Empowering the Tree

Bagging, Random Forests, and Boosting

Bagging and Random Forests

Goal

Decrease the variance of our estimates while increasing the flexibility of the prediction.

Strategies

Bagging: create many **b**oostrapped data sets, fit a full tree to each, then **agg**regate the predictions.

Random Forests: perform bagging, but for each tree, at each split, only consider $m \leq p$ predictors.

$$m = \sqrt{(p)} \qquad m = p/3$$

Random Forests: tree 1

Let's take a look at what goes into building the first random tree. We're working with the same MLB data set.

```
library(ISLR)
data(Hitters)
```

The first step is to bootstrap a data set.

```
boot_ind <- sample(1:nrow(Hitters),
                   replace = TRUE)
hit_boot <- Hitters[boot_ind, ]
```

A random candidate set

```
m <- sqrt(ncol(hit_boot))
names(hit_boot)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "
##  [6] "Walks"     "Years"     "CAtBat"    "CHits"     "
## [11] "CRuns"     "CRBI"      "CWalks"    "League"    "
## [16] "PutOuts"   "Assists"   "Errors"    "Salary"    "
```

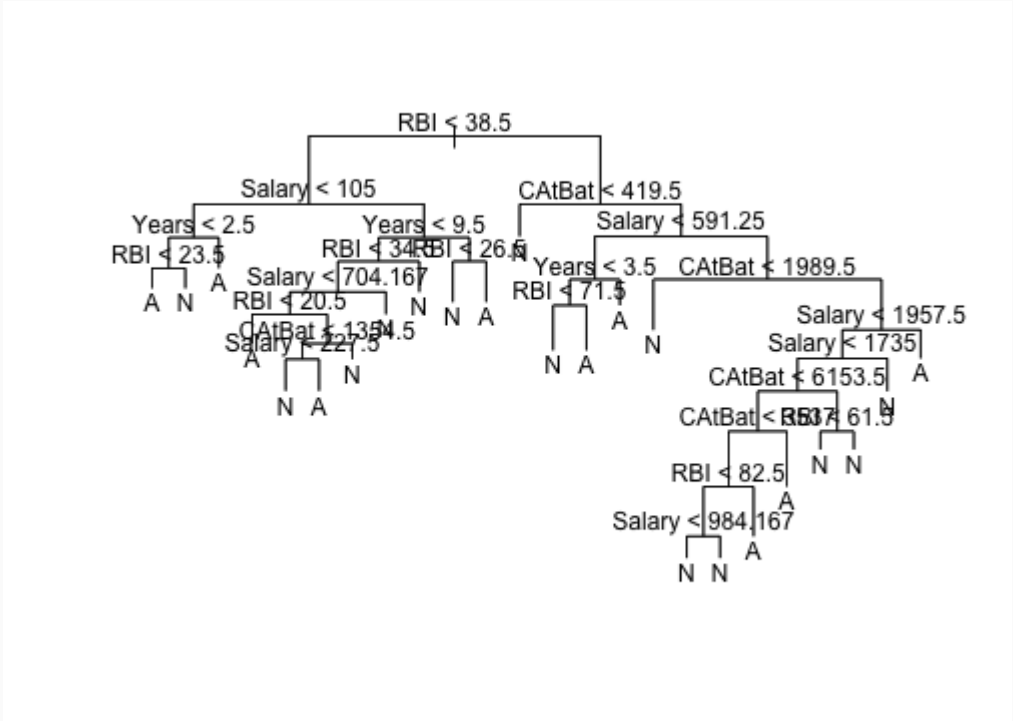
```
rforest_ind <- sample(1:ncol(hit_boot),
                      size = m, replace = FALSE)
rforest_ind
```

```
## [1]  5  8 19  7
```

```
hit_rforest <- hit_boot[ , c(rforest_ind, 14)]
```

The first random split

```
library(tree)
t1 <- tree(League ~ ., data = hit_rforest)
plot(t1)
text(t1, pretty = 0)
```



Subsequent splits

- Each split takes a new random sample (without replacement) of m predictors as candiates.
- This can be done more succinctly with `randomForest()` in `library(randomForest)`.

Random Forests: tree 2

Create a new bootstrap a data set.

```
boot_ind <- sample(1:nrow(Hitters),
                   replace = TRUE)
hit_boot <- Hitters[boot_ind, ]
```

A random candidate set

```
m <- sqrt(ncol(hit_boot))
names(hit_boot)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "
##  [6] "Walks"     "Years"     "CAtBat"    "CHits"     "
## [11] "CRuns"     "CRBI"      "CWalks"    "League"    "
## [16] "PutOuts"   "Assists"   "Errors"    "Salary"    "
```

