Computer lab 1 block 2

Martynas Lukosevicius, Alejo Perez Gomez, Shwetha Vandagadde Chandramouly

02/12/2020

Statement of Contribution

- Assignment 1 - Shwetha Vandagadde Chandramouly
- Assignment 2 - Alejo Perez Gomez
- Assignment 3 - Martynas Lukosevicius

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.

Mean and variance of misclassification errors for condition $x1 < x2$:

	mean	variance
1 tree	0.211323	0.0032972
10 trees	0.141111	0.0009715
100 trees	0.116073	0.0009574

2.

Repeating the same procedure as above with condition : $x1 < 0.5$

Mean and variance of misclassification errors :

	mean	variance
1 tree	0.094046	0.0173643
10 trees	0.015022	0.0005776
100 trees	0.006120	0.0000780

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 :

	mean	variance
1 tree	0.250956	0.0136136
10 trees	0.119573	0.0031013
100 trees	0.074899	0.0012565

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.

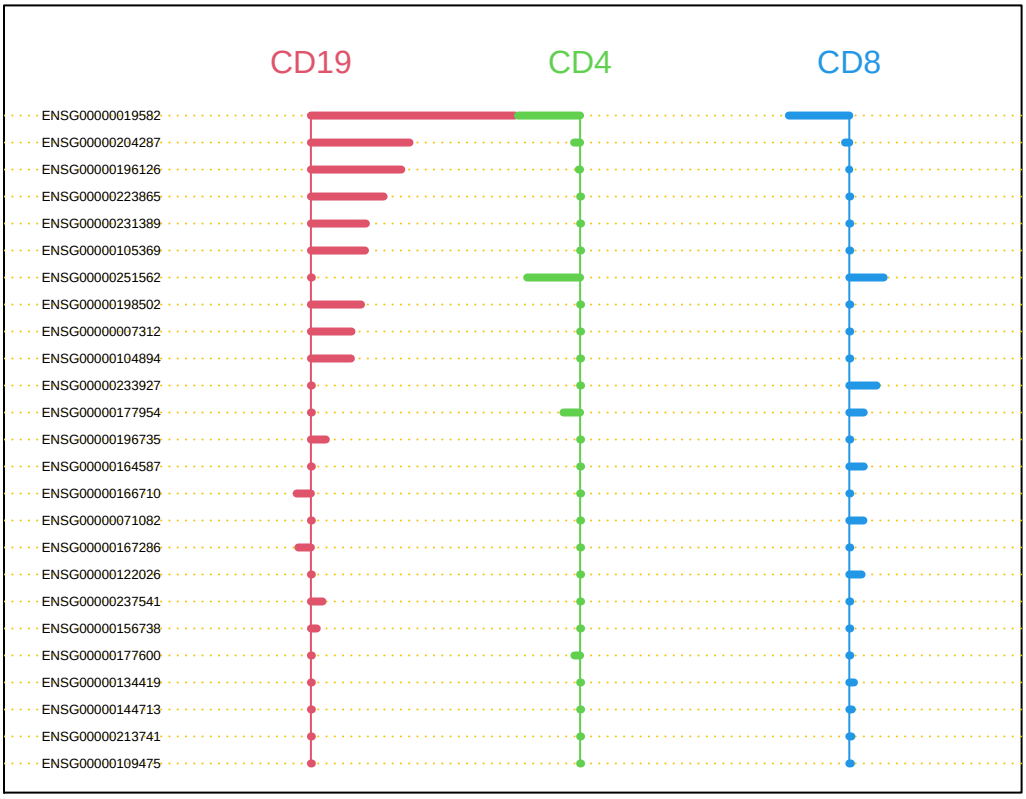
c.

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.

Assignment 2. Mixture models

Assignment 3. High-dimensional methods

1.



Centroid plot shows distance from gene mean value to global mean value. Positive or negative shows if mean is greater or smaller than the global mean. No, it can't happen that all values for some gene in centroid plot are positive, because distance is calculated to the mean of all values of the gene . Number of genes selected when threshold = 7.212944 : 25.

2.

