Part 10. Tree-based models - part 2

Dr. Nguyen Quang Huy

July 22, 2020

Tree-based model

- Tree-based model is a low-bias and high-variance method
- In this section, we introduce variance-reduction techniques in tree-based model including *bagging*, *random forests*, and *boosting*.
- These approaches reduce the variance of tree-based models by producing multiple trees which are then combined to yield a single prediction.
- The improvements in prediction accuracy of these approaches are at the expense of some loss in model interpretation.
- The topic to be discuss in this section
    - Bootstrapping
    - Bagging and random forest
    - Boosting

Boostrapping

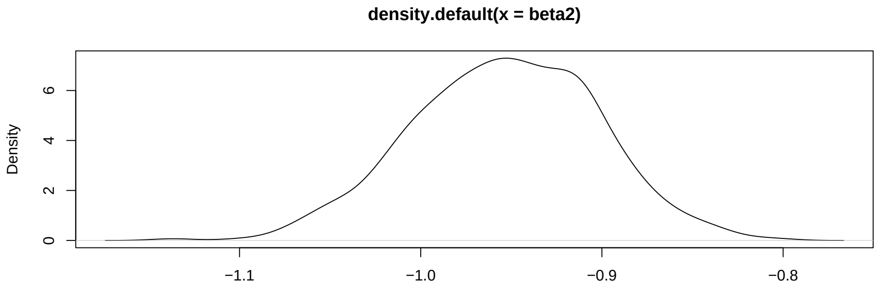
Bootstrap is a widely applicable and extremely powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator.

```
summary(lm(medv~lstat,data=Boston))
```

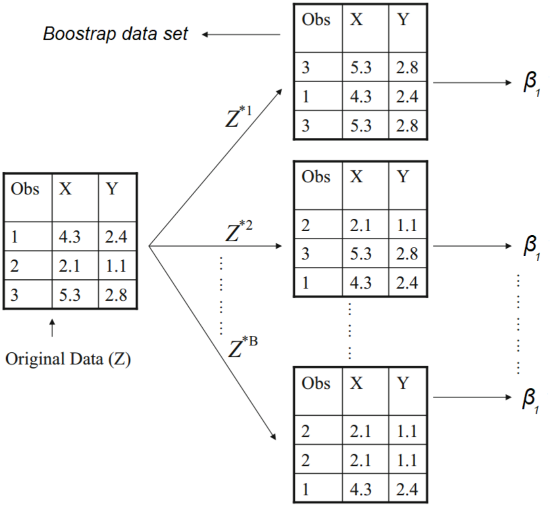
```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
```

Boostrapping

```r
N<-1000
beta2<-rep(0,N)
for (i in 1:N){
  x<-sample(1:nrow(Boston),nrow(Boston),replace=TRUE)
  beta2[i]<-lm(medv~lstat,data=Boston[x,])$coef[2]
}
plot(density(beta2))
```



Boostrapping



Original Data (Z):

Obs	X	Y
1	4.3	2.4
2	2.1	1.1
3	5.3	2.8

Z^{*1}:

Obs	X	Y
3	5.3	2.8
1	4.3	2.4
3	5.3	2.8

Z^{*2}:

Obs	X	Y
2	2.1	1.1
3	5.3	2.8
1	4.3	2.4

Z^{*B}:

Obs	X	Y
2	2.1	1.1
2	2.1	1.1
1	4.3	2.4

Boostrap is useful in situations in which it is hard to compute the standard deviation of an estimator

Bagging

- Given a set of n independent observations $Z_1, \cdots, Z_n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations is given by σ^2/n.
- If we have n independent trainning data sets, we could calculate $\hat{f}_1, \cdots, \hat{f}_n$ and average them in order to obtain a single low-variance model.
- It is not practical because we do not have access to multiple training sets. Instead, we can bootstrap, by taking repeated samples from the (single) training data set.
- Suppose that B different bootstrapped training data sets, we can train our method on the b^{th} bootstrapped training set in order to get $f^b(x)$, and finally average all the predictions

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

Bagging

To apply bagging to regression trees,

- Construct B regression trees using B bootstrapped training sets.
- These trees are grown deep and are not pruned.
- Each individual tree has high variance, but low bias.
- Averaging these B trees to obtain prediction with lower variance than single tree

To apply bagging to classification trees, we can record the class predicted by each of the B trees, and take a majority vote: the overall prediction is the most commonly occurring majority class among the B predictions.

