression is the goal
  - Take a majority vote if classification is the goal
- Then the overall OOB test error can be computed.

Why is there $\frac{1}{3}$ of the observations not used?

- The chance of not being selected in any of the n draws from n samples with replacement is

$$\left(1 - \frac{1}{n}\right)^n$$

$$\lim_{n \to \infty} \left(1 - \frac{1}{n}\right)^n = \exp(-1) \approx \frac{1}{3}$$

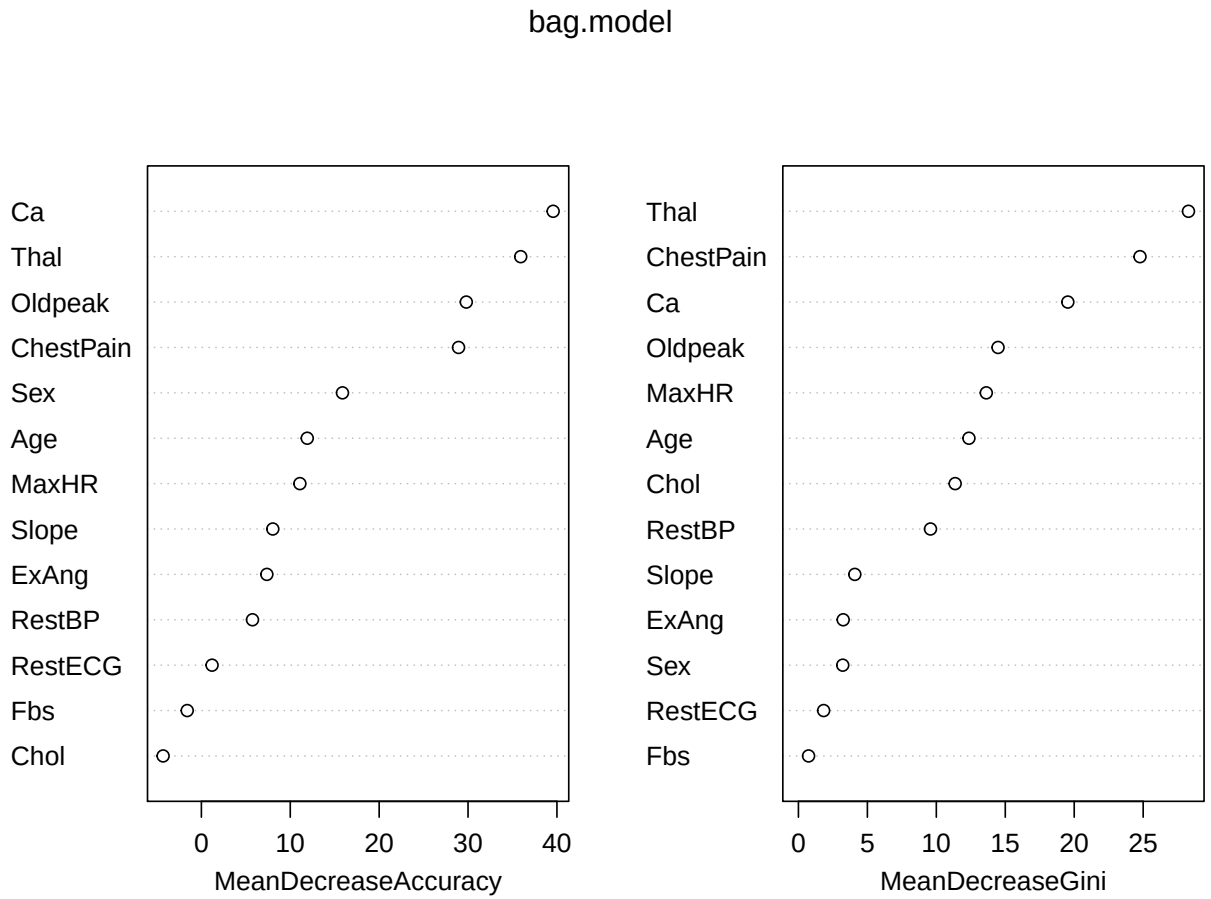
Variable Importance Measures

- Recall advantage of decision trees is the attractive and easily interpreted diagram that results.
- When we bag a large number of trees, the resulting statistical learning procedure is no longer using a single tree and no long clear which variable are most important to the procedure.
- We use **relative variable importance measures**
  - If variable is important, the tree split over that variable causes the **RSS** (or **Gini index** for classification trees) to decrease the most.
  - The measure of variable's importance is the total amount by which RSS (Gini) decreased after each split over that variable.
  - The larger the value the more importance is that variable.

Variable Importance Measures In R

In R we used int the `randomForest()` function importance = TRUE.

The following is the plot of the variable importances for *Heart* data. The variables with the largest mean decrease in Gini index are Thal, Ca, and ChestPain.



