More on Tree Based Methodss

ISLR Chapter 8

April 24, 2017

Outline

Ways to improving trees through a multiple trees in some ensemble:

- Bagging

Outline

Ways to improving trees through a multiple trees in some ensemble:

- Bagging
- Boosting

Outline

Ways to improving trees through a multiple trees in some ensemble:

- Bagging
- Boosting
- Random Forests

Outline

Ways to improving trees through a multiple trees in some ensemble:

- Bagging
- Boosting
- Random Forests
- BART (Bayesian Additive Regression Trees)

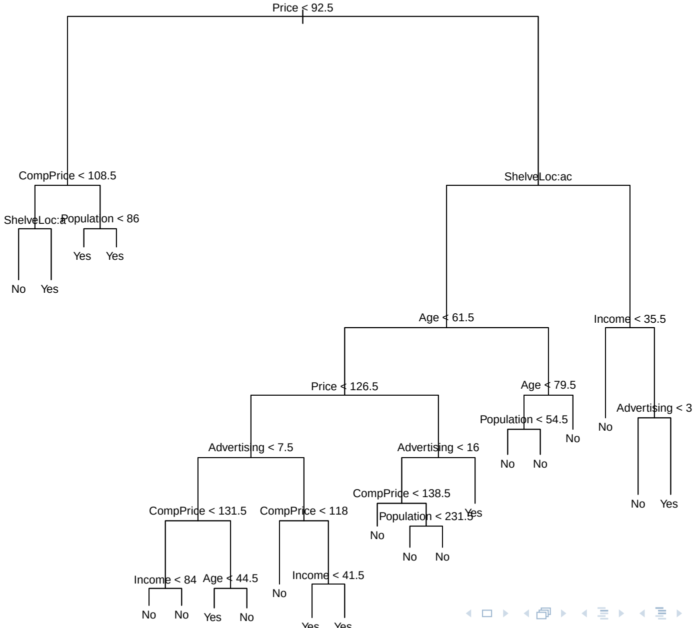
Outline

Ways to improving trees through a multiple trees in some ensemble:

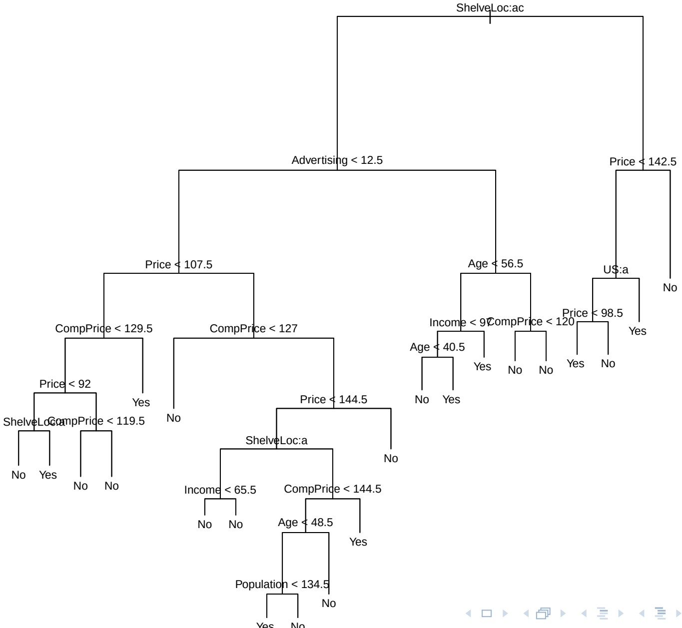
- Bagging
- Boosting
- Random Forests
- BART (Bayesian Additive Regression Trees)

Combining trees will yield improved prediction accuracy, but with loss of interpretability.

Tree with Random Split of Data



Tree with another Random Split of Data



Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)

Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)
- Reduce variability by averaging over multiple training sets!

Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)
- Reduce variability by averaging over multiple training sets!
- do not have access to multiple training sets so create them via bootstrap samples (sample of size n with replacement)

Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)
- Reduce variability by averaging over multiple training sets!
- do not have access to multiple training sets so create them via bootstrap samples (sample of size n with replacement)
  - Generate B bootstrap sample of observations from the single training data

Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)
- Reduce variability by averaging over multiple training sets!
- do not have access to multiple training sets so create them via bootstrap samples (sample of size n with replacement)
  - Generate B bootstrap sample of observations from the single training data
  - Calculate predictions for the bth sample $\hat{f}^b(x)$

Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)
- Reduce variability by averaging over multiple training sets!
- do not have access to multiple training sets so create them via bootstrap samples (sample of size n with replacement)
  - Generate B bootstrap sample of observations from the single training data
  - Calculate predictions for the bth sample $\hat{f}^b(x)$
  - Bagging (Bootstrap Aggregation) estimate is

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

- Trees are grown deep so little bias (although could prune)

Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)
- Reduce variability by averaging over multiple training sets!
- do not have access to multiple training sets so create them via bootstrap samples (sample of size n with replacement)
  - Generate B bootstrap sample of observations from the single training data
  - Calculate predictions for the bth sample $\hat{f}^b(x)$
  - Bagging (Bootstrap Aggregation) estimate is

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

- Trees are grown deep so little bias (although could prune)
- Reduce variance by averaging many trees across the bootstrap samples

Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)
- Reduce variability by averaging over multiple training sets!
- do not have access to multiple training sets so create them via bootstrap samples (sample of size n with replacement)
  - Generate B bootstrap sample of observations from the single training data
  - Calculate predictions for the bth sample $\hat{f}^b(x)$
  - Bagging (Bootstrap Aggregation) estimate is

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

- Trees are grown deep so little bias (although could prune)
- Reduce variance by averaging many trees across the bootstrap samples

Random Forests

Closely related to Bagging, but attempts to de-correlate the trees used in the average

Random Forests

Closely related to Bagging, but attempts to de-correlate the trees used in the average

- take B bootstrap samples

Random Forests

Closely related to Bagging, but attempts to de-correlate the trees used in the average

- take B bootstrap samples
- Each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the set of p predictors.

Random Forests

Closely related to Bagging, but attempts to de-correlate the trees used in the average

- take B bootstrap samples
- Each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the set of p predictors.
- a new sample is taken at each split

Random Forests

Closely related to Bagging, but attempts to de-correlate the trees used in the average

- take B bootstrap samples
- Each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the set of p predictors.
- a new sample is taken at each split
- Recommended m around $\sqrt{p}$

Random Forests

Closely related to Bagging, but attempts to de-correlate the trees used in the average

- take B bootstrap samples
- Each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the set of p predictors.
- a new sample is taken at each split
- Recommended m around $\sqrt{p}$
- Random Forests with $m = p$ is Bagging

Random Forests

Closely related to Bagging, but attempts to de-correlate the trees used in the average

- take B bootstrap samples
- Each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the set of p predictors.
- a new sample is taken at each split
- Recommended m around $\sqrt{p}$
- Random Forests with $m = p$ is Bagging
- Predictions are based on average over the bootstrap samples

Random Forests

Closely related to Bagging, but attempts to de-correlate the trees used in the average

- take B bootstrap samples
- Each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the set of p predictors.
- a new sample is taken at each split
- Recommended m around $\sqrt{p}$
- Random Forests with $m = p$ is Bagging
- Predictions are based on average over the bootstrap samples

Bagging Example

```
suppressMessages(library(randomForest))
bag.carseats = randomForest(High ~ . -Sales,
                            data=Carseats, subset = train,
                            mtry=10, importance =TRUE)
#  mtry = 10 (the number of predictors)
yhat.bag = predict(bag.carseats,newdata = Carseats.test)
tab = table(yhat.bag,Carseats.test$High)
tab

##
## yhat.bag No Yes
##      No   96  16
##      Yes  20  68

CE.tab["RandomForests", 1]= (tab[1,1] + tab[2,2])/sum(tab)

CE.tab["RandomForests",]
```

RandomForests

```
 rf.carseats = randomForest(High ~ . -Sales,data=Carseats,
                            subset =train,
                            mtry=3, importance =TRUE)
 yhat.rf= predict(rf.carseats ,newdata =Carseats.test)
tab = table(yhat.rf,Carseats.test$High)

 tab

##
## yhat.rf No Yes
##     No   99  23
##     Yes  17  61

CE.tab["RandomForests", 1]= (tab[1,1] + tab[2,2])/sum(tab)
# (101+63)/200
CE.tab["RandomForests",]

## [1] 0.8
```

Variable Importance Measures in Bagging & RandomForests

- For Regression Trees use the total reduction in Sum of Squares Error due to splits with that variable averaged over all trees

Variable Importance Measures in Bagging & RandomForests

- For Regression Trees use the total reduction in Sum of Squares Error due to splits with that variable averaged over all trees
- For Classification Trees use the total reduction in Gini Index due to splits of the that variable averaged over all trees. Gini Index is defined as

