732A95: Lab 1 block 2
Introduction to Machine Learning

Carles Sans Fuentes

November 21, 2016

Assignment 1

On assignment 1, the csv-file cube.csv that contains points in a two-dimensional plane must be read and a myspline function that depends on vectors that fits a piecewise linear model to the data with target and feature and knot positions given by knots must be implemented.

Assignment 1.1-1.2

In order to do so, a matrix is created with the existing independent variable and 2 new ones is designed a result of resting the knots to the independent variable $(X - E_i)$. In our case, a matrix of 3 variables + the intercept is created and then linearly regressed to do predict how the model must behave. From the function, the beta coefficients, the predicted values, the MSE and a plot between true and predicted values are presented (see Figure 1 below).

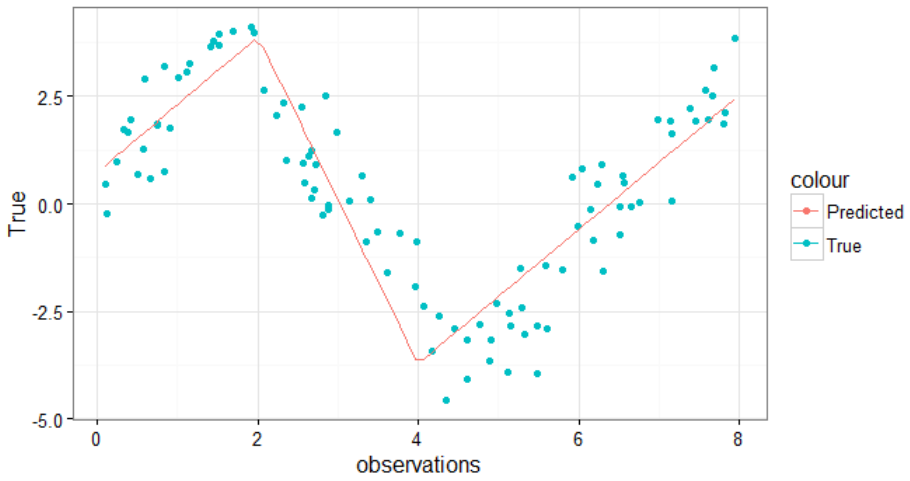


Figure 1: Plot on Predicted vs True estimations of Y

It can be seen that our model predicts quite well the data, having as a Mean Squared residual 0.9651468. Nevertheless, our prediction could be better since data seems to be continuous following a waving shape. Nevertheless, a linear model is easier to interpret and predict.

Assignment 1.3

In this section it is asked to use the smooth.spline function to fit the same data and compare it with our previous model. It can be seen that smooth splines give less accurate predictions since

it has a MSE of 11.08196. The smoothing function, which is an ordinary cubic spline is not able to predict well since there is a decrease at the beginning of the model where there should be an increase on the values. This model has more parameters than the previous one, but it is not able to capture the variables as good as the other one.

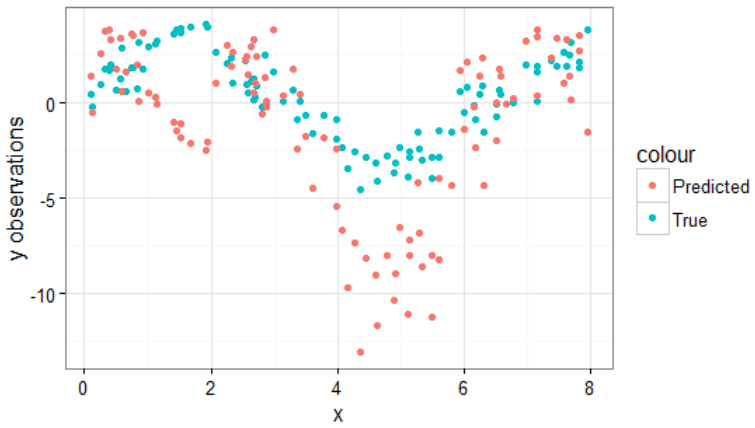


Figure 2: Plot on Predicted vs True estimations of Y

Assignment 2

In this assignment, it is read the *influenza.xlsx* excel file which contains weekly data on the mortality and the number of laboratory-confirmed cases of influenza in Sweden.

Figure 3: Levels of influenza and mortality across time

Assignment 2.1

In this section, it is been asked to use time series plots to visually inspect how the mortality and influenza number vary with time, using Time in the x-axis (see Figure 3 below).

From this plot, it can be seen that it exists a positive correlation between influenza and mortality across time as well as some seasonality every year.

Assignment 2.2

In this section, it is asked to use gam function from mgcv package to fit a GAM model in which Mortality is normally distributed and modeled as a linear function of Year and spline function of Week. The underlying probabilistic model is

$$Y_t = a + Year_t + S[Week]_t + \epsilon_t,$$

being Y the dependent variable, a the intercept, Year the time variable, Week as a spline and epsilon as the error measurement of the regression.

Assignment 2.3

In this section we are asked to Plot predicted and observed mortality against time for the fitted model and comment on the quality of the fit and say whether it exists some trend or not.

The coefficients that are non-significant are Intercept and Year , which means that the parametric coefficients are not needed in fitting this model. The probabilistic model can be written as:

$$Y = \beta_0 + \beta_1 Year + s_1(Week) + \ldots + s_9(Week) + \epsilon$$
$$Y = -658.058 + 1.219 Year + 230.322 Week + \ldots + 108.338 Week + \epsilon$$

The coefficients can be seen in the Figure 4 below.[2]

```
> gm_model$coefficients ##Probabilistic model
 (Intercept)          Year     s(Week).1     s(Week).2     s(Week).3     s(Week).4
-652.058068     1.218568   230.321746  -867.306717   226.364890   652.224843
  s(Week).5     s(Week).6     s(Week).7     s(Week).8     s(Week).9
-112.723164  -675.208550   177.412152 -1687.678156   108.337507
```

Figure 4: Coefficients of the variables selected by GAM

The MSE of the data is 8603.573. Here below in Figure 5 , predicted and observed mortality across Time is represented and in 6 observed Mortality is represented along Weeks and Years respectively. From the former, it can be seen that the model fits quite well the data. Nevertheless, the variance of the predicted values is always limited, being smaller than the real variance of the model. From both plots it can be seen that it exists a seasonality pattern of an increase in Mortality every year at the same weeks each year. Further evaluation should be done from different fields to evaluate the meaning of that, and what represents.

Figure 5: Regression of the

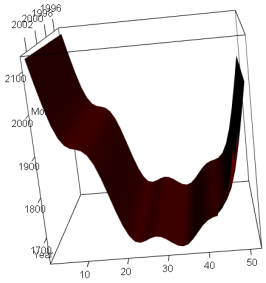


Figure 6: 3D Mortality against Year and Week

Assignment 2.4

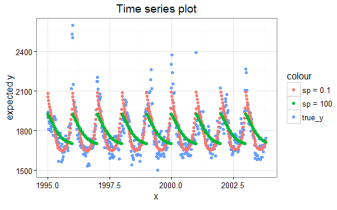
In this section we are asked to examine how the penalty factor of the spline function in the GAM model from step 2 influences the estimated deviance of the model.

The penalties chosen for the model have been respectively 0.1 and 100, and the following plot has been represented in 7 below. It can be seen that the bigger the smoothing parameter of λ, the lower the variance of the model is. When the maximum smoothness of λ_i is provided, the regression is just done for the most important parameter which leads to a linear regression (explaining the regression with just one variable), and the less flexible the model is, being in this case worse fitted.

The relation of the penalty factor to the degrees of freedom is given by

$$df_\lambda = \sum_{k=1}^{n} 1/(1 + \lambda * d_k)$$

which is confirmed by the results got, leading for higher λ to to more linear regressions and less flexible models since the degrees of freedom of the model decrease, increasing the penalty factor.



Assignment 2.5

In this section we are asked to use the model obtained in step 2 to plot the residuals and the influenza values against time in one plot (see figure 8). It can be seen that it does exist a pattern on the residuals of the data and the outbreaks of Influenza, because it happens at the same point in time. For that reason, it can be evaluated that it exists a positive correlation between the residuals and the outbreaks of influenza.

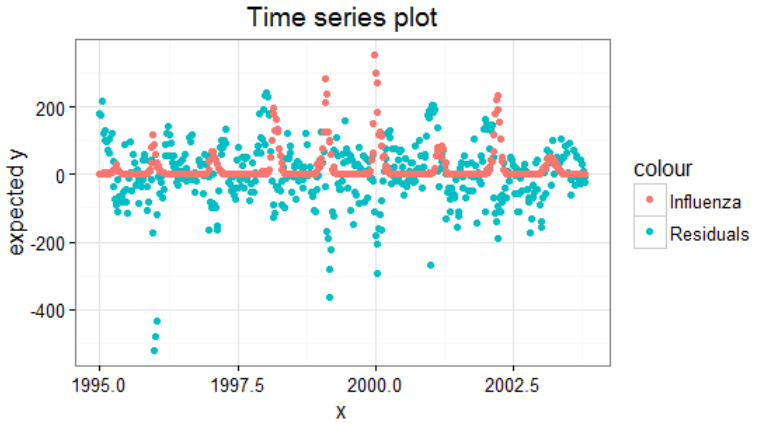


Figure 8: Plot across time between the residuals of Mortality and the values of Influenza

Assignment 2.6

In this section, a GAM model must be calculated in which mortality is to be modelled as an additive function of the spline functions of year, week, and the number of confirmed cases of influenza.

For doing so, the GAM function is used and, for each variable, a parameter is specified equivalently to the number of unique elements there exist on each variable. The result can be seen in the following figure 9 below.

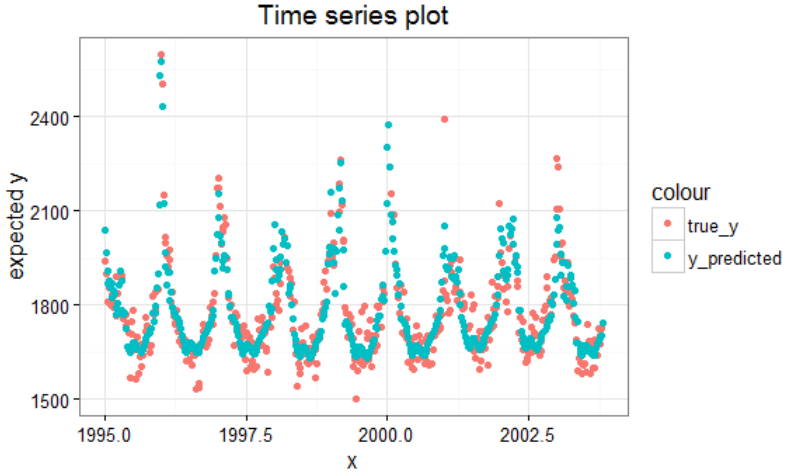


Figure 9: True versus real values of Mortality against Influenza

It can be seen that this last model fits better the data than the previous one. The mean squarer error of this model is lower than the ones in previous activities, being 3777.822. For sure, mortality is influenced by the outbreaks of influenza since the MSE predicted is better when taking into account Influenza.

Appendix

Assignment 1

```
##R script activity 1

##Assignment 1

setwd("C:/Users/M/Desktop/Statistics and Data Mining Master/Semester 1/Second part o

library(readxl)

mydata<- read.csv2("C:/Users/M/Desktop/Statistics and Data Mining Master/Semester 1/


#1.1

myspline=function(X = mydata$x,Y= mydata$y, knots= c(2,4)){
  m=length(knots)
  n=length(X)
  H=matrix(0,nrow=n,ncol=m+2) #new features
  H[,1]=1
  H[,2]=X
  for(i in 1:m){
    for(j in 1:n){
      #MISSING: Insert equation computing basis function values from X values
      H[j,i+2]<- H[j,2]-knots[i]
      if(H[j,i+2] < 0) {H[j,i+2] <- 0}
    }
  }
  colnames(H)<- c("intercept", "y", "x-2", "x-4")
  #use H matrix and Y in order to get the optimal basis function coefficients 'beta'
  data<-as.data.frame(cbind(Y,H[,2:(m+2)]))
  beta<-lm(Y~., data = data)$coefficients
  #predicted values
  Yhat=H%*%beta
  MSE<- sum((Yhat-Y)**2)/length(Y)
  ##creating a list to store Beta coefficients and the plot
  res<- list()
  #plot the original and predicted data in one graph
  library(ggplot2)
  dataplot<- data.frame(observations= X,True= Y, Predicted=Yhat)
  myplot<-ggplot2::ggplot(data = dataplot)+
    geom_point(aes(x=observations, y = True, color = "True"))+
    geom_line(aes(x = observations, y = Predicted, color = "Predicted"))+
    theme_bw()

  res["beta.coef"]]<-beta
  res["predicted.values"]]<-Yhat
  res["plot predicted vs true"]]<-myplot
  res["sum squared residuals"]]<-MSE

  return(res)
}

myspline()
```

```
##3

mysmooth<-smooth.spline(x= mydata$x, y = mydata$y)
predicted_y<-predict(mysmooth, x = mydata$y)$y

MSE<-sum((predicted_y-mydata$y)**2)/length(mydata$y)
datfram<- data.frame(x= mydata$x, True=mydata$y, Predicted = predicted_y )
myplot2<-ggplot2::ggplot(data = datfram)+
  geom_point(aes(x = x, y = True, color = "True"))+
  geom_point(aes(x = x, y = Predicted, color = "Predicted"))+
  ylab("y observations")+   theme_bw()

myplot2
```