Most contributed genes: ENSG00000019582, ENSG00000204287. alternative names: CD74, HLA-DRA. These genes are marker genes for B cells

Test error rate: 0.1111111

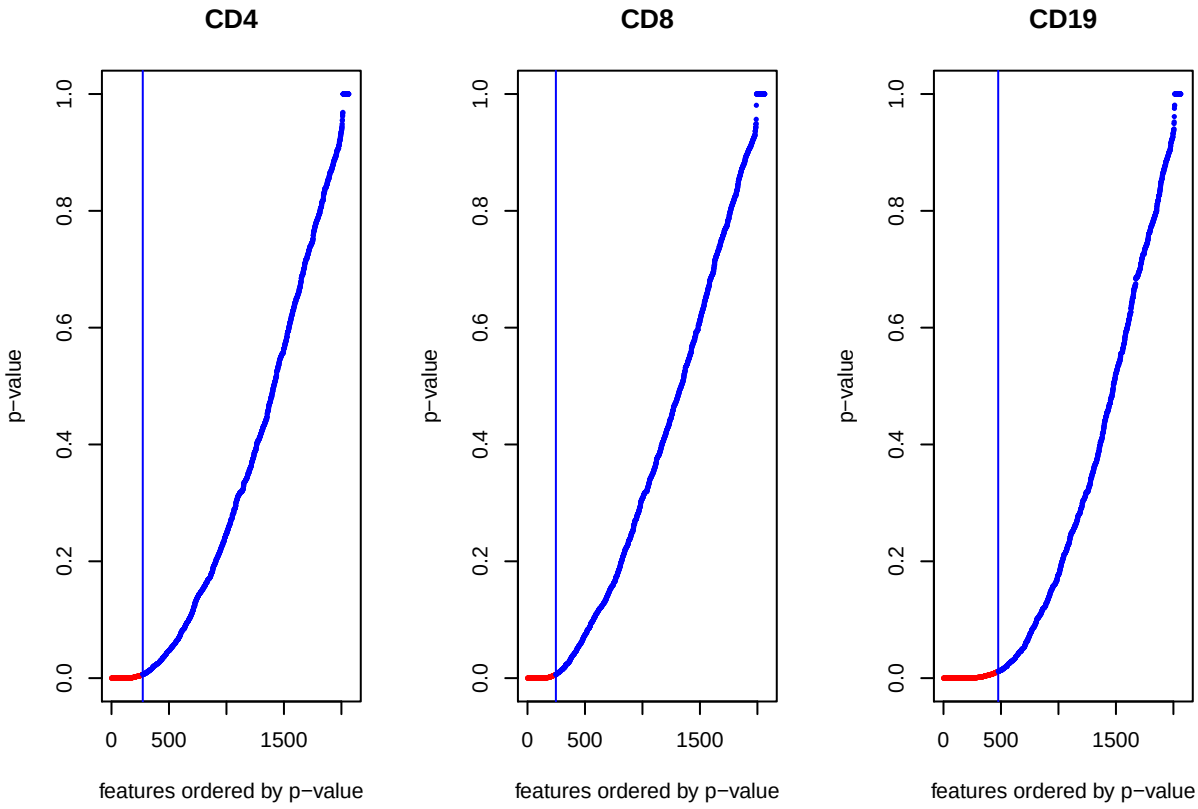
3.

Table bellow shows test error rate and number of features used.

	Test error rate	N. features
NSC	0.110	25
EN	0.056	54
SVM	0.022	2085

I prefer Elastic net, because SVM might overfit and use a lot of parameters, NSC might have underfited.

4.



points symbolize genes, red genes are rejected meaning that they have an effect on corresponding cell. Table bellow shows the amount of rejected genes for each cell.

cell	rejected
CD4	272
CD8	247
CD19	476

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(randomForest)
set.seed(12345)
x1<-runif(100)
x2<-runif(100)
traindata<-cbind(x1,x2)
y<-as.numeric(x1<x2)
trainlabels<-as.factor(y)

set.seed(12345)
```

```
x1<-runif(1000)
x2<-runif(1000)
testdata<-cbind(x1,x2)
y<-as.numeric(x1<x2)
testlabels<-as.factor(y)
#plot(x1,x2,col=(y+1))
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)


missclass_large_sample = function(condition,ns = 25){
mce_1 = mce_10 = mce_100 = c()
set.seed(12345)
  x1<-runif(1000)
  x2<-runif(1000)
  test<-cbind(x1,x2)
  y<-as.numeric(eval(parse(text = condition)))
  testlabels<-as.factor(y)
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)