Bagging and Boston data set

```
library(randomForest)
dat<-Boston
standardize<-function(x){x<-(x-mean(x,na.rm=TRUE))/sd(x,na.rm
for (col in names(dat)){
  if((col!="medv")&class(dat[,col]) %in% c("integer","numeric'
    dat[,col]<-standardize(dat[,col])
  }
}
set.seed(1)
test_index<-createDataPartition(dat$medv, times = 1, p = 0.2,l
train<-dat[-test_index,]
test<-dat[test_index,]
```

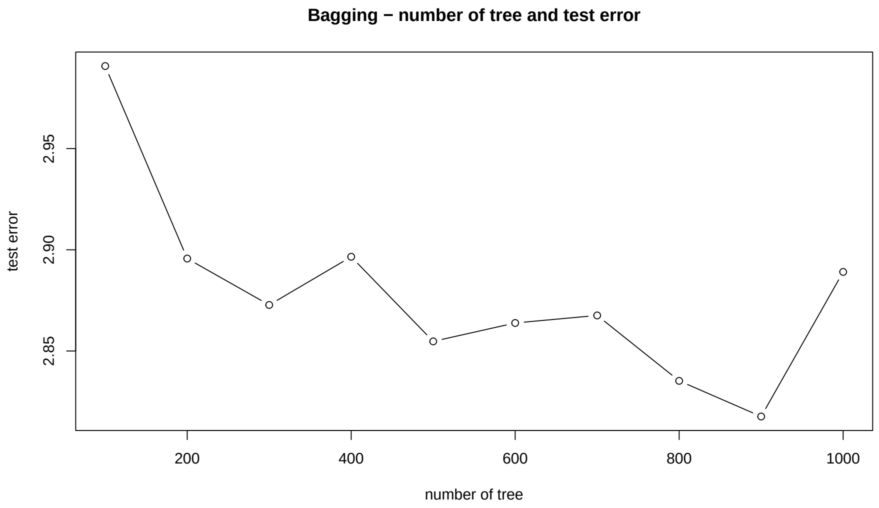
Bagging and Boston data set

```
bag.fit<-randomForest(medv~.,
                      data=train,
                      mtry=ncol(train)-1, #number of predicto
                      importance=TRUE,
                      ntree=200) # number off tree
bag.pred<-predict(bag.fit,newdata=test)
sqrt(mean((bag.pred-test$medv)^2))
```

```
## [1] 2.942067
```

Increase the number of tree and observe the test error ?

Bagging and Boston data set



Bagging

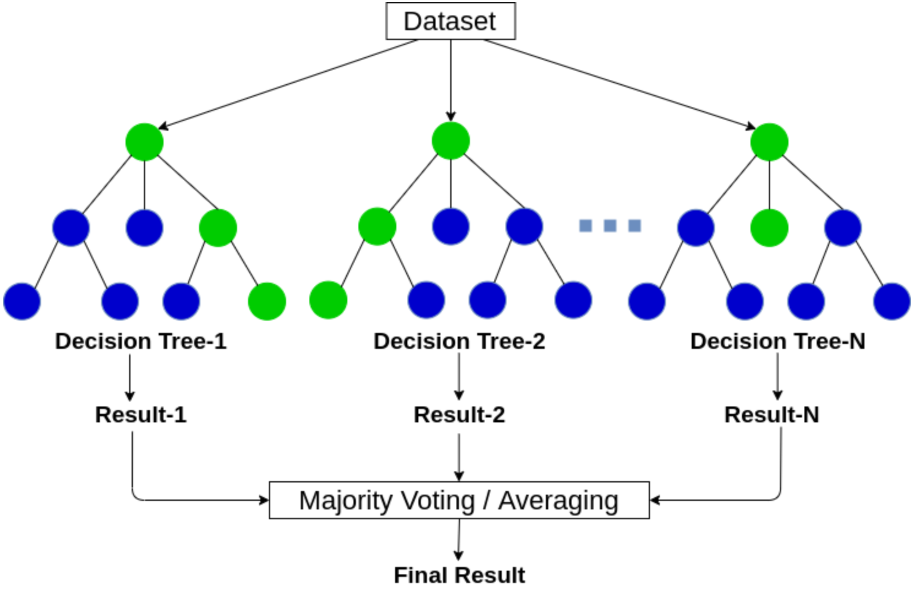
- Bagging results in improved accuracy over prediction using a single tree.
- Unfortunately, it can be difficult to interpret the resulting model.
- Bagging improves prediction accuracy at the expense of interpretability.
- We can obtain an overall summary of the importance of each predictor using the RSS (for bagging regression trees) or the Entropy index (for bagging classification trees)

```
##           %IncMSE IncNodePurity
## crim    21.346694     999.88054
## zn       2.203785      30.26316
## indus   12.217769     231.67418
## chas     0.418869      39.55540
## nox     32.784004     838.88029
## rm      68.741107   14131.42916
```