Assignment 2

```
##Assignment 2

#1


library(readxl)

mydata<-
  read_excel("C:/Users/M/Desktop/Statistics and Data Mining Master/Semester 1/
  Second part of the semester/intro to machine learning/lab3/influenza.xlsx")

library(ggplot2)
myplot3<- function(x, variable1, variable2 ){
  datframe<- data.frame(Time= x, Mortality =variable1, Influenza = variable2 )
  ggplot2::ggplot(data = datframe)+
  geom_point(aes(x = Time, y = Mortality, color = "Mortality"))+

  geom_point(aes(x = Time, y = Influenza, color = "Influenza"))+
  ylab("expected y")+ggtitle("Time series plot")+
  theme_bw()
}


myplot3(x= mydata$Time, variable1 =mydata$Mortality, variable2 = mydata$Influenza )

#2


library(mgcv)


gm_model<- gam(Mortality~Year+s(Week), data= mydata)
summary(gm_model)
gm_model$coefficients ##Probabilistic model

y_pred<-predict(object = gm_model, newdata = mydata)


##3
```

```
myplot4<- function(x, true_y, y_predicted ){
  datframe<- data.frame(x= x, true_y=true_y, y_predicted = y_predicted )
  ggplot2::ggplot(data = datframe)+
    geom_point(aes(x = x, y = true_y, color = "true_y"))+
    geom_point(aes(x = x, y = y_predicted, color = "y_predicted"))+
    ylab("expected_y") + ggtitle("Time_series_plot")+
    theme_bw()
}

myplot4(x = mydata$Time , true_y = mydata$Mortality, y_predicted =y_pred )

#install.packages("rgl")
#install.packages("akima")
library(rgl)
library(akima)



s=interp(x=mydata$Year,y = mydata$Week, fitted(gm_model))
persp3d(s$x,s$y,s$z, col = "red", xlab = "Year", ylab = "Week" , zlab = "Mortality")



MSE<-sum((mydata$Mortality-y_pred)**2)/length(mydata$Mortality)


##4

##Model for low lambda value  = 0.1
gm_model_1<- gam(Mortality~Year+s(Week), data= mydata, sp = 0.1)

##Model for high lambda value  = 100

gm_model_2<- gam(Mortality~Year+s(Week), data= mydata, sp = 100)

gm_model_1$coefficients
gm_model_2$coefficients

###Predicting new values of Y
y_pred_1<-predict(object = gm_model_1, newdata = mydata)
y_pred_2<-predict(object = gm_model_2, newdata = mydata)



## Plotting new values with the old ones
library(ggplot2)
myplot5<- function(x, true_y, y_predicted, y_predicted2 ){
  datframe<- data.frame(x= x,
                        true_y=true_y,
                        y_predicted = y_predicted,
                        y_predicted2= y_predicted2  )
  ggplot2::ggplot(data = datframe)+
    geom_point(aes(x = x, y = true_y, color = "true_y"))+
    geom_point(aes(x = x, y = y_predicted, color = "sp_=_0.1"))+
    geom_point(aes(x = x, y = y_predicted2, color = "sp_=_100"))+
        ylab("expected_y") + ggtitle("Time_series_plot")+
    theme_bw()
}
```

```
myplot5(x = mydata$Time , true_y = mydata$Mortality,
        y_predicted = y_pred_1, y_predicted2 = y_pred_2 )
```

###5

```
residuals<- y_pred - as.vector(mydata$Mortality)

myplot6<- function(x, variable1, variable2 ){
  datframe<- data.frame(x= x, Residuals=variable1, Influenza = variable2 )
  ggplot2::ggplot(data = datframe)+
    geom_point(aes(x = x, y = Residuals, color = "Residuals"))+
    geom_point(aes(x = x, y = Influenza, color = "Influenza"))+
    ylab("expected_y") + ggtitle("Time_series_plot")+
    theme_bw()
}


myplot6(x= mydata$Time, variable1 =residuals, variable2 = mydata$Influenza)
```

###6

```
k_infulenza<-length(unique(mydata$Influenza))
k_Year<-length(unique(mydata$Year))
k_Week<-length(unique(mydata$Week))
gm_model_3<- gam(Mortality~s(Year, k=k_Year)+
                   s(Week, k = k_Week)+
                   s(Influenza, k = k_infulenza), data= mydata)

gm_model_3$coefficients

y_pred_3<-predict(object = gm_model_3, newdata = mydata)


myplot4(x = mydata$Time , true_y = mydata$Mortality, y_predicted =y_pred_3 )

MSE3<-sum((as.vector(mydata$Mortality)-y_pred_3)**2)/length(mydata$Mortality)
```

732A54: Lab 1
Introduction to Machine Learning

Carles Sans Fuentes

November 9, 2016

Assignment 1

On assignment 1, the K-nearest neighbor algorithm must be implemented to predict whether an e-mail is spam or not (spam = 1, not spam = 0). This is done for the *spambase.xlsx* data set which is split in two parts (train and test) in order to model on the train data and then predicting for test.

Assignment 1.1-1.3

After partitioning data into train and test, containing each partition 1370 observations, the **knearest** function is implemented with the cosine similarity as the distance measure.
To do so, the train data is used for classification, indexing by distances for the Kth elements that are nearer to each observation, taking K = 5. Then, all these indexes are substituted by the real values of the e-mail train data (spam column being spam=1 or 0 otherwise). By doing this, a 1370 x K data frame (5 in our case) is got. With this information, the mean is computed for the 5 observations and spam = 1 is set if the probability > .5 or 0 otherwise (not spam = 0). The resulting confusion matrix is shown below on table 1.

	Predicted Class 0	**Predicted Class 1**
True Class 0	787	158
True Class 1	119	306

Table 1: Confusion matrix for train data, K = 5.

The corresponding missclassification rate when classifying the training data is $(158 + 119)/1370$, which is approximately 20 %.

Now the classification for spam is made for the test data with the same probability 0.5 and K = 5 nearest neighbor. The confusion matrix is shown below in table 2.

	Predicted Class 0	**Predicted Class 1**
True Class 0	695	242
True Class 1	193	240

Table 2: Confusion matrix for test data, K = 5.

The corresponding missclassification rate when classifying the test data is $(242 + 193)/1370$, which is approximately 32 %.

Assignment 1.4

On 1.4 we are asked to project the training and test data set, but now for K = 1. First, the training data is classified and the confusion matrix is shown in the table 3.

	Predicted Class 0	**Predicted Class 1**
True Class 0	939	6
True Class 1	2	423

Table 3: Confusion matrix for train data, K = 1.

The misclassification rate for the prediction of the training data with K = 1 is $(6+2)/1370 = 0.6\%$, which is reasonable because each observation gets classified to the same class as it self. Still,there is an overfit for this train model because we are just taking the class of the nearest neighbor into account, which leads to a really accurate model for our training data but probably not as good for the test later on.

The test data is also used to classify emails with K = 1 (see table 4).

	Predicted Class 0	**Predicted Class 1**
True Class 0	639	298
True Class 1	178	255

Table 4: Confusion matrix for test data, K = 1.

The missclassification rate is $(298 + 178)/1370$ which is approximately 35 % . Comparing both missclassification rates for K= 1 and K = 5 in the table 2, we see that K =5 provides with a better model than K=1 for prediction. As stated above, the model for K = 1 is overfitted for the training data, leading to worse prediction results.

Assignment 1.5

On assignment 1.5 it is asked to use the kknn formula on R for K = 5 and report the confusion matrix and the missclassification rate while comparing it with previous results in the other steps.

The kknn algorithm returns a vector of probabilities for the observations. For that reason, the same procedure as with the knearest implementation must be followed, setting each probability to 1 if it is bigger than 1 (spam = 1) and 0 otherwise (not spam = 0) and then creating the confusion matrix for the predicted values.

The kknn algorithm with K = 5 provides with the following confusion matrix (see table 5).

	Predicted Class 0	**Predicted Class 1**
True Class 0	640	297
True Class 1	177	256

Table 5: Confusion matrix for test data, K = 5 of kknn R algorithm

The missclassification rate is $(297 + 177)/1370$ which is approximately 35 %. Comparing missclassification rates for the implemented model with K= 5 (see table 2), and the kknn R algorithm in table 5, it is seen that the model implemented provides with a lower missclassification rate for the same K.

Assignment 1.6

On 1.6 we are asked to model the kknn and our implemented knearest function on a vector of probabilities that goes from 0.05 to 0.95 and compare its results with FPR and TPR. The plot can be seen below in figure 1.

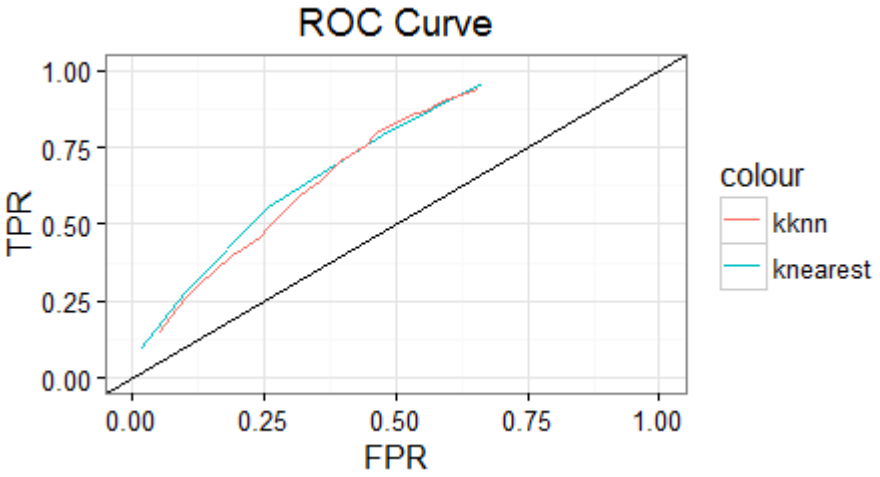


Figure 1: ROC curves for knearest(model implemented) and kknn(R model)

As it can be seen, the model implemented is strictly better in general terms but both gives really similar output predictions. the kknn seems to have better performance for some values around 0.5 for FPR. In general terms, our two models behave the same way: as pi increases for 0.05 to 0.95, the probability of detecting positives (TPR) and the probability of false alarms (FPR), increases. Since the best option should be TPR 1, FPR = 0, the smallest distance to that point must be the best probability predictor for our model. In this case, that should be near where the TPR should be near 0.6 and the FPR on 0.25.

Assignment 2

On assignment 2, the *machines.xlsx* data set, which contains information about the lifetime of certain machines, is used in order to determine the warranty time of the process.

Assignment 2.1-2.2

On this part the data set is imported and we must find the maximum log-likelihood for our data. To do so, the log-likelihood of the exponential is submitted directly inside the function. Two inputs are given to the function: data and theta (which is a vector of theta that goes from 0 to 4 by 0.01), and it gives as a result the vector of log-likelihoods based on the theta vector . It cannot be directly seen in the plot (see figure 2) which is the theta value that maximizes the model so it must be checked on the vector to have the best likelihood. The best theta is is 1.13 for the maximum log-likelihood of -42.2948.

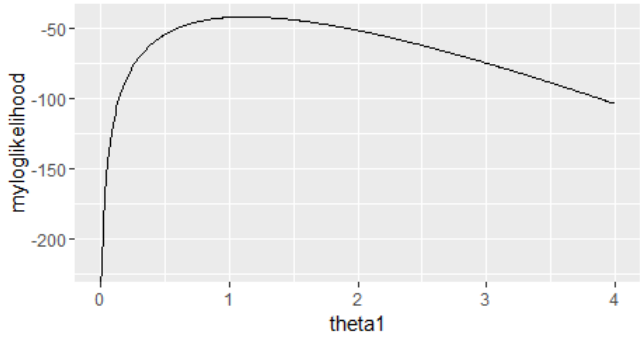


Figure 2: Plot of the log likelihood for the whole data of machines

Assignment 2.3

On this section we are asked to plot both likelihood estimators for the whole machine data set and just the first 6 estimations (see figure 3). From the plot,it can be seen that the MLE distribution obtained for the six first observations is strictly better than for the whole data case. Nonetheless, the MLE is only asymptotically consistent for large samples so that it can leads to bias for small samples, as in our case for just 6 observations. That is why, the likelihood of the whole data is more reliable and probably less biased so that it is better.

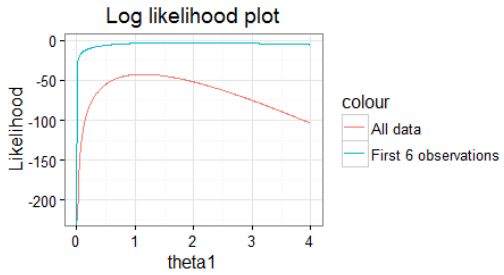


Figure 3: Likelihood for the whole data and the first 6 observations of the data set given theta

Assignment 2.4

In this section the maximum log-likelihood for the posterior must be calculated. This is done through multiplying the prior with likelihood.

Different likelihoods are obtained for different thetas of the posterior. It cannot be directly seen in the plot (see figure 4) which is the best value so it must be checked on the vector to see the best likelihood. The best theta is is 0.91 for the maximum log-likelihood of -50.10905.

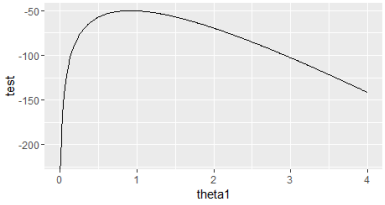


Figure 4: Posterior log-likelihood distribution

Assignment 2.5

In this section it is meant to create 50 new observations given our model and create a histogram to visualize the results (see figure 5). It can be seen as a conclusion that our theta is accurate since the shape of the original and generated data in plotted looks similar.

