# Block2assign1

Shwetha

11/23/2020

## Assignment 1. Ensemble methods

Fitting random forest to training data and test data

```
rforest_1 = randomForest(trainlabels~., data = traindata, ntree = 1, nodesize = 25,
                         keep.forest = TRUE)
rforest_10 = randomForest(trainlabels~., data = traindata, ntree = 10, nodesize = 25,
                          keep.forest = TRUE)
rforest_100 = randomForest(trainlabels~., data = traindata, ntree = 100, nodesize = 25,
                           keep.forest = TRUE)

y_1 = predict(rforest_1,testdata)
y_10 = predict(rforest_10,testdata)
y_100 = predict(rforest_100,testdata)

missclass=function(X,X1){
  n=length(X)
  return(1-sum(diag(table(X,X1)))/n)
}

y1_missclass = missclass(y_1,testlabels)
y10_missclass = missclass(y_10,testlabels)
y100_missclass = missclass(y_100,testlabels)
```

Misclassification for test data of n = 1000 with 1 tree = 0.188

Missclassification for test data of n = 1000 with 10 trees = 0.142

Missclassification for test data of n = 1000 with 100 trees = 0.112

Here we can observe that missclassification error reduces as the number of trees increases.

### 1

Repeating the procedure for 1000 training datasets of size 100.

```
missclass_large_sample = function(condition,ns = 25){
mce_1 = mce_10 = mce_100 = c()
set.seed(12345)
for(i in 1:1000){
```

```
  x1<-runif(100)
  x2<-runif(100)
  train<-cbind(x1,x2)
  y<-as.numeric(eval(parse(text = condition)))
  trainlabels<-as.factor(y)

  rf_1 = randomForest(trainlabels~., data = train, ntree = 1, nodesize = ns,
                      keep.forest = TRUE)
  rf_10 = randomForest(trainlabels~., data = train, ntree = 10, nodesize = ns,
                       keep.forest = TRUE)
  rf_100 = randomForest(trainlabels~., data = train, ntree = 100, nodesize = ns,
                        keep.forest = TRUE)

  x1<-runif(1000)
  x2<-runif(1000)
  test<-cbind(x1,x2)
  y<-as.numeric(eval(parse(text = condition)))
  testlabels<-as.factor(y)

  y_1 = predict(rf_1,newdata = test)
  y_10 = predict(rf_10,test)
  y_100 = predict(rf_100,test)

  mce_1[i] = missclass(y_1,testlabels)
  mce_10[i] = missclass(y_10,testlabels)
  mce_100[i] = missclass(y_100,testlabels)

}
mat = matrix(c(mean(mce_1),mean(mce_10),mean(mce_100),var(mce_1),
               var(mce_10),var(mce_100)),ncol = 2,nrow = 3)
colnames(mat) = c("mean","variance")
rownames(mat) = c("1 tree","10 trees","100 trees")
return(mat)
}
```

Mean and variance of misclassification errors :

```
a = missclass_large_sample(condition = "x1 < x2")
knitr::kable(a)
```

| | mean | variance |
|---|---|---|
| 1 tree | 0.204920 | 0.0030569 |
| 10 trees | 0.136552 | 0.0009773 |
| 100 trees | 0.111754 | 0.0008972 |

## 2

Repeating the same procedure as above with condition : $x1 < x2$

Mean and variance of misclassification errors :

```
b = missclass_large_sample(condition = "x1 < 0.5")
knitr::kable(b)
```

| | mean | variance |
|---|---|---|
| 1 tree | 0.102442 | 0.0196025 |
| 10 trees | 0.015674 | 0.0006173 |
| 100 trees | 0.005717 | 0.0000417 |

## 3

Repeating the same procedure as above with condition : (x1 $<$ 0.5 AND x2 $<$ 0.5) OR (x1 $>$ 0.5 AND x2 $>$ 0.5)

Mean and variance of misclassification errors :

```
c = missclass_large_sample(condition = "(x1 < 0.5 & x2 < 0.5) |
                           (x1 > 0.5 & x2 > 0.5)", ns = 12)
knitr::kable(c)
```

| | mean | variance |
|---|---|---|
| 1 tree | 0.249110 | 0.0134500 |
| 10 trees | 0.116820 | 0.0027368 |
| 100 trees | 0.071714 | 0.0013713 |

## 4

### a

The mean and variance of the error decreases as the number of trees grow. In general misclassification reduces and one gets better results with increasing the number of trees. However rate improvement in the result decreases with increase in trees, in above examples we can see that difference in mean of misclassification error between 1 tree and 10 trees is more that between 10trees and 100 trees.

### b

For the third condition, we can see that misclassification error was slightly higher than the other two conditions, but the results improve with the use of more trees. Random forest uses bagging , ie it picks a sample of observations rather than all of them. For a large observation, when the number of trees are too small, then some observations will be predicted only once or even not at all. This will result in a chance of high missclassification error. But using large number of trees solves this, resulting in better classification.

### c

Higher the variance , higher will be sensitivity of the model to the training data , and this will reduce the accuracy of the prediction when we fit it to the test data which which may be different from training data.