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

where K is the number of classes for region m or use the reduction in deviance

Variable Importance Measures in Bagging & RandomForests

- For Regression Trees use the total reduction in Sum of Squares Error due to splits with that variable averaged over all trees
- For Classification Trees use the total reduction in Gini Index due to splits of the that variable averaged over all trees. Gini Index is defined as

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

  where K is the number of classes for region m or use the reduction in deviance
- Out of Bag Prediction Error

Variable Importance Measures in Bagging & RandomForests

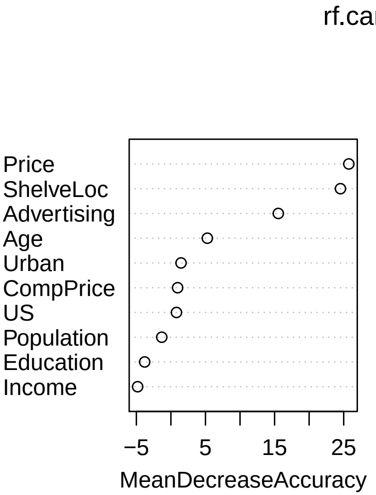
- For Regression Trees use the total reduction in Sum of Squares Error due to splits with that variable averaged over all trees
- For Classification Trees use the total reduction in Gini Index due to splits of the that variable averaged over all trees. Gini Index is defined as

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

  where K is the number of classes for region m or use the reduction in deviance
- Out of Bag Prediction Error
- May be normalized by dividing by the maximum

Variable Importance Measures in RandomForests

```r
varImpPlot(rf.carseats)
```



Boosting

Algorithm 8.2 *Boosting for Regression Trees*

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 in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$
   (c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$
3. Output the boosted model,
   $$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

Boosting

- Mean is a sum of B trees

Boosting

- Mean is a sum of B trees
- Boosting can over-fit if B is too large

Boosting

- Mean is a sum of B trees
- Boosting can over-fit if B is too large
- The number of splits, d, in each tree controls the complexity of the boosted ensemble; $d = 1$ corresponds to models of stumps

Boosting

- Mean is a sum of B trees
- Boosting can over-fit if B is too large
- The number of splits, d, in each tree controls the complexity of the boosted ensemble; $d = 1$ corresponds to models of stumps
- d is the interaction depth; d splits can involve d variables. (default in `gbm` package is 4)

Boosting

- Mean is a sum of B trees
- Boosting can over-fit if B is too large
- The number of splits, d, in each tree controls the complexity of the boosted ensemble; $d = 1$ corresponds to models of stumps
- d is the interaction depth; d splits can involve d variables. (default in `gbm` package is 4)

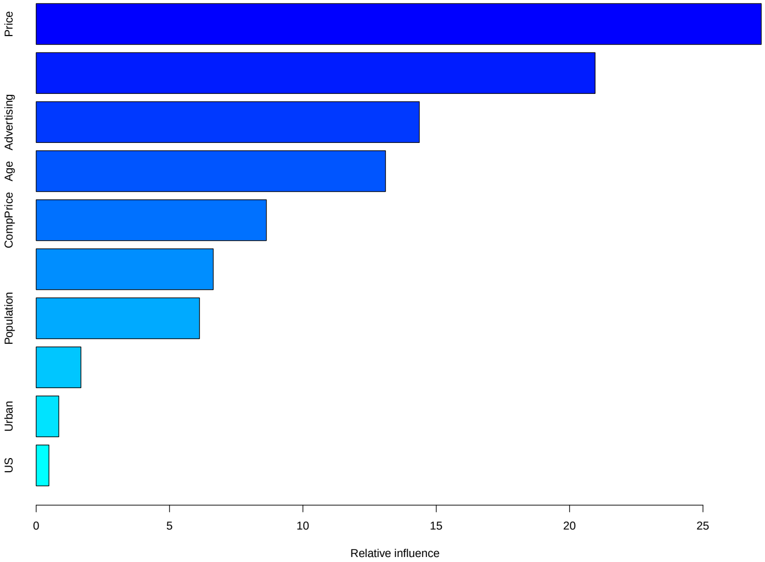
Boosting Example

```
suppressMessages(library(gbm))
boost.car =gbm(I(as.numeric(High)-1) ~ . -Sales,
               data=Carseats[train ,],
               distribution="bernoulli",
               n.trees =5000, interaction.depth =4)
yhat.boost = ifelse(predict(boost.car,
                      newdata=Carseats.test,
                      n.trees=5000,
                      type="response") > .5, 1, 0)
tab =table(yhat.boost, Carseats.test$High)
tab = table(yhat.boost,Carseats.test$High)
tab

##
## yhat.boost  No Yes
##          0 100  16
##          1  16  68
```