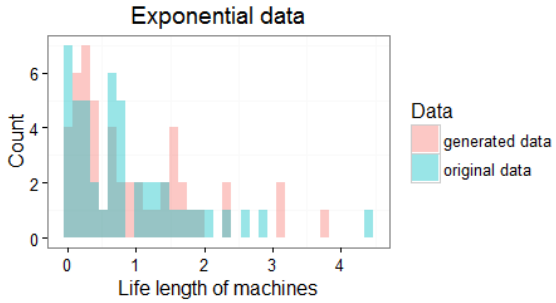


Figure 5: Histogram of random generated and original data

Appendix

Assignment 1

```
####Assignment 1
#a

library(readxl)

##Reading table

data<- read_excel("C:/Users/M/Desktop/Statistics and Data Mining Master
/Semester 1/Second part of the semester/intro to machine learning/spambase.xlsx")

n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
test=data[-id,]



col_length<-ncol(data)-1
train_complete<-train[,1:(col_length+1)] #used for the kknn algorithm
train<- train[,1:col_length] #used for our algorithm

test_complete<- test[,1:(col_length+1)] #used for the kknn algorithm
test<-  test[,1:col_length] #used for our algorithm

#Columns to compare and train and test on our algorithm
train_spam = data[id,ncol(data)]
test_spam = data[-id,ncol(data)]

###2
knearest<- function( data, K , newdata, prob){
  data<- as.matrix(data)
  mytest <- as.matrix(newdata)

  x_est <- data/sqrt(rowSums(data**2))
  y_est <- mytest/sqrt(rowSums(mytest**2))

  matrix_c <- x_est %*% t(y_est)
  D <- 1 - matrix_c #Distance  cosines computed

  #Matrix with the position of the K closest values
  K_matrix <- t(apply(D, 2, order)[ 1:K, ])



  for(i in 1:nrow(K_matrix)){
    for(j in 1:ncol(K_matrix)){
      K_matrix[i,j] = train_spam[K_matrix[i,j]]
      #replaced the five K values with the values of 1,0 in the column of train
    }
  }
  if(length(prob)==1){
  # doing the mean of the K closest values and replace it for 1 or 0
    if(K == 1){
```

```
    K_matrix = apply(K_matrix,1, function (i){ifelse(i>prob,1,0)})
  }
  else{
    K_matrix = apply(K_matrix,1, function (i){ifelse(mean(i)>prob,1,0)})
  }

  if(identical(data,mytest)){
  #If the data introduced and the data to predict are the same,
  #then table on train

    result = table(K_matrix,train_spam) #Confusion matrix
    #missclassification rate
    misclassification_rate = (result[1,2]+result[2,1])/sum(result)

    res<- list()
    res[["missclassification_rate"]]<-misclassification_rate
    res[["result"]]<-result
    return(res)

  }else{ #If the data introduced and the data to predict are
        #the same, then table on test
    result = table(K_matrix,test_spam) #Confusion matrix
    #missclassification rate
    misclassification_rate = (result[1,2]+result[2,1])/sum(result)
    res<- list()
    res[["missclassification_rate"]]<-misclassification_rate
    res[["result"]]<-result
    return(res)
  }

}else{
  ## Setting values for the confusion matrix
  misclassification_rate <- integer(length(prob))
  confusion_matrix <-list()

  FP<- integer(length(prob))
  TP<- integer(length(prob))
  FN<- integer(length(prob))
  TN<- integer(length(prob))
  N_pos <- integer(length(prob))
  N_neg <- integer(length(prob))


  for(i in 1:length(prob)){
    if(K == 1){
      ## it is just a data frame of one column
      Kmatrix <- apply(K_matrix,1, function(a){ifelse(a>prob[i],1,0)})
    }
    else{
      Kmatrix <- apply(K_matrix,1, function(a){ifelse(mean(a)>prob[i],1,0)})
    }
  confusion_matrix[[i]]<-table(Kmatrix,test_spam)
  result <- table(Kmatrix,test_spam)
  misclassification_rate[i] <-(result[1,2]+result[2,1])/sum(result)


  FP[i]<- result[2,1]
  TP[i]<- result[2,2]
```

```
    FN[i]<- result[1,2]
    TN[i]<- result[1,1]
    N_pos[i]<- result[2,2]+ result[1,2]
    N_neg[i]<- result[1,1]+ result[2,1]

    }
    sensitivity <-TP/N_pos
    FPR<- (FP/N_neg)
    specificity <- 1- FP/N_neg

    #creating a list to store the results
    res<- list()
    res[["missclassification_rate"]]<-misclassification_rate
    res[["confusion_matrix"]]<-confusion_matrix
    res[["sensitivity"]] <- sensitivity
    res[["FPR"]]<- FPR
    res[["specificity"]]<-specificity

    return(res)
}}


###3

knearest( data= train, K =5, newdata= test, prob = 0.5)
knearest( data= train, K =5, newdata= train, prob = 0.5)



###4
knearest( data= train, K =1, newdata= test, prob = 0.5)
knearest( data= train, K =1, newdata= train, prob = 0.5)


# For k = 1, the number of missclassifications increases from 40.1%
# when K = 3 to 42.9%.

###5

library(kknn)

mykknn_fittedValues <-
kknn(Spam~., train =train_complete, test = test_complete, k = 5)$fitted.values

mykknfunction <- function(fittedvalues=mykknn_fittedValues, prob=mypi){
  if(length(prob)==1){
    # We just get a vector of probabilities that need to be changed to 1
    # if> prob, else 0
    mychoiceofprob <- ifelse(fittedvalues > prob, 1, 0)

    confusion_matrix<- table(mychoiceofprob,test_spam)

    misclassification_rate <-
    (confusion_matrix[1,2]+confusion_matrix[2,1])/sum(confusion_matrix)

    #creating a list to store the results

    res<- list()
```

```
    res[["missclassification_rate"]]<-misclassification_rate
    res[["confusion_matrix"]]<-confusion_matrix
    return(res)


  }else{
    misclassification_rate <- integer(length(prob))
    # The probability is a vector,
    # so it needs to be stored n a vector for each number
    FP<- integer(length(prob))
    TP<- integer(length(prob))
    FN<- integer(length(prob))
    TN<- integer(length(prob))
    N_pos <- integer(length(prob))
    N_neg <- integer(length(prob))
    confusion_matrix<- list()
    for(i in 1:length(prob)){

      mychoiceofprob <- ifelse(fittedvalues > prob[i], 1, 0)


      confusion_matrix[[i]]<- as.matrix(table(mychoiceofprob,test_spam))
      simpleconfusion <- table(mychoiceofprob,test_spam)

      misclassification_rate[i] <-
        (simpleconfusion[1,2]+simpleconfusion[2,1])/sum(simpleconfusion)
      FP[i]<- simpleconfusion[2,1]
      TP[i]<- simpleconfusion[2,2]
      FN[i]<- simpleconfusion[1,2]
      TN[i]<- simpleconfusion[1,1]
      N_pos[i]<- simpleconfusion[2,2]+ simpleconfusion[1,2]
      N_neg[i]<- simpleconfusion[1,1]+ simpleconfusion[2,1]
    }
    sensitivity <-TP/N_pos
    FPR<- (FP/N_neg)
    specificity <- 1- (FP/N_neg)


    res<- list()
    res[["missclassification_rate"]]<-misclassification_rate
    res[["confusion_matrix"]]<-confusion_matrix
    res[["sensitivity"]] <- sensitivity
    res[["FPR"]]<-FPR
    res[["specificity"]]<-specificity

  return(res)
  }
}



mykknfunction(mykknn_fittedValues, 0.5)



###6

mypi<- seq(0.05,0.95, 0.05)
```

```
#Storing the values to plot

Myknn_sensitivity<-
        mykknfunction(mykknn_fittedValues, prob =mypi)$sensitivity
Myknn_specificity<-
        mykknfunction(mykknn_fittedValues, prob =mypi)$specificity
Myknn_FPR<-
        mykknfunction(mykknn_fittedValues, prob =mypi)$FPR
Myfunction_sensitivity<-
        knearest(data = train, K=5, newdata =test, prob =mypi)$sensitivity
Myfunction_specificity<-
        knearest(data = train, K=5, newdata =test, prob =mypi)$specificity
Myfunction_FPR<-
        knearest(data = train, K=5, newdata =test, prob =mypi)$FPR


#False Positive Rates (FPR)
#Probability of false alarm:
#system alarms (1) when nothing happens (true=0)

library(ggplot2)


myPlot<- function(yourfunction_sens,yourfunction_fpr , packagefunction_sens,
                                packagefunction_fpr, title = "ROC_Curve"){

  #Creating data frame for the plot with the vectors entered in the function
 data <- data.frame(my_sens = yourfunction_sens,
                    my_fpr = yourfunction_fpr,
                    kknn_sens = packagefunction_sens,
                    kknn_fpr = packagefunction_fpr)

  result<-ggplot2::ggplot(data = data)+
            geom_line(aes( x = my_fpr, y = my_sens, color = "knearest"))+
            geom_line(aes(x = kknn_fpr, y = kknn_sens, color = "kknn"))+
            xlab("FPR" )+ylab("TPR")+ ggtitle(title)+
            geom_abline(intercept =0,slope =1)+
            xlim(0,1)+ylim(0,1)+theme_bw()
  return(result)
  }

ROCplot<-myPlot(Myfunction_sensitivity ,Myfunction_FPR,Myknn_sensitivity , Myknn_FPR)
```

Assignment 2

```
##### Assignment 2

  ##1.Importing data
  set.seed(12345)
  library(readxl)
  data<-read_excel("machines.xlsx")

  ##2. The distribution type of x is an exponential distribution
  logLikelihood<- function(theta, x){

    result<-integer(0)
    ##it is the loglikelihood(product and logarithms)
```

```
  for(i in 1:length(theta)){
    result[i]<- nrow(x)*log(theta[i])-theta[i]*sum(x)
   }
  return(result)
}

theta1<-seq(0,4,0.01)   #theta between 0 and 4 by 0.01
myloglikelihood<-logLikelihood(theta = theta1,x = data)

library(ggplot2)
#plotting with ggplot2
qplot(theta1,myloglikelihood, geom="line")
#find the best theta for the data created
my_theta_loglikelihood<-data.frame(theta1,myloglikelihood)

mybesttheta<- my_theta_loglikelihood[order
(my_theta_loglikelihood$myloglikelihood, decreasing = TRUE),][1,]

print(mybesttheta)
#According to the plot it looks like 1, but by checking on the data.frame
#created the best theta is 1.13 for the max loglikelihood of -42.2948

##3.
# the first six first observations of the data frame
myfirst_6_observations<-as.matrix(data[1:6,])

#calling the function to the six obs
myloglikelihood_6_obs<-logLikelihood(theta1, myfirst_6_observations)

mydatatoplot<-
data.frame(theta1 = theta1,
                      all_data= myloglikelihood,
          six_obs = myloglikelihood_6_obs )
myplot_activity3 <- ggplot(mydatatoplot)+
                       geom_line(aes(x=theta1, y= all_data,
                                     color = "All data"))+
                       geom_line(aes(x=theta1, y= six_obs,
                                     color = "First 6 observations"))+
                       ggtitle("Log likelihood plot")+
                       ylab("Likelihood")+ theme_bw()

# The first 6 observations give a better likelihood.
# However, having only six observations is not enough to get a
#reliable good likelihood so that the all data likelihood is more reliable


##4.
bayes<- function(theta, x, lambda=10){ # lambda default 10
  prior<-numeric(length(theta)) #empty vector to fill with new values
  for ( i in 1:length(theta)){
    prior[i]<-log(lambda * exp(-lambda*theta[i]))  # log of prior

  }
  loglikke<-logLikelihood(theta,x) # calling loglike function above
  io<-prior+loglikke  # adding prior and loglike according to ln rules
  return(io)
}
test<-bayes(theta=theta1, x= data, lambda = 10)
```

```
qplot(theta1,test, geom="line")  #plotting

my_theta_loglikelihood2<-data.frame(theta1,test)

mybesttheta2<- my_theta_loglikelihood2[order
(my_theta_loglikelihood2$test, decreasing = TRUE),][1,]

print(mybesttheta2)

#Previous best theta was 1.13 for the max loglikelihood of -42.2948
#New theta is 0.91 for the max loglikelihood of -50.10905.


##5.

#random from exp distribution with the best theta from 2.2
randomexp<-rexp(50, rate=mybesttheta[[1]])
randomexp<- as.data.frame(randomexp)
#Setting the same name as the original data to get it together
colnames(randomexp)<- "Length"

binding<-rbind(randomexp,data)
Data<- c(rep("generated data",50), rep( "original data",48))
Data<- as.data.frame(Data)
# Setting the data frame for the plot with data and if this data is
mydataframe <- data.frame(binding$Length, Data)  original or generated
hist_plot<- ggplot(mydataframe, aes(binding$Length, fill = Data))+
  geom_histogram(alpha = 0.4, bins=35, position = 'identity')+
  labs(title= "Exponential data",
       x = "Life length of machines",
       y = "Count")+theme_bw()+
  theme(panel.grid.major = element_blank())
```

732A95: Lab 2 block 2
Introduction to Machine Learning

Carles Sans Fuentes

December 8, 2016

Lab2a block 2 Exercise 2

The file bodyfatregression.xls, which contains records of waist measure (cm), weight (kg) and body fat (%) for 110 persons will be used to to carry out a regression tree analysis using the function tree() of the R package tree. For the prediction, body fat will be the response variable and waist measure and weight the predictor variables.

Exercise 2.1

In this section it an upper bound of the squared error of the bagging regression tree is going to be estimated using 2/3 of the data as train and 1/3 as test.

By bagging (doing samples of the same size of the data set with replacement) 1000 times on the train data set, different trees are estimated with different estimated Mean standard errors (having at the end 1000). Assuming that error terms are 0 and uncorrelated, we get that the maximum upper bound that we can get (that is, the largest bagged error that we can get from our model) is the mean of our different MSE models. That is in my implementation 36.81453.

Exercise 2.2

In this section the error must be again calculated but using Cross-validation of three folds. To do so, instead of using the train data, the whole data set is used, and using my implementation of a function called random split data, a the folds are created and passed to each sample, predicting and getting standards errors three times for each bagging, and computing the mean of it. Then, 1000 MSE has been got from my function, calculating again the mean to know the MSE of the model. This is in my implementation 31.28953, which is lower than just using bagging.

Exercise 2.3

In this section it the bagging regression tree that you would return to the user must be implemented.

A function has been implemented to return all the trees that has been used to fit the model. The code is provided in Appendix 2, 2.3.

Assignment 4

In this section the file spambase.xls, which contains information about the frequency of various words, characters, etc. for a total of 4601 e-mails must be used. The last column of the data set is called Spam, where it can be see whether these e-mails have been classified as spams (spam $\bar{1}$)

or regular e-mails (spam $\bar{0}$). It must to evaluated the performance of Adaboost classification trees and random forests on the spam data, providing a plot showing the error rates when the number of trees considered are 10, 20 and so on until 100. To do so, data must be partition in 2/3 for training and 1/3 as hold-out test data.