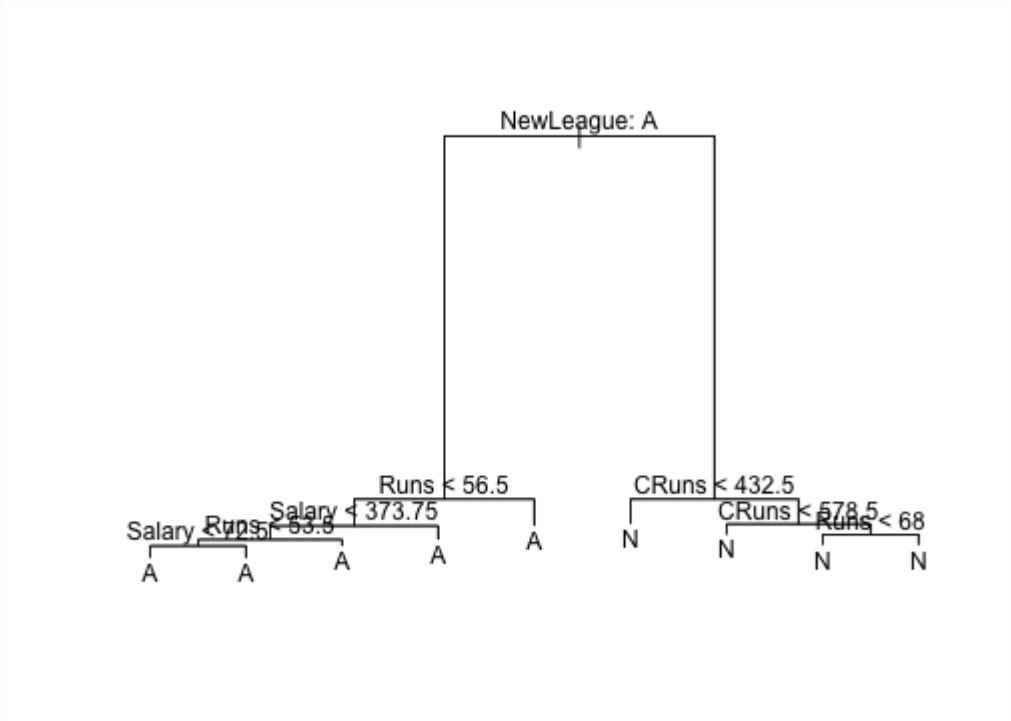
```
rforest_ind <- sample(1:ncol(hit_boot),
                      size = m, replace = FALSE)
rforest_ind
```

```
## [1] 19 11 20  4
```

```
hit_rforest <- hit_boot[ , c(rforest_ind, 14)]
```

The first random split

```
library(tree)
t1 <- tree(League ~ ., data = hit_rforest)
plot(t1)
text(t1, pretty = 0)
```



Making a final prediction

- Predictor values are plugged into each random tree to make a prediction.
- Final prediction is made by majority vote (classification) or averaging (regression).

boardwork

Consider the ant and the bee

On his own, the bee is far too weak to move the bee ...

Consider the ant and the bee

... but combine many complementary weak components, and you get something that's very powerful.

Boosting

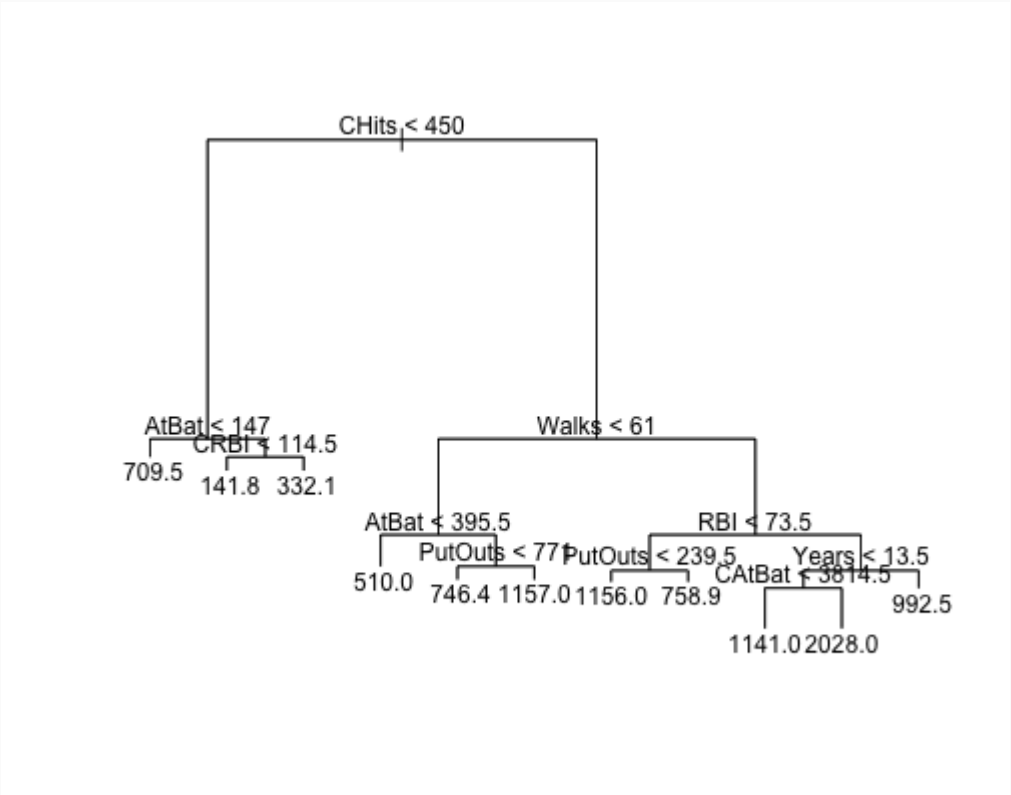
A very general statistical technique that builds up a sequence of weak complementary learners into a boosted learner that is strong.