Details of Variable Importance Measures

- The first measure is error rate for classification, MSE for regression
  1. For each tree, the prediction error on the out-of-bag portion of the data is recorded.
  2. Then the same is done after permuting each predictor variable.
  3. The difference between the two are then averaged over all trees, and normalized by the standard deviation of the differences.

- The second measure is the total decrease in node impurities from splitting on the variable, averaged over all trees. For classification, the node impurity is measured by the Gini index. For regression, it is measured by residual sum of squares.

Random Forests

- **Random forests** adds a small tweak to further improve on bagging:
  1. Random forests also grow B large un-pruned trees, but
  2. Each time a tree split is considered, it picks a random subset of $m \approx \sqrt{p}$ predictors from the full set of p predictors.

- This leads to **decorreltaing** the bagged trees. Less chance to have same variable dominating each bagged tree (which would have led to high correlation between estimates).

- This stabilizes the variance of the estimate.

- If $m = p$, then this is bagging.

Random Forest in `R`

- Growing a random forest proceeds in exactly the same way as Boosting, except that we use a smaller value of the `mtry` argument.
- By default, `randomForest()` uses $p/3$ variables when building a random forest of regression trees and $\sqrt{p}$ when building a random forest of classification trees.

Random Forests In R

The *mtry* in the `randomForest()` function is changed to fit a random forest. This is the number of predictors to be considered at tree splits.

```
> #Random Forest Model
> AHD.test = Heart[-train,"AHD"]
> X.test = Heart[-train,-14] #Take away the AHD column
> rf.model = randomForest(AHD ~., data = Heart,
+                         subset = train,
+                         xtest = X.test, ytest = AHD.test,
+                         ntree = B,
+                         mtry = sqrt(p),
+                         importance = TRUE)
> rf.model


Type of random forest: classification
                Number of trees: 1000
No. of variables tried at each split: 4

OOB estimate of  error rate: 20.13%
Confusion matrix:
    No Yes class.error
No  74  11   0.1294118
Yes 19  45   0.2968750

Test set error rate: 17.57%
Confusion matrix:
    No Yes class.error
No  67   8   0.1066667
Yes 18  55   0.2465753
```

In R type and run the following

```
varImpPlot(rf.model)
```

5. Which variable is most important to predict heart disease?
   a) *Thal*
   b) *Ca*
   c) *ChestPain*
   d) *Age*

Boosting

- **Boosting** is another approach for improving the predictions resulting from a decision tree.
- Also a general approach that can be applied the many statistical learning methods for regression or classification.
- In *boosting* each tree is grown using information from previously grown trees. Such that each the trees are grown sequentially.

Algorithm for Boosting

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.

2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, R).
   (b) Update $\hat{f}$ by adding a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

   (c) Update the results,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$

3. Output the boosted model,

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

Boosting Approach

- Learns slowly
- We fit a decision to the residuals from the models.
- We then add this new decision tree into the fitted function in order to update the residuals.
- The terminal nodes are determined by the parameter d in the algorithm.
- We slowly improve $\hat{f}$ in areas where it does not perform well.

Three Tuning Parameters

Boosting has three tuning parameters:

1. The number of trees B. Use cross-validation to select B. Otherwise boosting can overfit if B is too large.

2. The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Typically, $\lambda = 0.01$ or 0.001.

3. The number of d splits in each tree. This controls the complexity of the boosted ensemble.

Boosting in R

- Boosting uses the `gmb` package and the `gmb()` function to fit the boosted regression trees.
- Again we will look at the Hitters data set.
- Since this is a regression problem we will use `distribution = "gaussian"`.
- If it was a classification problem, then we would use `distribution = "bernoulli"`.
- Type and run the following in R:

```
install.packages("gbm")
library(gbm)
set.seed(1)
boost.hitters = gbm(log(Salary) ~. -NewLeague-League-Division,
                    data = Hitters2,
                    distribution = "gaussian")
summary(boost.hitters)
```

MSE obtained Boosting

Type and run the following in R:

```
train = sample(1:nrow(Hitters2), .5*nrow(Hitters2))
hitters.test = Hitters2[-train,"Salary"]
yhat.boost = predict(boost.hitters,newdata = Hitters2[-train,],
                     n.trees = 100)
mean((yhat.boost- log(hitters.test))^2)
```

We can use a different value of the shrinking parameter. The default is 0.001, let's use $\lambda = 0.2$.

```
boost.hitters = gbm(log(Salary) ~. -NewLeague-League-Division,
                    data = Hitters2,
                    distribution = "gaussian",
                    shrinkage=0.2,verbose=F)
summary(boost.hitters)
yhat.boost=predict(boost.hitters,newdata=Hitters2[-train,],n.trees=100)
mean((yhat.boost-log(hitters.test))^2)
```