  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)
}
a = missclass_large_sample(condition = "x1 < x2")
knitr::kable(a)
b = missclass_large_sample(condition = "x1 < 0.5")
knitr::kable(b)

c = missclass_large_sample(condition = "(x1 < 0.5 & x2 < 0.5) |
                           (x1 > 0.5 & x2 > 0.5)", ns = 12)
knitr::kable(c)
library(readr)
data0=read_csv("geneexp.csv")
data=data0[, -1]
data$CellType=as.factor(data$CellType)
set.seed(12345)
n <- dim(data)[1]
id=sample(1:n, floor(n*0.7))
train=data[id,]
test=data[-id,]
library(pamr)
rownames(train)=1:nrow(train)
genenames <- colnames(train[ ,-ncol(train)])
x <- t(train[ ,-ncol(train)])
y=train[[ncol(train)]]
mydata=list(x=x,y=as.factor(y),geneid=as.character(1:nrow(x)), genenames=genenames)
model=pamr.train(mydata)
cvmodel=pamr.cv(model,mydata)
minError <- which.min(cvmodel$error)
bestTh <- cvmodel$threshold[minError]

pamr.plotcen(model, mydata, threshold = bestTh)
numberOfGenes <- model$nonzero[minError]
listOfGenes  <- pamr.listgenes(model, mydata, bestTh)
bestGenes <- mydata$genenames[as.vector(as.numeric(listOfGenes[1:2,1]))]
rownames(test)=1:nrow(test)
genenames <- colnames(test[ ,-ncol(test)])
x <- t(test[ ,-ncol(test)])
y=test[[ncol(test)]]
testdata=list(x=x,y=as.factor(y),geneid=as.character(1:nrow(x)), genenames=genenames)
predictions <- pamr.predict(model, testdata$x, threshold = bestTh)

#knitr::kable(table(testdata$y,confusionMatrix))
library(glmnet)
x <- as.matrix(train[ ,-ncol(train)])
```

```r
cv.elastic <- cv.glmnet(x, train$CellType, alpha = 0.5, family = "multinomial")

res1 <- predict(cv.elastic, newx = as.matrix(test[ ,-ncol(test)]), s = cv.elastic$lambda.min, type = "cl
#knitr::kable(table(testdata$y,res1))
library("kernlab")

x <- as.matrix(unname(train[ ,-ncol(train)]))
svp = ksvm(x = x,y = train$CellType,  kernel='vanilladot')

res2 <- predict(svp, as.matrix(test[ ,-ncol(test)]))

results <- matrix(c(signif(sum(testdata$y != predictions)/length(predictions),digits = 2), numberOfGenes
results <- rbind(results,c(signif(sum(testdata$y != res1)/length(res1),digits = 2), cv.elastic$nzero[whi
results <- rbind(results, c(signif(sum(testdata$y != res2)/length(res2),digits = 2), nrow(mydata$x)))
row.names(results) <- c("NSC", "EN", "SVM")
colnames(results) <- c("Test error rate", "N. features")
knitr::kable(results)
test <- function(name){
  y <- ifelse(data$CellType == name, 1, 0 )
  df <- data.frame(character(), numeric())
  for (i in 1:(dim(data)[2]-1)) {
    test <- t.test(unlist(data[ ,i])~y, data = data, alternative = "two.sided")
    df <- rbind(df, c(colnames(data)[i], test$p.value))
  }
  colnames(df) <- c("name", "pvalue")
  return(df)
}

plotres <- function(name){

  results <- test(name)

  results <- results[order(as.numeric(results$pvalue)), ]

  M <- length(results$pvalue)
  for (i in 1:M) {
    if(as.numeric(results$pvalue[i]) > (0.05 * i / M)){
      break
    }

  }
  L <- i-1
  amount <<- rbind(amount, c(name, L))
  p <- as.numeric(results$pvalue[L])
  rejected <- ifelse(as.numeric(results$pvalue) <= p, 1, 0 )

  row.names(results) <- c(1:length(results$pvalue))

  title <- paste0(name)
  plot(c(1:L),
       results$pvalue[1:L],
       pch = 20,
       col = "red",
```

```
       xlim = c(0,M),
       ylim = c(0,1),
       cex = 0.5,
       xlab = "features ordered by p-value",
       ylab = "p-value",
       main = title)
  abline(v=L, col="blue")
  points(c(L+1:M), results$pvalue[L+1:M], pch = 20, col = "blue", xlim = c(0,M), ylim = c(0,1), cex = 0
}

amount= data.frame(name= character(), amount = numeric())
par(mfrow=c(1,3))
plotres("CD4")
plotres("CD8")
plotres("CD19")
colnames(amount) <- c("cell", "rejected")
knitr::kable(amount)
```