Random forest and Adaboost are two techniques used for modeling. Adaboost is a technique of classification that combine (weak) classifiers to produce a more accurate (committee) one. This is done through weighting higher the wrong predictions (using an exponential loss function) to predict again and draw a new regression prediction line until a good prediction is done by returning a weighted average of the different output predictions. Random Forest uses bagging on decision trees for modeling outputs. The main advantage of it is its simplicity to understand the output but this is normally of low accuracy and of high variance, given its dummy classification if there are not many leaves. A plot with the number of error predictor is provided for the number of trees considered (see figure 1 below).

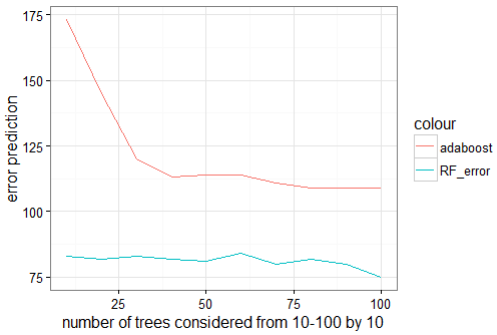


Figure 1: Plot of the number of error predictions using the Spam base data set for Adaboost and RandomForest models for different number of trees

It can be seen that there are lower number of errors in both cases for higher number of trees, though it goes down slowly. Nevertheless, random forest gives much better output for whatever number of trees, and its prediction improve does not change by allowing more number of trees. On the other side, Adaboost needs at least 40 trees to get an stabilized error measure.

Lab2b block 2 Exercise 1

In this exercise the EM algorithm for mixtures of multivariate Bernoulli distributions must be implemented using some data created from distributions.

In order to compute the EM algorithm, three main steps that has been followed:

- Calculation of the Z matrix as function: The Z matrix is meant to be the conditional probability of each ten Bernoulli independent observations for each π_k on the sum of all probabilities. The result must be a matrix of 1000x3 conditional probabilities.
- The loglikelihood function, which basically must be the log sum of the conditional probabilities shown as below:

  The likelihood is meant to be the product of our bernoulli distribution.

$$Likelihood(x) = \prod_{N=1}^{1000} p(x_i)$$

  This can be translated as the probability of each x as the product of the sum our bernoulli conditional distributions.

$$Likelihood(x) = \prod_{N=1}^{1000} \sum_{k=1}^{3} p(x_i|k) * p(k)$$

which is in our case the product of our different and independent probability Bernoulli observations

$$Likelihood(x) = \prod_{N=1}^{1000} \sum_{k=1}^{3} \prod_{k=1}^{3} (\prod_{D=1}^{10} p(x_i|k) * p(k))$$

and if we take the log, we get that

$$Log(Likelihood(x)) = \prod_{N=1}^{1000} log(\sum_{k=1}^{3} \prod_{k=1}^{3} (\prod_{D=1}^{10} p(x_i|k) * p(k)))$$

The loglikelihoods are calculated for 100 iterations. Theorically, the loglikelihood should converge at some point to be able to choose a model.

- The last step is to compute the new μ and π that will be passed to the model. This is just done by applying a couple of formulas that have been shown on the slides.

At the end, a graph is provided with the evolution of the likelihood as well as the last μ and π.

During the iteration, it can be seen the changing of the values of μ for each π as well as the evolution of mu. They are reported below:

```
> pi
[1] 0.3416794 0.2690298 0.3892909

> mu
           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] 0.4727544 0.3869396 0.6302224 0.3156325 0.6875038 0.2030173
[2,] 0.4939501 0.4757687 0.4584644 0.4711358 0.5413928 0.4976325
[3,] 0.5075441 0.5800156 0.4221148 0.7100227 0.2965478 0.7571593
         [,7]      [,8]      [,9]      [,10]
[1,] 0.7832090 0.1435650 0.8827796 0.03422816
[2,] 0.4569664 0.4869015 0.4909904 0.37087402
[3,] 0.2400675 0.8424441 0.1188864 0.99033611
```

It can be seen that the likelihood gets stable around -6300 (see figure 2 below).Whatever point from that line could be taken as a proper likelihood estimator with its parameters.

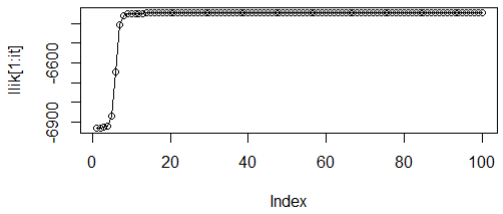


Figure 2: Plot of evolution of the likelihood

Here it can be seen also the plot of the last mu (see figure 3 below). Not in the image though, but on the process of iteration, it can be seen how the mu are changing and moving through the process, which is seen how the model learns by itself.

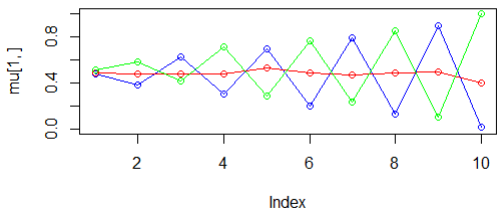


Figure 3: Plot of the mu

Appendix

Assignment 1

```
##R script activity 1
####Assignment 1

##2

#2.1
set.seed(1234567890)
data<- read.csv2("C:/Users/M/Desktop/Statistics and Data Mining Master/Semester 1/Second part
    of the semester/intro to machine learning/lab2block2/bodyfatregression.csv")
library(tree)

n<-dim(data)[1]
id<-sample(1:n, floor(n*2/3))
train=data[id,]
test= data[-id,]


random_split_data<- function(data, Nfolds){

  set.seed(12345)
  n <- dim(data)[1]
  sample_numbers<-sample(1:n)

  ##indexes rows for train
  randomsplitvector<-split(sample_numbers, ceiling(seq_along(1:n)/(n/Nfolds)))

  # indexes rows complementary of train
  randomsplitvector_compl<- list()
  for(i in 1:Nfolds){
    randomsplitvector_compl[[i]]<- sample_numbers[!sample_numbers %in% randomsplitvector[[i]]]

  }
  res<-list()
  res[["train"]]<-randomsplitvector_compl
  res[["test"]]<- randomsplitvector
  return(res)

}



bootstrap<- function(data, B, testdata, method, Nfolds){
  #inititializing data
  if(method == "holdout"){
    MSE<- integer(B)
    for(i in 1:B){
      n<- dim(data)[1]
      id <- sample(1:n, replace = TRUE)
      in_train<-data[id,]

      fit <- tree(formula = Bodyfat_percent~Waist_cm+Weight_kg, in_train)
      my_pred<- predict(fit, newdata = testdata)
      residuals<- testdata$Bodyfat_percent-my_pred

      MSE[i]<- (sum((residuals)**2))/nrow(testdata)
    }

    res<- list()
    res[["MSE"]]<- MSE


    return(res)

    }

  if(method == "CV"){
    randomsplitvector_train<- random_split_data(data = data, Nfolds = Nfolds)$train
    randomsplitvector_test<- random_split_data(data = data, Nfolds = Nfolds)$test
    ###Partitioning data with its real data
    datapartitioned_train<-list()
    datapartitioned_test<-list()


    MSE<- integer(B)


    for(k in 1:B){
      n<- dim(data)[1]
      id <- sample(1:n, replace = TRUE)
      datasample<-data[id,]
      for(i in 1:Nfolds){
```

```
81
82          datapartitioned_train[[i]]<- datasample[randomsplitvector_train[[i]],]
83
84        }
85
86
87        for(i in 1:Nfolds){
88
89          datapartitioned_test[[i]]<- datasample[randomsplitvector_test[[i]],]
90
91        }
92        #Setting differents numbers using sample to partition data in an outside function
93
94
95        ##Calculating residuals and chossing the smaller one of the Nfolds
96
97        for(j in 1:Nfolds){
98          fit <- tree(formula = Bodyfat_percent~Waist_cm+Weight_kg, datapartitioned_train[[j]])
99          my_pred<- predict(fit, newdata = datapartitioned_test[[j]])
100         residuals<- datapartitioned_test[[j]]$Bodyfat_percent-my_pred
101         MSE[k]<- (sum((residuals)**2))/length(my_pred)
102         }
103
104     }
105     return(MSE)}
106 }
107 bootstrap_holdout<-bootstrap(data= train, B=1000, testdata = test, method = "holdout")
108 mean(bootstrap_holdout$Tree)
109 bootstrap_CV<-bootstrap(data= data, B=1000, testdata = data, method = "CV", Nfolds = 3)
110
111 my_mean_CV<-mean(bootstrap_CV)
112
113 ##average error of your models cannot be higher than its mean so that
114 upper_bound_holdout<-mean(bootstrap_holdout)
115
116 hist(bootstrap_holdout)
117
118
119 #2.2
120
121 fit <- tree(formula = Bodyfat_percent~Waist_cm+Weight_kg, data= train)
122 N_tree<- cv.tree(fit, K = 3)
123
124 best_size <- N_tree$size[which.min(N_tree$dev)]
125
126
127 #2.3
128
129 TreeBagRegression<- function(data, B){
130   #inititializing data
131
132     fit<- list()
133     for(i in 1:B){
134       n<- dim(data)[1]
135       id <- sample(1:n, replace = TRUE)
136       in_train<-data[id,]
137
138       fit[[i]] <- tree(formula = Bodyfat_percent~Waist_cm+Weight_kg, in_train)
139
140     }
141     return(fit)
142 }
143
144 TreeBagRegression(data = data, B=10)
145
146
147 ####4
148
149 data2<- read.csv2("C:/Users/M/Desktop/Statistics and Data Mining Master/Semester 1/Second part
      of the semester/intro to machine learning/lab2block2/spambase.csv")
150 data2[["Spam"]]<- as.factor(data2[["Spam"]]) #necessary for data to work
151 names(data2)
152 n<-dim(data2)[1]
153 set.seed(1234567890)
154 id <- sample(1:n, floor(n*2/3))
155 train2 <- data2[id,]
156 test2 <-  data2[-id,]
157
158 #install.packages("mboost")
159 library(mboost)
160 library(tree)
161 library(party)
162 library(randomForest)
163 library(ggplot2)
164 ##Drawing sequence of number of trees
165 myseq<- seq(10,100,10)
166 adaboost_error<- integer(length(myseq))
```

```
RF_error<- integer(length(myseq))
for(size in 1:length(myseq)){
  #Adaboost formula
  ada_boost<-blackboost(formula = Spam~.,
                        data = train2,
                        family = AdaExp(),
                        control = boost_control(mstop = myseq[size]))
  #Random forest usage
  RF<- randomForest::randomForest(formula = Spam~., data =train2, ntree = myseq[size])

  #Predicting and getting errors for adaboost
  ada_pred<-predict(ada_boost, newdata = test2, type ="class")
  conf_ada <-table(test2$Spam, ada_pred)
  adaboost_error[size] <- conf_ada[1,2]+conf_ada[2,1]

  #Predicting and getting errors for RF
  RF_pred  <-predict(RF, newdata = test2, type ="class")
  conf_RF <-table(test2$Spam, RF_pred)
  RF_error[size] <- conf_RF[1,2]+conf_RF[2,1]


}
result <- data.frame(adaboost_error= adaboost_error, RF_error = RF_error)

ggplot2::ggplot(result)+
  geom_line(aes(x = seq(10,100,10), adaboost_error, color = "adaboost"))+
  geom_line(aes(x = seq(10,100,10), RF_error, color = "RF_error"))+
  xlab("number of trees considered from 10-100 by 10")+ ylab("error prediction")+
  theme_bw()



tre<-tree(formula = Bodyfat_percent~Waist_cm+Weight_kg, train)
tre$frame
```

Assignment 2

```

###Assignment 2
set.seed(1234567890)
max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
N=1000 # number of training points
D=10 # number of dimensions
x <- matrix(nrow=N, ncol=D) # training data
true_pi <- vector(length = 3) # true mixing coefficients
true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions
true_pi=c(1/3, 1/3, 1/3)
true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
points(true_mu[2,], type="o", col="red")
points(true_mu[3,], type="o", col="green")
# Producing the training data
for(n in 1:N) {
  k <- sample(1:3,1,prob=true_pi)
  for(d in 1:D) {
    x[n,d] <- rbinom(1,1,true_mu[k,d])
  }
}
K=3 # number of guessed components
z <- matrix(nrow=N, ncol=K) # fractional component assignments
pi <- vector(length = K) # mixing coefficients
mu <- matrix(nrow=K, ncol=D) # conditional distributions
llik <- vector(length = max_it) # log likelihood of the EM iterations
# Random initialization of the paramters
pi <- runif(K,0.49,0.51)
pi <- pi / sum(pi)
for(k in 1:K) {
  mu[k,] <- runif(D,0.49,0.51)
}
pi
mu


for(it in 1:max_it) {
  plot(mu[1,], type="o", col="blue", ylim=c(0,1))
  points(mu[2,], type="o", col="red")
  points(mu[3,], type="o", col="green")
  #points(mu[4,], type="o", col="yellow")
  #Sys.sleep(0.5)

```

```
47
48    ###E-Step
49    E_step<- function(x,mu,pi){
50
51      #the number of diision output
52      ##computing the likelihoods for the  = 3 cases different cases
53        ##new way
54        probx_mu<- integer(N)
55
56        for(thousand in 1:N) {
57          prob<- 0
58          for(three in 1:K) {
59            prob <- prob + (pi[three] * prod((mu[three,]^x[thousand,]) * (1-mu[three,])^(1-x[
                  thousand,])))
60            ########################product of ######pi_k*##mu**x*(1-mu)**(1-x)
61            }
62          probx_mu[thousand]<- prob
63        }
64
65        # Calculating the matrix Z
66        for (thousand in 1:N) {
67          for (three in 1:K) {
68            z[thousand,three] <-(pi[three]*(prod((mu[three,]^x[thousand,])*(1-mu[three,])^(1-x[
                  thousand,])))))/probx_mu[thousand]
69          }
70        }
71        return(z)
72      }
73
74
75
76    z<-E_step(x=x,mu=mu,pi=pi)
77
78    count<-1
79
80    #2-llik[it] = the sum log of my z1 total
81    newLike<-matrix(0,nrow=1000,ncol=3)
82    for (n in 1:N) {
83      for (k in 1:K) {
84        newLike[n,k] <- (pi[k]*(prod((mu[k,]^x[n,]) * (1-mu[k,])^(1-x[n,]))))
85      }
86    }
87
88    llik[it]<-sum(log(rowSums(newLike)))
89
90    ##Breaking loop
91    if(count>1){
92      if(abs(llik[it])/abs(llik[it-1])<0.1){
93        stop(return(llik[it]))
94      }
95    }
96
97    count<-count+1
98
99
100   mstep<-function(x,z){
101     #new pi
102     pi<-apply(z,2,mean)
103
104     #new mu
105     den<-apply(z,2,sum)
106
107     for(ten in 1:D){
108       for(three in 1:K){
109
110       up_fraction <-sum(x[,ten]*z[,three])
111       mu[three,ten]<-up_fraction/den[three]
112     }}
113     res<-list(mu=mu, pi=pi)
114     return(res)
115   }
116     mu<-mstep(x=x,z=z)$mu
117     pi<-mstep(x=x,z=z)$pi
118
119   cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
120   flush.console()
121   # Stop if the lok likelihood has not changed significantly
122   # Your code here
123   #M-step: ML parameter estimation from the data and fractional component assignments
124   # Your code here
125 }
126 pi
127 mu
128 z
129 plot(llik[1:it], type="o")
130
131
```

732A95: Lab 2
Introduction to Machine Learning

Carles Sans Fuentes

November 16, 2016

Assignment 1

On assignment 1, an R function that performs feature selection in linear regression by using k-fold cross-validation must be implemented.