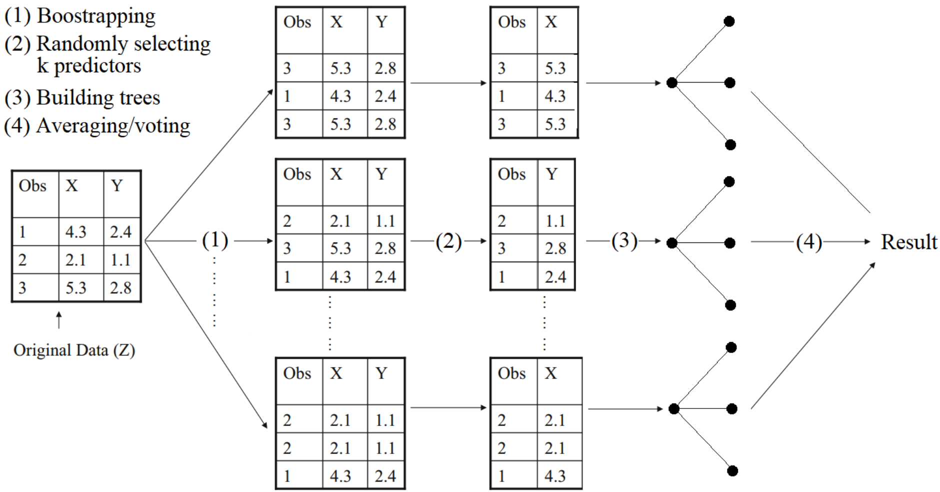
Random forest

- Random forests provide an improvement over bagged trees by way of a random small tweak that *decorrelates* the trees
- If there is one very strong predictor in the data set, along with a number of other moderately strong predictors, all of the trees in bagging will use the strongest predictor in the top split.
- The predictions from the bagged trees will be highly correlated.
- Averaging many highly correlated responses does not lead to as large of a reduction in variance.
- Random forests overcome this problem by forcing each split to consider only a subset of the predictors (m random predictors instead of all p predictors)
- How to choose value of m, the number of random predictors using in each split, is out of scope of this course. In practice, they use $m = [\sqrt{p}]$ in classification and $m = [p/3]$ in regression

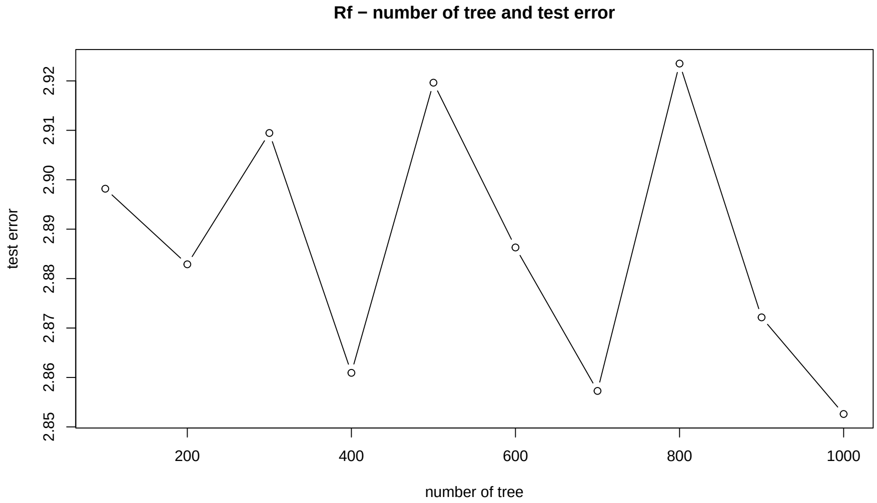
Random forest



Random forest



Random forest and Boston data set



Random forest and Default data set

Using bagging and random forest to predict *default* variables.

```
dat<-Default
standardize<-function(x){x<-(x-mean(x,na.rm=TRUE))/sd(x,na.rm
for (col in names(dat)){
  if((col!="default")&class(dat[,col]) %in% c("integer","numer
    dat[,col]<-standardize(dat[,col])
  }
}
set.seed(1)
test_index<-createDataPartition(dat$default, times = 1, p = 0.
train<-dat[-test_index,]
test<-dat[test_index,]
```