Note: there is no bootstrapping needed.

Growing a weak tree

Back to the regression setting...

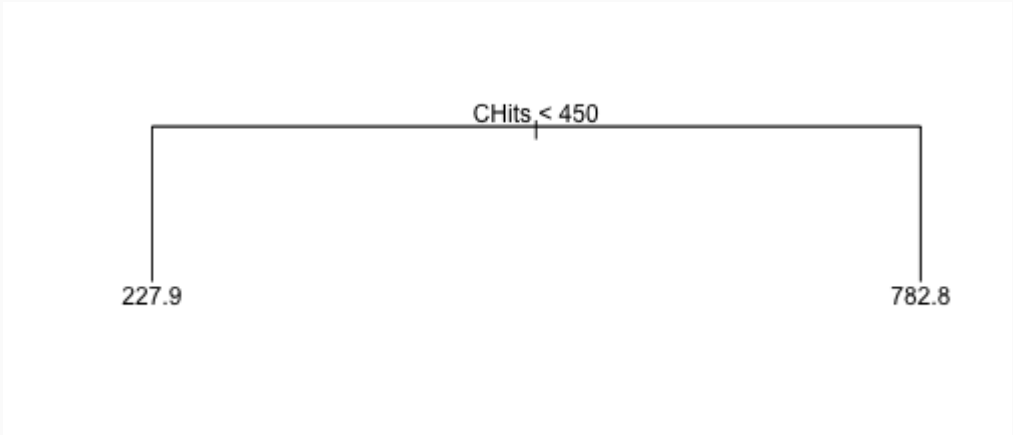
```
t2 <- tree(Salary ~ ., data = Hitters)
```



Growing a weak tree

Let's say we want each weak tree to contain only one split $d = 1$.

```
weak1 <- prune.tree(t2, best = 2)
```



Finding a complement

How can we figure out where this weak learner failed to explain the data well?

Residuals

Residuals

$r_i = y_i - \hat{y}_i$

```
Hitters$Salary[5:8]
```

```
## [1]  91.5 750.0  70.0 100.0
```

```
predict(weak1, newdata = Hitters)[5:8]
```

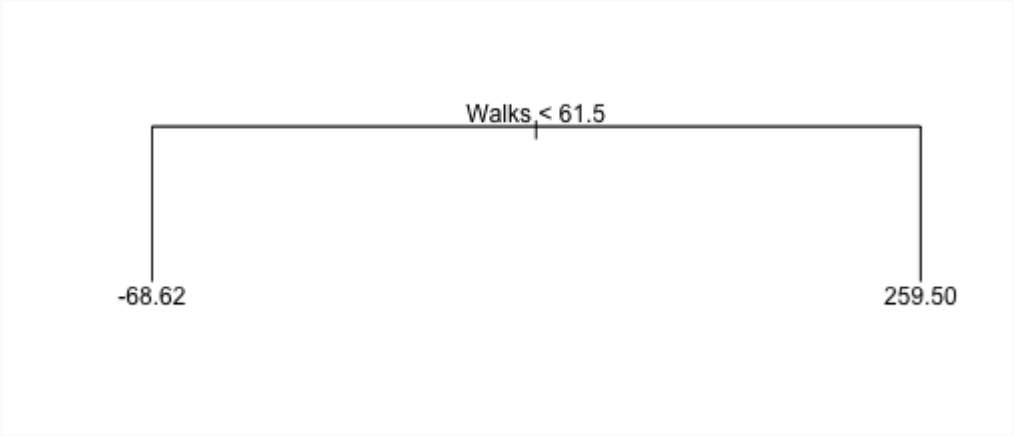
```
## -Andres Galarraga  -Alfredo Griffin         -Al Newman
##          227.8547           782.8048          227.8547
```

```
Hitters$Salary[5:8] -
  predict(weak1, newdata = Hitters)[5:8]
```

```
## -Andres Galarraga  -Alfredo Griffin         -Al Newman
##         -136.35470          -32.80484         -157.85470
```

Growing a new weak tree

```
r1 <- Hitters
r1$Salary <- Hitters$Salary -
  predict(weak1, newdata = Hitters)
t3 <- tree(Salary ~ ., data = r1)
weak2 <- prune.tree(t3, best = 2)
```



And so on...

- Continue process of fitting simple trees (weak learners) to the residuals from the previous model.
- Stop after you've made a total of B trees.
- Final boosted prediction:

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

well, that's close, but we can be a bit more flexible.