Assignment 1.1-1.3

In order to do so, different small functions have been implemented to evaluate the best model: an *all_Beta_subsets* function that returns a list of all possible combinations for the different input variables passed through and a *random_split_data* function that provides with the indexes for the K-fold cross-validation of the model.
These two functions are passed to the LmFold main function, which performs the linear regression calculations and returns a vector of the Mean Squared Errors. Then, the MSE is plotted for each of the 31 combinations of the data variables got in the *all_Beta_subsets* function with the numbers of variables each model has (see Figure 1). The lowest CV value found in my case is is 55.89168 for the case on the index 29, being the variables on the regression Agriculture + Education + Catholic + Infant.Mortality.

Agriculture and Education are variables that affect negatively the Fertility rates. I don't understand Agriculture negative correlation, whereas it is feasible that the more education a country has, the lower is the fertility rate (e.g. this can be seen today in Europe).Infant mortality rate also has a positive correlation to fertility measure. This is due to the fact that when high mortality ratios are expected on a population, the number of sons per family increases dramatically to ensure the survival of your family (e.g. High fertility rates and high mortality ratio in Africa). One omitted variable that could be interesting to regress and that will probably improve the quality of our regression would be a measure of income or GDP per capita of the country which will be probably negatively correlated.

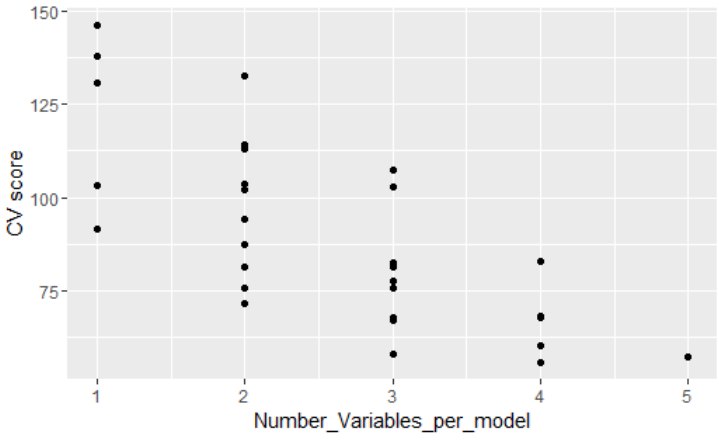


Figure 1: CV score vs number of variables

Assignment 2

In this assignment, it is aimed to investigate whether a near infrared absorbance spectrum can be used to predict the fat content of samples of meat from the the excel file *tecator.xlsx*.

Assignment 2.1

In this section, it is been asked to plot Moisture versus Protein (see figure 2 below). From the plot, it seems to exist a linear relation between both variables.

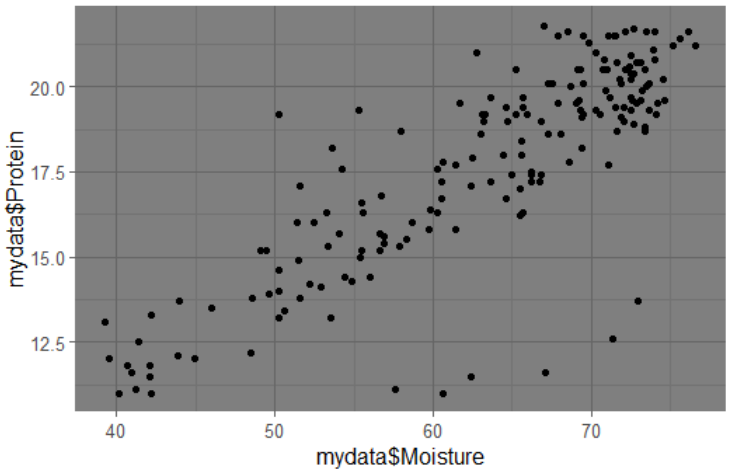


Figure 2: Plot Moisture against Protein

Assignment 2.2

In this section, it is asked to report a probabilistic model for the M_i. The model function proposed is a normal distribution with expected value as a polynomial function of protein to the ith power and variance σ^2. The probabilistic model is then given by:

$$Moisture_i \sim N(\sum_{i=1}^{M} W_i Protein^i, \sigma^2)$$

It is appropriate to use MSE criterion when fitting this model since data follows a linear relationship and the model is simple.

Assignment 2.3

In this section we are asked to divide the data in two parts (train and test, 50% and 50% respectively), and fitting each linear model created to evaluate which gives lower Mean Squared Error prediction according to the different plots and data observations. For that reason, MSE squared has been calculated and plotted linearly in Figure 3 for the train and test data. The plot works as follows: The first observation on the left stands for the simple model with just one variable, having on the second observation the second model as $Protein + Protein^2$, and so on. It turns out that the model that seems to fit better the data from the plot is model one, with just one variable. Also, and by checking the MSE, this one is also the one with lower Mean Squared Errors, being equal to 32.28 for the test, though it is the highest one checking for train data.

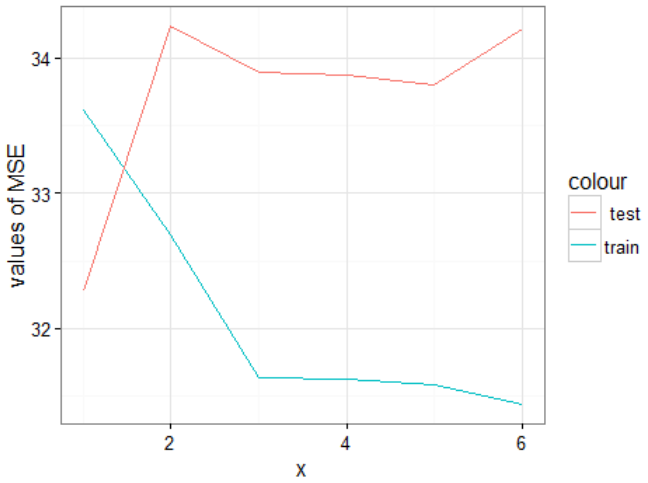


Figure 3: MSE comparison between train and test data

This can be explained from the bias-variance tradeoff, which stands for the minimization problem of error variance and overfitness of modeling when choosing the best model for your predictions. In our case, the first model seems to be the worst one for the trained data but it is the best one since the MSE is the smaller than other models as the error for each observation and prediction is more constant as well. This means that the other models are overfitted compared to the first one and that its complexity does not perform better for prediction.

Assignment 2.4

In this section a variable selection of a linear model in which Fat is response and Channel1-Channel100 are predictors is performed by using stepAIC. By using this system selection, we can see that only 63 variables are at the end chosen.

```
Coefficients:
(Intercept)     Channel1     Channel2     Channel4     Channel5     Channel7     Channel8
      7.093    10559.894   -12636.967     8489.323   -10408.967    -5376.018     7215.595
   Channel11    Channel12    Channel13    Channel14    Channel15    Channel17    Channel19
  -9505.520    37240.918   -41564.547    34938.179   -23761.451     4296.572    14279.808
   Channel20    Channel22    Channel24    Channel25    Channel26    Channel28    Channel29
 -23855.616    18444.906   -20138.426    18137.432    -7670.318    20079.898   -36351.014
   Channel30    Channel32    Channel34    Channel36    Channel37    Channel39    Channel40
  18071.276     3838.013    -9242.884     8070.938    -9045.588    18664.454   -20069.709
   Channel41    Channel42    Channel45    Channel46    Channel47    Channel48    Channel50
  22257.776   -21760.853    18145.804    -8225.696    -4986.549     2876.075   -13009.410
   Channel51    Channel52    Channel54    Channel55    Channel56    Channel59    Channel60
  29251.161   -26833.976    30954.862   -35183.287    14912.986    -8030.278    13071.416
   Channel61    Channel63    Channel64    Channel65    Channel67    Channel68    Channel69
  -7850.189    15059.275   -19909.466     4190.184    13850.508   -25873.365    18362.385
   Channel71    Channel73    Channel74    Channel78    Channel79    Channel80    Channel81
  -9223.910    12456.498    -5624.411    -7927.105    15473.188   -22391.895    13852.453
   Channel84    Channel85    Channel87    Channel88    Channel92    Channel94    Channel98
 -11442.630    20228.671   -15938.315     5647.072     6595.995    -5497.846    -8728.596
   Channel99
   8554.587
```

Figure 4: Coefficients of the variables selected by StepAIC

Assignment 2.5-2.6

In this section we are asked to fit a Ridge regression and Lasso model with the same predictor and response variables as in the previous step as well as presenting a plot showing how model coefficients depend on the log of the penalty factor and report how the coefficients change with λ.

By normalizing our data and using the glmnet package, we get to use ridge and lasso directly and getting the following plots (Figure 5 and 6 for Ridge and Lasso respectively.

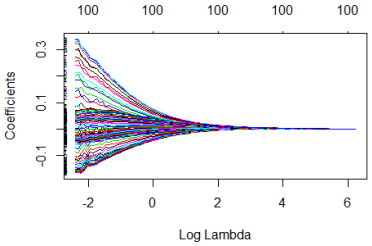


Figure 5: Ridge Shrinkage numbers

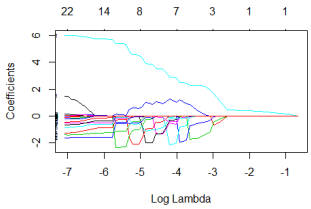


Figure 6: Lasso Shrinkage numbers

Comparing both plots (Ridge and Lasso), we can see that Ridge diminishes the values of all the variables to almost 0 and then one should take the larger values of the variables for the evaluation of the most important ones. By doing so, the choice it is not directly given, and even though there at the end there are a great amount which are almost discarded by having a value almost equal to 0, it is really difficult to see which variables are more important and useful quickly. On the other side, when using LASSO, which basically shrinks variables subject to the parameter of λ to 0, a variable selection is automatically performed for the ones with stronger relation easing the model and the variable selection.

Assignment 2.7

In this section we are asked to use cross-validation to find the optimal LASSO model as well as reporting and answer how many variables were chosen by the model. Also, conclusions as well as presenting a plot showing the dependence of the Cross Validation score and comment how the CV score changes with lambda must be presented.

Using again the glmnet package, a cross-validation of the LASSO model can be easy calculated and plotted. It can be seen that only 13 variables are selected (see figure 7).

```
  (Intercept)     channel14     channel15     channel16     channel17     channel18     channel39
2.430320e-15 -1.602254e+00 -5.920294e-01 -4.978047e-01 -3.471019e-01 -1.377282e-01  5.881677e-03
   channel40     channel41     channel49     channel50     channel51     channel52     channel53
2.533079e-02  5.714424e+00 -2.466071e-05 -4.082507e-03 -8.735307e-01 -1.245545e+00 -1.069154e-03
>
```

Figure 7: Coefficients of the variables chosen using LASSO

On figure 8, it can be seen how the bigger the lambda gets, the more parameters are shrinked and the less complex the model is. For instance, when lambda equals 2 only one variable is selected and so different than 0. On the other hand, the MSE can be seen also evaluated as increasing as lambda gets bigger. Inductively, it can be understood that important variables are being eliminated from the model when lambda is increased. For that reason, the MSE starts increasing as the model is less fitted, or maybe, less over-fitted. For answering so, different predictions shall be performed to address which is the best value of lambda .

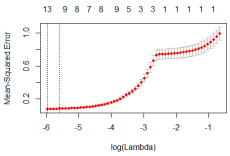


Figure 8: Plot of the change of the variables selected and and the MSE in terms of λ

Assignment 2.8

In this section results must be compared from section 4 and 7. The number of variables selected for the StepAIC was 63 whereas in LASSO it was just 13 as maximum. LASSO works properly for linear models but it can be difficult assessing which is the appropriate value of lambda that should be used. As far as I am concerned, the StepAIC model just evaluates if by discarding some variables if the MSE of the model gets lower and the adjusted R squared values remains constant, and if so, it continues until it reaches an optimal point.