Random forest and Default data set

Using bagging and random forest to predict *default* variables.

```
bag.fit<-randomForest(default~.,
                      data=train,mtry=3, #number of predictors
                      importance=TRUE,ntree=500) # number off
bag.pred<-predict(bag.fit,newdata=test,type="class")
table(bag.pred,test$default)
```

```
##
## bag.pred   No  Yes
##      No  4788  114
##      Yes   46   53
```

Boosting

- Boosting is a general approach that can be applied to many statistical learning methods for regression or classification.
- In random forest method, the trees are grown independently using multiple copies of the original training data set using the bootstrap.
- Applying boosting to tree-based model, the trees are grown *sequentially*: each tree is grown using information from previously grown trees
- Instead of growing some large trees which fit the data hard and potentially overfitting, the boosting approach *learns slowly*
- Boosting involves combining a large number of decision trees, $f^1, \cdots, f^B$ where f^{j+1} is the tree grown from fitting predictors to the residuals from f^j.

Boosting for regression tree

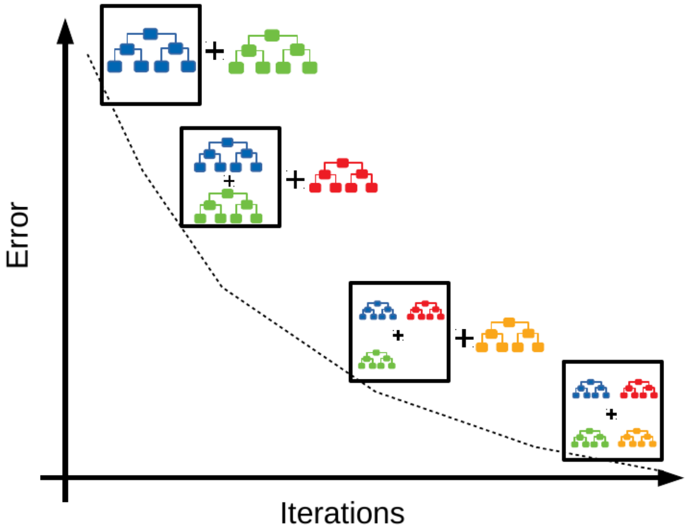
Boosting with parameter B, d, λ.

- Step 1: Let $\hat{f} = 0$ and $res_i = y_i$ for all i in the training set.
- Step 2: For each $b = 1, 2, \cdots, B$
    - Fit a d-nodes tree to (**X,res**) and get a prediction $\hat{f}_b$
    - Update the prediction: $\hat{f} = \hat{f} + \lambda \times \hat{f}_b$
    - Update the residual: $res = res - \lambda \times \hat{f}_b$
- Step 3: Final output of boosted model:

$$Output = \sum_{b=1}^{B} \lambda \, \hat{f}_b$$

where B is the number of trees and λ is the learning rate. (Typical values of λ are 0.01 and 0.001)

Boosting



Boosting and Boston data set

```r
library(gbm)
boost.fit<-gbm(medv~.,data=train,
          n.trees = 5000, # parameter B
          interaction.depth=5, # parameter d
          shrinkage = 0.005) # parameter lambda
```

```
## Distribution not specified, assuming gaussian ...
```

```r
boost.pred<-predict(boost.fit, newdata=test, n.trees=5000)
sqrt(mean((boost.pred-test$medv)^2))
```

```
## [1] 2.755437
```

Xgboost

```
library(xgboost)
dtrain<-data.matrix(dplyr::select(train,-medv))
xgb<-xgboost(data = dtrain,
            label = train$medv,nround=1000,
             max.depth = 3, eta = 0.05,verbose = 0)

pred <- predict(xgb, data.matrix(dplyr::select(test,-medv)))
sqrt(mean((pred-test$medv)^2))
```

```
## [1] 2.572969
```