Variable Importance: Boosting

```
summary(boost.car, plotit=FALSE)

##                     var    rel.inf
## Price             Price 27.1860994
## ShelveLoc     ShelveLoc 20.9558458
## Advertising Advertising 14.3652543
## Age                 Age 13.0972300
## CompPrice     CompPrice  8.6307712
## Income           Income  6.6391956
## Population   Population  6.1235720
## Education     Education  1.6769694
## Urban             Urban  0.8472994
## US                   US  0.4777628
```

Variable Importance: summary(boost.car)



Bayesian Additive Regression Trees

Gaussian Model: Single Tree model

$$Y = g(x, T, M) + \epsilon$$

Where T is a tree and $M = (\mu_1, \ldots \mu_b)^T$ is the vector of means at the terminal nodes of the tree given by T

Bayesian Additive Regression Trees

Gaussian Model: Single Tree model

$$Y = g(x, T, M) + \epsilon$$

Where T is a tree and $M = (\mu_1, \ldots \mu_b)^T$ is the vector of means at the terminal nodes of the tree given by T

BART represents the mean function as

$$Y = \sum_j g(x, T_j, M_j) + \epsilon$$

as a sum of trees.

Bayesian Additive Regression Trees

Gaussian Model: Single Tree model

$$Y = g(x, T, M) + \epsilon$$

Where T is a tree and $M = (\mu_1, \ldots \mu_b)^T$ is the vector of means at the terminal nodes of the tree given by T

BART represents the mean function as

$$Y = \sum_j g(x, T_j, M_j) + \epsilon$$

as a sum of trees.
Priors control the complexity of each tree; back-fitting as in boosting.

Bayesian Additive Regression Trees

Gaussian Model: Single Tree model

$$Y = g(x, T, M) + \epsilon$$

Where T is a tree and $M = (\mu_1, \dots \mu_b)^T$ is the vector of means at the terminal nodes of the tree given by T

BART represents the mean function as

$$Y = \sum_j g(x, T_j, M_j) + \epsilon$$

as a sum of trees.
Priors control the complexity of each tree; back-fitting as in boosting.

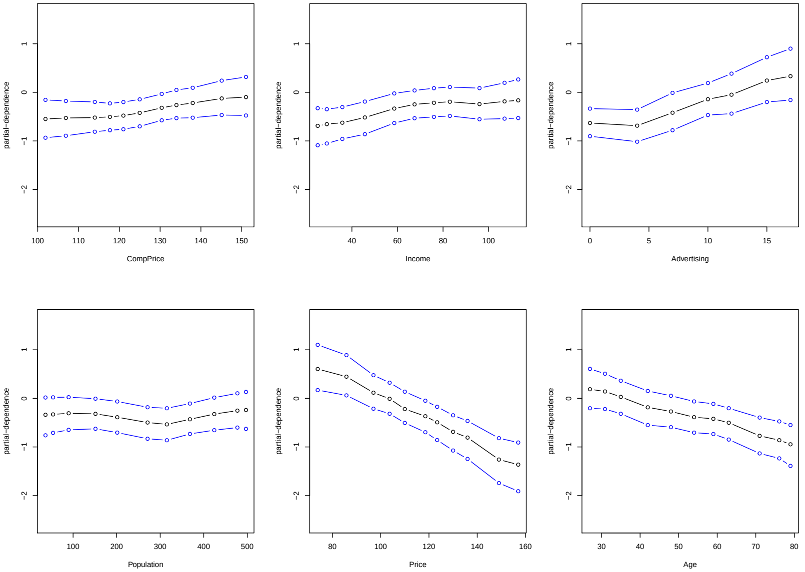
BART: Bayesian Additive Regression Trees

```
library(BayesTree)

## Error in library(BayesTree):  there is no package
called 'BayesTree'

set.seed(42)
bart.carseats = bart(x.train=Carseats[train,-c(1,12)],
                     y.train=Carseats$High[train],
                     x.test=Carseats[-train,-c(1,12)],
                     verbose=FALSE    )

## Error in eval(expr, envir, enclos):  could not
find function "bart"

pihat.bart = apply(pnorm(bart.carseats$yhat.test), 2, mean)

## Error in pnorm(bart.carseats$yhat.test):  object
'bart.carseats' not found
```

Uncertainty in Mean Function

```
plot(bart.carseats)

## Error in plot(bart.carseats):  object
'bart.carseats' not found
```

Partial Dependendence (computationally intensive!)