Appendix

Assignment 1

```
##R script activity 1
data(swiss)


##FUnction that gives all the combinatorial of the data entered
all_beta_subsets<-function(xvar,n){

  binaryList <- rep(list(0:1), 5)

  exgrid<-expand.grid(binaryList)

  expand<-rowSums(exgrid)!=0 # Eliminating the line of all 0
  es<-exgrid[expand,]
  logisk<-es==1
  namn<-colnames(xvar)
  betasubsets<-list()
  for (i in 1:nrow(logisk)){
    betasubsets[i]<-paste( namn[logisk[i,]], collapse= " + ")

  }
  res<- list()
  res["betasubsets"]]<-betasubsets
  res["number of variables per model"]]<- rowSums(es)
  return(res)
}


## Function that includes y a list
includeNameInList<-function(name, lista){
  funct_list<-lista
  for (i in 1:length(funct_list)){
    funct_list[[i]]<- paste(name, funct_list[[i]], sep = " + ")
  }

  return(funct_list)
}

##It random splits data
random_split_data<- function(data, Nfolds){

  set.seed(12345)
  n <- dim(data)[1]
  sample_numbers<-sample(1:n)

  ##indexes rows for train
  randomsplitvector<-split(sample_numbers, ceiling(seq_along(1:n)/(n/Nfolds)))

  # indexes rows complementary of train
  randomsplitvector_compl<- list()
  for(i in 1:Nfolds){
    randomsplitvector_compl[[i]]<-
      sample_numbers[!sample_numbers %in% randomsplitvector[[i]]]

  }
```

```
  res<-list()
  res["train"]<-randomsplitvector_compl
  res["test"]<- randomsplitvector
  return(res)

}

random_split_data(data = swiss, Nfolds = 5)




lmFold <- function(x=swiss[,2:6], y=swiss[,1], Nfolds=5){

    set.seed(12345)
    data<- cbind(y=y,x)
    ##All combinations for data
    mybeta_subsets<- all_beta_subsets(x,Nfolds)$betasubsets
    vector_comb_with_pred<- includeNameInList("y", mybeta_subsets)

    #Setting differents numbers using sample to partition data in an outside functio

    randomsplitvector_train<- random_split_data(data = data, Nfolds = 5)$train
    randomsplitvector_test<- random_split_data(data = data, Nfolds = 5)$test
    ###Partitioning data with its real data
    datapartitioned_train<-list()
    datapartitioned_test<-list()
    for(i in 1:Nfolds){

      datapartitioned_train[[i]]<- data[randomsplitvector_train[[i]],]

    }


    for(i in 1:Nfolds){

      datapartitioned_test[[i]]<- data[randomsplitvector_test[[i]],]

    }

    res <- list()
     ## getting the string to pass in the model matrix as part of the data model
      for( j in 1:length(vector_comb_with_pred)){
        str=strsplit(vector_comb_with_pred[[j]], split = " [+] ")

        for(i in 1:Nfolds){
          for(name in str){
          #Intialization of matrixes for train and test

        x_matrix_train <- model.matrix(y~. - y, datapartitioned_train[[i]][name])
        y_vector_train <- datapartitioned_train[[i]][[1]]
        x_matrix_test <- model.matrix(y~. - y, datapartitioned_test[[i]][name])
        y_vector_test <- datapartitioned_test[[i]][[1]]
          }

        #Calculations of the coefficients
        res["coefficients_train"]] <-     solve((t(x_matrix_train)%*%x_matrix_train)
```

```
        res[["fitted.values"]] <- x_matrix_test%*%res[["coefficients_train"]]
        res[["residuals"]] <- y_vector_test - res[["fitted.values"]]
        res[["sum_of_residuals_squared"]][i]<-sum(res[["residuals"]]**2)



        #Transform matrix object to vector format
        res[["coefficients_train"]] <- as.vector(res[["coefficients_train"]])
        names(res[["coefficients_train"]]) <- colnames(x_matrix_train)

        res[["fitted.values"]] <- as.vector(res[["fitted.values"]])

        res[["residuals"]] <- as.vector(res[["residuals"]])

        res[["sum_of_residuals_squared"]][i]<-
        as.vector(res[["sum_of_residuals_squared"]][i])



        }
        ##Calculation of the MSE
        res[["MSE"]][j]<- sum(res[["sum_of_residuals_squared"]])/dim(data)[1]
      }
      #Class assignment
    class(res) <- "lmFold"

    #Output

    return(res[["MSE"]])
}

set.seed(12345)



##2

#Storing result
result<-lmFold(x = swiss[,2:6],y= swiss[,1], Nfolds = 5)
##storing numbers of variables in the combinatorial analysis
Number_Variables_per_model <-
  all_beta_subsets(swiss[,2:6], 5)$`number of variables per model`
##Plotting
library(ggplot2)
myframe<- data.frame(Number_Variables_per_model, result)
ggplot2::ggplot(myframe) +
        geom_point(aes(x = Number_Variables_per_model, y = result))+
        ylab("CV_score")

indexminmse<-which.min(myframe$result)
minmse<-min(myframe$result)
regressionresultmse<-
all_beta_subsets(swiss[,2:6], 5)$betasubsets[indexminmse]
```

```
##Results of the best model with the variables
minresultmse<-data.frame(indexminmse, minmse, regressionresultmse)
```

Assignment 2

```
############ Assignment 2
mydir<-
setwd("C:/Users/M/Desktop/Statistics and Data Mining Master/Semester 1
      /Second part of the semester/intro to machine learning/lab2")

mydir
library(readxl)

mydata<- read_excel("tecator.xlsx")

# 1)PLot moisture against protein
myPlot<- function(x, y){
  datfram<- data.frame(x = x,y = y)
  library(ggplot2)
  ggplot2::ggplot(datfram)+
            geom_point(aes(x = x, y = y))+
            xlab(deparse(substitute(x)))+ ylab(deparse(substitute(y)))+
            theme_dark()
  }

myPlot(x = mydata$Moisture, y = mydata$Protein)


# It looks like a linear regression works properly


##2 &3 )

##Partitioning data
n = dim(mydata)[1]
set.seed(12345)
id = sample(1:n, floor(n*0.5))
train = mydata[id,]
test= mydata[-id,]

## Setting the model for different lm regressions
Protein <- list()
Protein[["Protein1"]]<-lm(Moisture~Protein, data = train)
Protein[["Protein2"]]<-lm(Moisture~Protein+ I(Protein**2), data = train)
Protein[["Protein3"]]<-
  lm(Moisture~Protein+ I(Protein**2)+I(Protein**3), data = train)
Protein[["Protein4"]]<-
  lm(Moisture~Protein+ I(Protein**2)+I(Protein**3)+I(Protein**4), data = train)
Protein[["Protein5"]]<-lm(Moisture~Protein+ I(Protein**2)+I(Protein**3)+
                            I(Protein**4)+I(Protein**5), data = train)
Protein[["Protein6"]]<-lm(Moisture~Protein+ I(Protein**2)+I(Protein**3)+
                          I(Protein**4)+I(Protein**5)+I(Protein**6), data = train)

#Creating predctions for my  test data
myprediction_test<-list()
```

```
for(i in 1:length(Protein)){
  myprediction_test[[i]]<- predict(Protein[[i]], newdata= test)
}

##Creating a data frame of all the test predictions
#data to pass to calculate the MSE
myprediction_test<- as.data.frame(myprediction_test)
colnames(myprediction_test)<-c(1:6)

#Creating predctions for my train data
myprediction_train<-list()
for(i in 1:length(Protein)){
  myprediction_train[[i]]<- predict(Protein[[i]], newdata= train)
}


##Creating a data frame of all the train predictions
#data to pass to calculate the MSE
myprediction_train<- as.data.frame(myprediction_train)
colnames(myprediction_train)<-c(1:6)



#install.packages("hydroGOF")
library(hydroGOF)
## Getting the different MSE values for both train and test
msevector_test<-integer(length(myprediction_test))

for( i in 1:length(myprediction_test)){
msevector_test[i] <- mse(myprediction_test[[i]], test$Moisture)
}

msevector_train<-integer(length(myprediction_train))

for( i in 1:length(myprediction_train)){
  msevector_train[i] <- mse(myprediction_train[[i]], train$Moisture)
}

# plotting both MSE vectors

myplot2<- function(train,test){
  thedataframe <- data.frame(x = 1:length(train),train = train, test = test)
  plot<-ggplot2::ggplot(thedataframe)+
        geom_line(aes(x = x, y = train, color = "train"))+
        geom_line(aes(x = x, y = test, color = "␣test"))+
        ylab("values␣of␣MSE")+
        theme_bw()
  return(plot)
}

myplot2(train =msevector_train, test = msevector_test)


##The best predictor for this case is the base one

##4
library(MASS)
fitlm <- lm(Fat~.- Moisture - Protein - Sample, data = mydata)
```

```
stepaicResultlm<- stepAIC(fitlm, direction ="both")

coefficientslm<- stepaicResultlm$coefficients #Sroring lmcoefficients from StepAIC
length(coefficientslm) ## 64 variables were selected

stepaicResultlm$fitted.values

myPlot(x = mydata$Fat, y  = stepaicResultlm$fitted.values)



##5 & 6
#install.packages("glmnet")
library(glmnet)

## normalizing data and using the data to regress on 0 = Ridge, 1 = LASSO
covariates<- scale(mydata[,2:101])
response <- scale(mydata$Fat)
fitridge <-
  glmnet(x = as.matrix(covariates), y = response, alpha = 0, family = "gaussian")
plot(fitridge, xvar = "lambda", label = TRUE)

fitlasso <-
  glmnet(x = as.matrix(covariates), y = response, alpha = 1, family = "gaussian")
plot(fitlasso, xvar = "lambda", label = TRUE)

##7
##Using crossvalidation formulas in glmnet and plotting
bestfitlasso <-
  cv.glmnet(x = as.matrix(covariates), y = response, alpha = 1, family = "gaussian")
coefbestfitlasso <- coef(bestfitlasso, s= "lambda.min", extract = TRUE)
coefbestfitlasso_result<-coefbestfitlasso[coefbestfitlasso[,1] !=0,]
length(coefbestfitlasso_result)#14- 1 as intercept
plot(bestfitlasso)
```

732A95: Lab 3
Introduction to Machine Learning

Carles Sans Fuentes

November 25, 2016

Assignment 1

On assignment 1 it has been asked to use the data file australian-crabs.csv contains which contains measurements of various crabs, such as Frontal lobe, Rear width and others to implement linear discriminant analysis.

Assignment 1.1-1.2

In this section the australian-crabs.csv must be used to make a scatterplot of carapace length (CL) versus rear width (RW) where observations are colored by Sex.

From Figure 1 below, it can be seen that there exists two different and separated linear regressions that can be modeled separately by Sex (Female and Male) since there are not many points points of conflict between them because of its low variance on the data and the clear separation of the regressions between themselves. for that reason, a linear discriminant analysis regression line can be implemented.

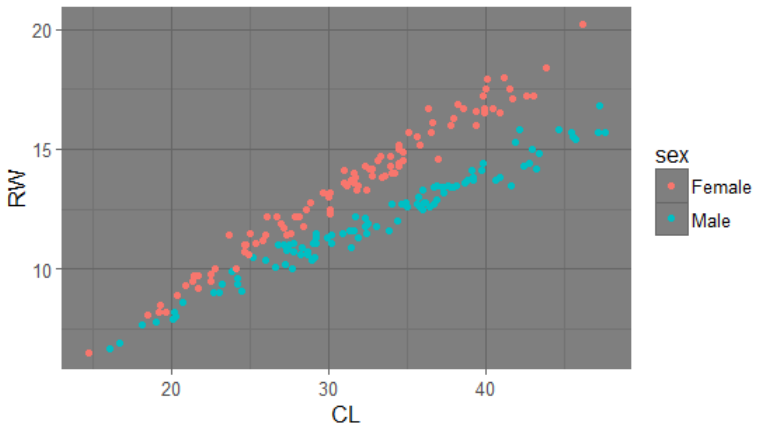


Figure 1: Plot RW vs CL colored by sex

Assignment 1.2

In this section a Linear discriminant algorithm must be implemented. For doing so, I first separate the data inside a function by Male and Female, calculating for each regressions its means, covariances and joint covariances. Using the formulas provided in the slides, the coefficients for each regression are calculated having as a discriminant function

$$w_0 + w_1 * CL + w_2 * RW$$

.

By equalizing both discriminant function, we get that

$$w_0^* + w_1^* * CL + w_2^* * RW = 0,$$

then, since we want to expressed as a function of RW:

$$RW = (-w_1^* * CL + W_0^*)/w_2^*,$$

having as intercept

$$W_0^*/w_2^*$$

and as slope

$$-w_1^* * CL/w_2^*$$

In the following figure (2 below), it can be seen the different coefficients for the discrimination function and the boundary decision.

```
$`Discrimination function`
$`Discrimination function`$male
[1] "male ~ -12.5634175116131 + -0.213814409833425 * CL + 2.56585136276431* RW "

$`Discrimination function`$female
[1] "female ~ -22.4287693550259 + -2.16131846691586 * CL + 8.24869811458541* RW  "


$`decision boundary`
intercept     slope
1.7359877 0.3426987
```

Figure 2: Information about the LDA implemented model

From the figure above, it can be seen that the decision boundary is

$$RW = 1.7359877 + 0.3426987 * CL$$

and the discrimination rates can be checked in the Figure above.

Here on Figure 3, it can be seen the original plot with the boundary decision line, which clearly is seen as an appropriate model. By plotting, it can be seen that the missclassification rate seems to be really low, being a quite accurate model for the data provided.

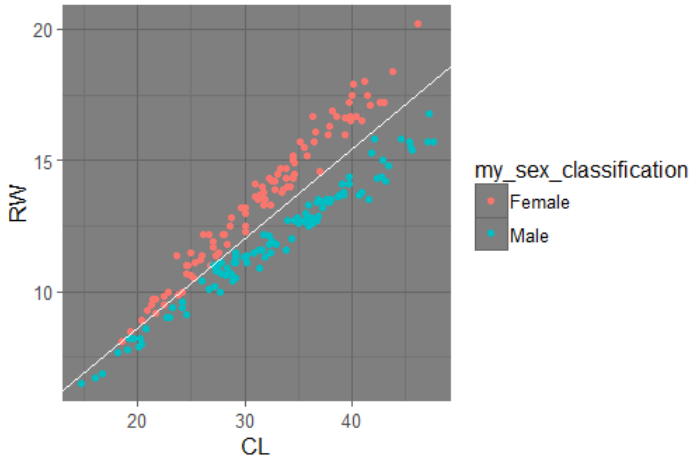


Figure 3: Predicted classification of data with the decision boundary of the model

Assignment 1.4

On assignment 1.4, it is asked to plot similar kind of classification by logistic regression using the glm() function, plotting the classified data as well as presenting the equation of the decision boundary (see figure 4 below). From it, a similar coefficients are obtained, being

$$RW = 1.083790477 + 0.368575779 * CL$$

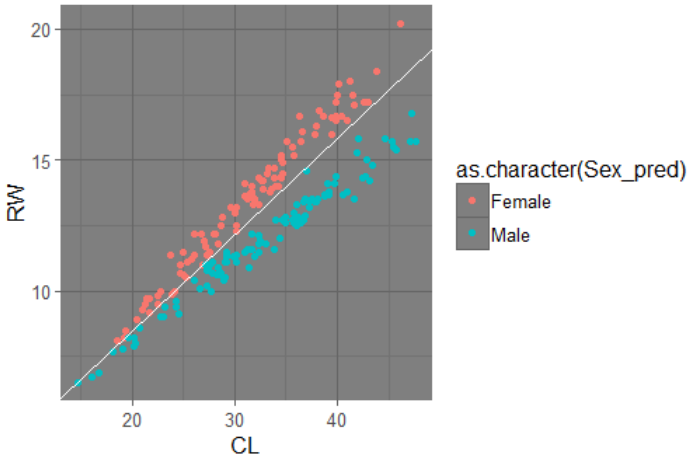


Figure 4: Predicted classification of data with the decision boundary of the R function model

Also, it can be seen that this model is also good, having a similar missclassification rate as the model implemented.

Assignment 2

In this assignment, it is read the *creditscoring.csv* file which contains data retrieved from a database in a private enterprise containing each row information about one customer indicating in one variable (good/bad) how the customers have managed their loans.

Assignment 2.1-2.2

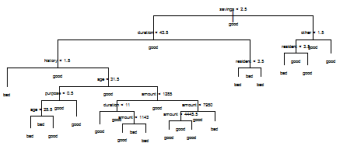
In this section, data has been divided into training/ validation/ test as 50%/25%/25% respectively and it has been fitted a tree model on train data for train and test prediction evaluating by "gini" and "deviance" as measures of impurity. The measure providing better classification rates is "deviance", as it can be seen just below when evaluating missclassification rates:

```
1
2           deviance  gini
3 missclass_train     0.212 0.238
4 missclass_test      0.268 0.372
```

The missclassification rates using deviance for training has been 0.212 for itself and 0,268 when predicted for test, whereas when using gini the rates are 0.238 for train and 0.372 for test.

Assignment 2.3

In this section, it is asked to use the training and validation sets to choose the optimal tree depth. The train decision tree from deviance is presented below in Figure 5.



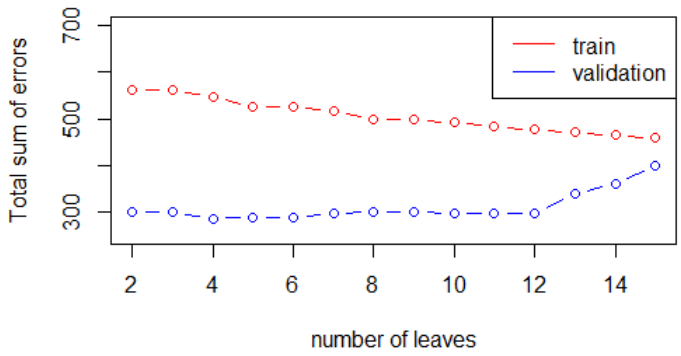


Figure 6: Rplot evolution dependence deviances between train and validation data

Figure 6 above shows the relation between the total sum of the residuals with the number variables in the model. The blue line stands for the validation score and the red one for the train score. The difference in the residuals seen is due to the fact that the train has 500 observations whereas validation only has 250 (as stated, the y-axis is the total sum of residuals, not the mean of it). Nevertheless, as a matter of interpretation, we can see that the more complex our model gets by having more number of prunes, the lower the error it gets for the trained data and the bigger for the validation data. Since it has been modeled on train data, we want to minimize the validation error sum, and so it is done when prunes (stated in the graphic as leaves) equals to 4, which values can be seen in the following Figure 6 below.

The tree depth is two since there are two extra decisions after the top one. The variables used on the tree are savings, duration and history. The meaning behind the tree stands for the following explanation: those who had more than 2.5 units of savings were directly classified as good, whereas the others where checked then also the variable of duration. If that variable was larger than 43.5, they were immediately classified as bad, whereas if so, the history variable was checked. If the latter history variables was registered as being larger than 1.5, then you were classified as good, else, you were classified as bad.

All in all, to be granted a loan, one must have more than 2.5 of savings, else, one must have duration lower than 43.5 and history bigger than 1.5.

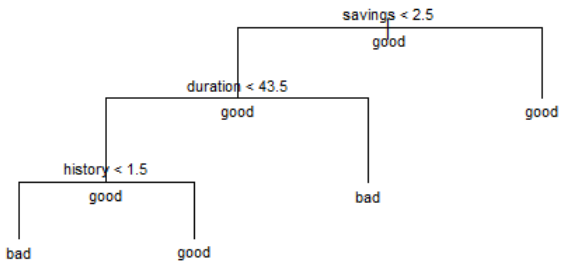


Figure 7: Tree plot of for leaves equals to four

Then, this model has been fitted and predicted for the test data, giving the following confusion matrix showed below with a missclassification rate of 0.256.

```
1
2     predicted
3 True   bad good
4   bad   18   58
5   good   6  168
```

Assignment 2.4

In this section training data must be used to perform classification using Naive Bayes and the confusion matrices and the missclassification rates for the training and for the test data must be reported. For training data and test data, the confusion matrix are the following ones in the list below. It can be seen that the missclassification rates are 0.3 for train and 0.316 for test. Comparing it to the previous step, it can be observed that the missclassification rate for the naive Bayes is larger than the tree classification, leading to worse prediction results.

```
1 > train_notweighted_table
2       predicted
3 true   bad good
4   bad   95   52
5   good  98  255
6
7 > test_notweighted_table
8       predicted
9 true   bad good
10  bad   46   30
11  good  49  125
```

Assignment 2.5

In this section, it has been told to minimize the missclassification rate by the loss matrix defined in the activity. For doing so, the predict for the naive train has been used but by getting the probability values of good and bad (using type = "raw"). To make our decision on the probability of good and bad based on the Loss matrix, the following formula has been used such tt an observation is classified as good if

$$P(C_{good})/P(C_{bad}) > L_{12}/L_{21}$$

By doing this, it can be seen that the missclassification rates are much higher (being 0.546 for the train and 0.508 for the test) , but the number of missclassification that we had in the loss matrix with higher values penalty has diminished, because its cost of it has been increased. When no loss matrix is given, all weights are the same for the ones missclassified so that an observation is classified as bad if

$$P(C_{bad}) * Cost_{L_{bad}} > P(C_{good}) * Cost_{L_{good}},$$

being $Cost_{Lbad} = Cost_{Lgood}$, then the expression is simplified to classifying as bad if

$$P(C_{bad}) > P(C_{good})$$

```
1 > train_weigthed_confusionmat
2       predicted
3 true   bad good
4   bad  137   10
5   good 263   90
6 > test_weigthed_confusionmat
7       predicted
8 true   bad good
9   bad   71    5
10  good 122   52
```

Appendix

Assignment 1

```
##R script activity 1



#Assignment 1

#install.packages("readr")
library(readr)
data <- read.csv("C:/Users/M/Desktop/Statistics and Data Mining Master/Semester 1/Second part
    of the semester/intro to machine learning/lab4/australian-crabs.csv")

library(ggplot2)
ggplot2::ggplot(data)+
          geom_point(aes(x=CL ,y=RW, color = sex))+
          theme_dark()




##Assignment 1.1
#PLot length (CL) versus rear width (RW) where observations are colored by Sex


#ASSIGNMENT 1.2
## It is necessary that the data provider has as name CL, RW and sex as colnames of data
## sex class must be either Male or Female

RegressionClassifier<- function(mydata=data){

   ##Creating data frame for the data interested in
  X <- data.frame(mydata[["sex"]], mydata[["CL"]], mydata[["RW"]])
  colnames(X)<- c("Y", "CL", "RW")

  ## splitting the data from by male and Female
  X_separated<- split(X, X$Y)
  X_male = split(X, X$Y)$Male ##Male data
  u_male<- c(mean(X_male$CL),mean(X_male$RW)) #mean Male
  S_male=cov(as.matrix(X_male[,2:length(X_male)]))*dim(X_male)[1] #cov of male

  X_female = split(X, X$Y)$Female #Female data
  u_female<- c(mean(X_female$CL),mean(X_female$RW)) #mean female
  S_female<- cov(as.matrix(X_female[,2:length(X_female)]))*dim(X_female)[1] # covariance female


  S=(S_male + S_female)/dim(X)[1] #covariance matrix of the female and male

  ## Values of w0 and w1 for Male and female, being w0 one number and w1 two
  w0_male = -1/2*t(u_male)%*%solve(S)%*%u_male + log(dim(X_male)[1]/dim(X)[1])
  w1_male = solve(S)%*%u_male

  w0_female = -1/2*t(u_female)%*%solve(S)%*%u_female+log(dim(X_female)[1]/dim(X)[1])
  w1_female = solve(S)%*%u_female
  #With the previously calculated numbers, it can be calculated my discrimination rates

  ##regression rest between both regressions
  joint_w0 <- w0_female-w0_male
  joint_w1 <- w1_female-w1_male

  ##Desicion boundary information
  slope = -joint_w1[1]/joint_w1[2]
  intercept = - joint_w0/joint_w1[2]

  ##my classification of outcomes
  my_sex_classification <-integer(length(mydata$CL))
  myframe_mysex_classification<- data.frame(Real_sex = as.character(mydata$sex),
                                            my_sex_classification= my_sex_classification,
                                            CL = as.numeric(mydata$CL),
                                            RW = as.numeric(mydata$RW))



  for(i in 1:length(mydata$CL)){
    #if the number obtained by the regression is bigger then Male

    if((intercept+slope*mydata[i,6])>mydata[i,5]){
      myframe_mysex_classification[i,2]<- "Male"

    }
    ##else female
    else{
      myframe_mysex_classification[i,2]<- "Female"
```

```
81
82
83      }
84    }
85    ##Initializing a list with outputs
86    res<- list()
87    res[["matrix classification"]]<- myframe_mysex_classification
88    res[["Confusion_matrix"]]<- table(True =myframe_mysex_classification[,1], Predicted =myframe_
          mysex_classification[,2])
89    res[["missclassification rate"]]<- (res[["Confusion_matrix"]][1,2]+res[["Confusion_matrix"
          ]][2,1])/sum(res[["Confusion_matrix"]])
90    res[["Discrimination function"]]<- list(
91                                          male = paste0("male ~ ", w0_male, " + ", w1_male[1]," *
                                                CL + ",w1_male[2],"* RW", sep = " "),
92                                          female = paste0("female ~ ", w0_female, " + ", w1_
                                              female[1]," * CL + ",w1_female[2],"* RW ", sep = "
                                              ")
93                                          )
94    res[["decision boundary"]]<- c(intercept= intercept, slope = slope)
95    res[["plot with real data"]] <- ggplot2::ggplot(mydata)+
96      geom_point(aes(x=CL ,y=RW, color = sex))+
97      geom_abline(slope = slope, intercept = intercept, color = "white")+
98      theme_dark()
99    res[["plot with my classification"]] <- ggplot2::ggplot(res[["matrix classification"]])+
100     geom_point(aes(x=CL ,y=RW, color = my_sex_classification))+
101     geom_abline(slope = slope, intercept = intercept, color = "white")+
102     theme_dark()
103
104   return(res)
105   }
106
107
108 a<-RegressionClassifier(mydata = data)
109
110 ##1.4
111
112 ###creating the regression
113 ##Female == 1, Male = 0
114 numericalsex<-ifelse(data$sex == "Female",1,0)
115
116 ##new data column binding this new variable
117 newdata<- cbind(data, numericalsex)
118
119 logisticR<-glm(formula = sex ~ CL+RW, family= "binomial", data = newdata)
120
121 w0<- logisticR$coefficients[1]
122 w1CL<- logisticR$coefficients[2]
123 w2RW<- logisticR$coefficients[3]
124
125 intercept<- as.numeric(-w0/w2RW)
126 slope<- as.numeric(-w1CL/w2RW)
127 ##getting the predicted RW
128 RW_pred<-round(predict.glm(logisticR, newdata = data, type = "response"))
129 RW_pred<-ifelse(RW_pred==0, "Female", "Male")
130
131 R_classification<- data.frame(Real_sex = as.character(data$sex),
132                               Sex_pred= RW_pred,
133                               CL = as.numeric(data$CL),
134                               RW = as.numeric(data$RW))
135
136
137 ##confusion matrix
138 conf<-table(True =R_classification$Real_sex, Predicted =R_classification$Sex_pred)
139 ##Missclassification rate
140 missrate<-(conf[1,2]+conf[2,1])/sum(conf)
141
142 Decision_boundary<- paste(intercept = intercept, slope = slope)
143 Decision_boundary
144 ##Prediction
145 library(ggplot2)
146 predicted_plot<-ggplot2::ggplot(R_classification)+
147   geom_point(aes(x=CL ,y=RW, color = as.character(Sex_pred)))+
148   geom_abline(slope =slope, intercept = intercept, color ="white")+
149   theme_dark()
150
151 predicted_plot
```