BartMachine

```r
suppressMessages(library(bartMachine))
set.seed(42)
bart.carseats = bartMachine(X=Carseats[train,-c(1,12)],
                            y=Carseats$High[train],
                            verb=FALSE, serialize=TRUE)

## serializing in order to be saved for future R sessions.

yhat.bart = predict(bart.carseats,
                    Carseats.test[, -c(1,12)],
                    type="class")
table(yhat.bart, Carseats.test$High)

##
## yhat.bart  No Yes
##       No  103  20
##       Yes  13  64
```

Variable Importance

```
## Error in library(bartMachine): there is no
package called 'bartMachine'
```

```
investigate_var_importance(bart.carseats)

## Error in eval(expr, envir, enclos): could not
find function "investigate_var_importance"
```

GLMs and Bayesian Variable Selection

```
set.seed(42); library(BAS)
bas.carseats = bas.glm(High ~ . - Sales, data=Carseats,
                       subset=train, family=binomial(),
                       method="MCMC",  n.models=10000,
                       betaprior=bic.prior(n=200))
yhat = predict(bas.carseats, newdata=Carseats[-train,])
tab = table(ifelse(yhat$fit > .5, 1, 0), Carseats.test$Hig
CE.tab["BMA", 1]= (tab[1,1] + tab[2,2])/sum(tab)

CE.tab["BMA",]

## [1] 0.905

#(107 + 74)/200
```

LASSO Variable Selection

```
set.seed(42); library(glmnet)

## Loading required package:  Matrix
## Loading required package:  foreach
## Loaded glmnet 2.0-5

glmnet.carseats = cv.glmnet(
  x=model.matrix(High ~ . - Sales, data=Carseats[train,]),
  y=(as.numeric(Carseats$High[train]) - 1), family="binomia
yhat = predict(glmnet.carseats,
               newx=model.matrix(High ~ . - Sales, data=Ca
tab = table(ifelse(yhat > .0, 1, 0), Carseats.test$High)
CE.tab["LASSO", 1]= (tab[1,1] + tab[2,2])/sum(tab)

CE.tab["LASSO",]

## [1] 0.89
```

Summary

- Trees are simple to understand, but have high variability

Summary

- Trees are simple to understand, but have high variability
- Bagging (Averaging) reduces variability

Summary

- Trees are simple to understand, but have high variability
- Bagging (Averaging) reduces variability
- Random Forests adds additional constraints to average trees that are different (reduced correlation leads to reduction in variance).

Summary

- Trees are simple to understand, but have high variability
- Bagging (Averaging) reduces variability
- Random Forests adds additional constraints to average trees that are different (reduced correlation leads to reduction in variance).
- Boosting builds a mean function that uses multiple trees where the growth takes into account the previous trees. Smaller trees can be used (larger trees can overfit)

Summary

- Trees are simple to understand, but have high variability
- Bagging (Averaging) reduces variability
- Random Forests adds additional constraints to average trees that are different (reduced correlation leads to reduction in variance).
- Boosting builds a mean function that uses multiple trees where the growth takes into account the previous trees. Smaller trees can be used (larger trees can overfit)
- BART is similar to Boosting in that the mean function is a sum of trees, but uses a Bayesian approach to control complexity. Trees can be of different sizes and the number of trees can be large without overfitting

Summary

- Trees are simple to understand, but have high variability
- Bagging (Averaging) reduces variability
- Random Forests adds additional constraints to average trees that are different (reduced correlation leads to reduction in variance).
- Boosting builds a mean function that uses multiple trees where the growth takes into account the previous trees. Smaller trees can be used (larger trees can overfit)
- BART is similar to Boosting in that the mean function is a sum of trees, but uses a Bayesian approach to control complexity. Trees can be of different sizes and the number of trees can be large without overfitting
- Plots may suggest that a (generalized) linear model may be appropriate!

Summary

- Trees are simple to understand, but have high variability
- Bagging (Averaging) reduces variability
- Random Forests adds additional constraints to average trees that are different (reduced correlation leads to reduction in variance).
- Boosting builds a mean function that uses multiple trees where the growth takes into account the previous trees. Smaller trees can be used (larger trees can overfit)
- BART is similar to Boosting in that the mean function is a sum of trees, but uses a Bayesian approach to control complexity. Trees can be of different sizes and the number of trees can be large without overfitting
- Plots may suggest that a (generalized) linear model may be appropriate!

Interpretability? For which methods can you explain how changes in a predictor change the response?

```r
save(CE.tab, file="OoS.accuracy")
```