Assignment 2

```
1
2 ###Assignment 2
3
4 ###Assignment 2
```

```
5
6
7
8 ###1
9 ## read data as csv
10 data<-read.csv2("C:/Users/M/Desktop/Statistics and Data Mining Master/Semester 1/Second part of
       the semester/intro to machine learning/lab4/creditscoring.csv", sep =";")
11
12 n=dim(data)[1]
13 set.seed(12345)
14 id=sample(1:n, floor(n*0.5))
15 train=data[id,]
16
17 id1=setdiff(1:n, id)
18 set.seed(12345)
19 id2=sample(id1, floor(n*0.25))
20 validation=data[id2,]
21
22 id3=setdiff(id1,id2)
23 test=data[id3,]
24
25
26
27
28 ##2
29
30 # install.packages("tree")
31 library(tree)
32
33 ##Checking the different models for train, test
34 fit_train_dev <-tree(formula = good_bad~., data = train, split = "deviance")
35 fit_train_gin <- tree(formula = good_bad~., data = train, split = "gini")
36
37 train_predict_dev <- predict(fit_train_dev, newdata = train, type ="class")
38 train_predict_gin <- predict(fit_train_gin, newdata = train, type ="class")
39 test_predict_dev <- predict(fit_train_dev, newdata = test, type ="class")
40 test_predict_gin <- predict(fit_train_gin, newdata = test, type ="class")
41
42 train_dev<-table(train$good_bad, train_predict_dev) ##misclassification = 0.212
43 train_gin<-table(train$good_bad, train_predict_gin) ##misclassification = 0.238
44 test_dev<- table(test$good_bad, test_predict_dev) ##misclassification = 0.268
45 test_gin<- table(test$good_bad, test_predict_gin) ##misclassification = 0.372
46
47
48 missclass_train<- c(0.212, 0.238)
49 missclass_test<- c(0.268, 0.372)
50 myres<-rbind(missclass_train,missclass_test)
51 colnames(myres)<- c("deviance", "gini")
52 ##3
53 #install.packages("rpart.plot")
54 library("rpart.plot")
55 plot(fit_train_dev, uniform=TRUE,margin=0)
56 text(fit_train_dev, use.n=TRUE, all=TRUE, cex=.4)
57
58 rpart::
59 ##train
60 trainScore=rep(0,15)
61 validationScore=rep(0,15)
62
63 for(i in 2:15) {
64   prunedTree=prune.tree(fit_train_dev,best=i)
65   pred=predict(prunedTree, newdata=validation, type="tree")
66   trainScore[i]=deviance(prunedTree)
67   validationScore[i]=deviance(pred)
68 }
69 plot(2:15, trainScore[2:15], type="b", col="red", ylim=c(250,700),
70      ylab="Total sum of errors", xlab="number of leaves")
71 points(2:15, validationScore[2:15], type="b", col="blue")
72 legend("topright", c("train", "validation"), lty=c(1,1), col=c("red","blue"))
73
74
75 best<-prune.tree(fit_train_dev, best = 4) #chosen as best number i equal to 4
76
77 plot(best, uniform=TRUE,margin=0)
78 text(best, use.n=TRUE, all=TRUE, cex=.7)
79
80 best$terms
81 best$y
82
83 besttest<-predict(best, newdata = test, type = "class")
84 table(True =test$good_bad, predicted = besttest) ##missclassification rate = 0.256
85
86 ##4
87
88 library(MASS)
89 library(e1071)
90
```

```
fit_train_naive = naiveBayes(formula = good_bad~., data=train)

Yfit_train=predict(fit_train_naive, newdata=train)
Yfit_test=predict(fit_train_naive, newdata=test)

train_notweighted_table<-table(true = train$good_bad, predicted= Yfit_train)
train_notweighted_missrate<-
  (train_notweighted_table[1,2]+train_notweighted_table[2,1])/sum(train_notweighted_table)

test_notweighted_table<-table(true = test$good_bad, predicted= Yfit_test)
test_notweighted_missrate<-
  (test_notweighted_table[1,2]+test_notweighted_table[2,1])/sum(test_notweighted_table)


##5


##the decisionboundary is good versus bad = 1/10
decision_prob = 10


test_raw_predict=predict(fit_train_naive, newdata=test, type = "raw")
train_raw_predict=predict(fit_train_naive, newdata=train, type = "raw")


test_myframe<- data.frame(good = test_raw_predict[,"good"],
                      bad = test_raw_predict[,"bad"],
                      Y_predicted_weighted=NA,
                      True=test$good_bad)

for(i in 1:length(test$good_bad)){
  if((decision_prob*test_myframe[i,2]>test_myframe[i,1])){
    test_myframe[i,3]<- "bad"
  }
    else{test_myframe[i,3]<- "good"}
  }


train_myframe<- data.frame(good = train_raw_predict[,"good"],
                           bad = train_raw_predict[,"bad"],
                           Y_predicted_weighted=NA,
                           True=train$good_bad)

for(i in 1:length(train$good_bad)){
  if((decision_prob*train_myframe[i,2]>train_myframe[i,1])){
    train_myframe[i,3]<- "bad"
  }
  else{train_myframe[i,3]<- "good"}
}




test_weigthed_confusionmat<-
  table(true = test_myframe$True, predicted =test_myframe$Y_predicted_weighted)
test_weighted_missrate<-
  (test_weigthed_confusionmat[1,2]+test_weigthed_confusionmat[2,1])/
  sum(test_weigthed_confusionmat)

train_weigthed_confusionmat<-
  table(true = train_myframe$True, predicted =train_myframe$Y_predicted_weighted)
train_weighted_missrate<-
  (train_weigthed_confusionmat[1,2]+train_weigthed_confusionmat[2,1])/
  sum(train_weigthed_confusionmat)
```

732A95: Lab 3 block 2
Introduction to Machine Learning, Group 3

Assignment 1- Exercise 1

In this section data must be divided into training and test sets (70/30) without scaling. After that a nearest shrunken centroid classification of training data in which the threshold is chosen by cross-validation must be performed. To do so, the library pamr as well as its functions pamr.train and pamr.cv has been used to choose the best threshold. The best threshold according to lowest error is 1.3. A centroid plot is provided for the best threshold as well as a plot for the classification of the error and the threshold(see figure 1 and 2 below).

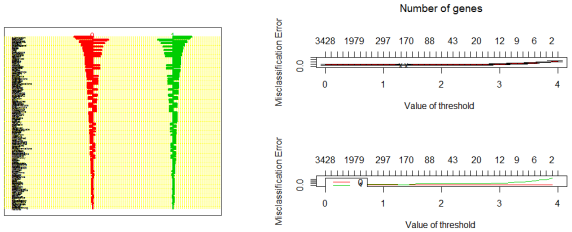


Figure 1: Plot for choosing best threshold considering error

Figure 2: Centroid plot for Conference with threshold equals to 1.3

The plot tries to capture significance of words, meaning that if those words are classified as 0 with big significance, it means that they are words that are clear for classifying an e-mail as non-conference e-mail. On the other side, those with 1 with high significance will be used to classify an e-mail as a Conference e-mail. For that reason, only those which seems to be quite clear for classifying should be used for that, otherwise the classification could be bias.

Also, a list of the 10 most contributing name features is provided below.

```
1
2
3
4
5
6
7
8  conference
9  dates
10 candidates
```

The number of features selected are 20. From the first 10, it can be seen that some of them are intuitively important, such as "call", "submission", "conference" or "position". Nevertheless, there are some which can seem not that evident at least for me for classification. Nevertheless, since this is chosen by modeling, it means it might be a good way to find patterns on e-mails that are not that intuitive. The test error is 0.0091. The confusion matrix is reported below.

```
1
2 0  1 Class Error rate
3 0 22  3       0.12000000
4 1  1 18       0.05263158
```

Exercise 2

In this section the test error must be calculated as well as he number of the contributing features for the methods:

- Elastic net with the binomial response and $\lambda = 0.5$ in which penalty is selected by the cross-validation. This is done using the glmnet package and using cross-validation (cv.glmnet). The lambda.min found is 0.165. After predicting on test, the found missclassification rate is 0.1. The confusion matrix is reported below.

```
1             Observed_values
2 elastic_pred  0  1
3             0 10  2
4             1  0  8
```

- Support vector machine with using the "vanilladot" kernel methods. This is done using the Kernlab() package, using the function ksvm and then predicting on test data. The missclassification rate got is 0.05.

  The confusion matrix is reported below.

```
          True
prediction  0  1
         0 10  1
         1  0  9
```

It can be seen that the best model for our data has been the linear kernel using the support vector machine method. A table with the reported values can be seen below.

```
> mymat
                          NSC Elasticnet ksvm
missclassification error 0.1        0.1 0.05
```

Exercise 3

In this section, the Benjamini-Hochberg method for the original data must be implemented.

The formula t.test() is using for computing the p-values and then the following formula from the Benjamini-Hochberg method has been used to rejected it or not getting α as 0.05. The null hypothesis is rejected if:

$$p_{(j)} < \alpha * j/M$$

where j is the position of the variable ordered by pvalue and M is the total number of variables.

In our case, only 39 variables has been rejected, that can be seen in the following list with its name variables and pvalues:

```
  myframe_res
           names      pvalues
1         papers 1.116910e-10
2     submission 7.949969e-10
3       position 8.219362e-09
4      published 1.835157e-07
5      important 3.040833e-07
6           call 3.983540e-07
7     conference 5.091970e-07
8     candidates 8.612259e-07
9          dates 1.398619e-06
10         paper 1.398619e-06
11        topics 5.068373e-06
12       limited 7.907976e-06
13     candidate 1.190607e-05
14        camera 2.099119e-05
15         ready 2.099119e-05
16       authors 2.154461e-05
17           phd 3.382671e-05
18      projects 3.499123e-05
19           org 3.742010e-05
20        chairs 5.860175e-05
21           due 6.488781e-05
22      original 6.488781e-05
23  notification 6.882210e-05
24        salary 7.971981e-05
25        record 9.090038e-05
26        skills 9.090038e-05
27          held 1.529174e-04
28          team 1.757570e-04
29         pages 2.007353e-04
30      workshop 2.007353e-04
31     committee 2.117020e-04
32   proceedings 2.117020e-04
33         apply 2.166414e-04
34        strong 2.246309e-04
35 international 2.295684e-04
36        degree 3.762328e-04
37     excellent 3.762328e-04
38          post 3.762328e-04
39     presented 3.765147e-04
```

Those selected are the most meaningful variables minimizing the FDR for α equal to 5%. These means, in other words, that given our statistical significant pvalues, we have not taken into account some of them subject to the formula above to minimize the probability of having called significant some that are not.

Assignment 2

The datafile used was **spambase.csv** which contains information of 4601 emails, classified as spam (class=1) or not spam (class=0). A budget online SVM was implemented on this data in which the inputs are different values of M and beta. The error rates were computed for each values of beta. The values of the inputs used are listed below:

- M=500, beta=0
- M=500, beta=-0.05
- M=20, beta=0
- M=20,beta=-0.05

The SVM first searched for the closest points which is also called as the support vectors. Once the model has found the closest point or support vector, it would draw a line connecting these points by subtracting vectors (point A - point B). The SVM then take the line that bisects and is perpendicular to the connecting and choose this line as the best separating line.

The error rates of the four different models were illustrated in figure 3. We can see that the SVM model with inputs (M=500, beta=0), represented by the purple line, outperformed the other three, as it has the lowest error rates in general. This suggests that this svm model found the best classification or separating line using the support vectors.

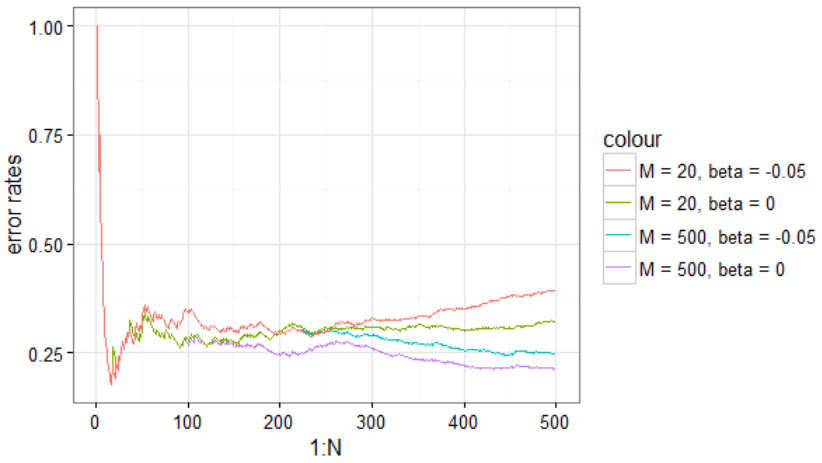


Figure 3: Error rates of different budget online svm models

The SVM with inputs (M=500, beta=0) gave better result than SVM with inputs (M=500, beta=-0.05) because taking lower values of beta makes the optimization choose a larger-margin hyperplane leading to more misclassified observations. It makes the cost of misclassification high. M decides how far the influence of the a single point reaches, with low values meaning far and high values meaning low.

It can be observed that SVM with inputs (M=20, beta=-0.05) gave smoother error rate plot than SVM with inputs (M=20, beta=0) which indicates that it is underfitted. The fact that it has very low beta and low M, the decision surface became smooth.

SVM with inputs (M=20, beta=0) is the slowest performing model because it is was very costly in computation. It tried to replace the original points by the support vectors found and then find the best separation for the data.

Appendix

Assignment 1

```
##R script activity 1
###Assignment 1

#importing data

data0<- read.csv2("C:/Users/M/Desktop/Statistics and Data Mining Master/Semester 1/Second part of the semester/intro to machine learning/lab3 block 2/data.csv", fileEncoding ="latin1"  )
#1

##dividing data
data<-data0
n<- dim(data)[1]

#.libPaths("C:/Users/Public/Temp/Installed")


#Sys.setenv(R_LIBS_USER = "C:/Users/Public/Temp/Installed")
data <- as.data.frame(data)

set.seed(12345)
id<- sample(1:n, 0.7*n)
train<- data[id,]
test<-data[-id,]
dim(test)
rownames(train) <- 1:nrow(train)




#install.packages("pamr")
library(pamr)

x <- t(train[,-ncol(train)])
y <- train$Conference

x_t <- t(test[,-ncol(test)])
y_t <- test$Conference
mydata <-list(x=x,y=as.factor(y),geneid=as.character(1:nrow(x)), genenames=rownames(x))
mydatatest <-list(x=x_t,y=as.factor(y_t),geneid=as.character(1:nrow(x_t)), genenames=rownames(x_t))

model <- pamr.train(mydata,threshold=seq(0,4, 0.1))


##cross-validation for the model
cvmodel<- pamr.cv(model,mydata)
print(cvmodel)

##chossing threshold which error is minimized
my_threshold<- cvmodel$threshold[which.min(cvmodel$error)]
pamr.plotcen(model, mydata, threshold=my_threshold)




#Necessary to expand my margins on the right side
pamr.plotcv(cvmodel)

bestmodel<- pamr.train(mydata,threshold=my_threshold)
mypred<-pamr.predict(fit = bestmodel, newx = as.matrix(test), threshold = my_threshold)
pamr.confusion(fit= model, threshold=my_threshold , extra=TRUE)

a<- pamr.listgenes(bestmodel,mydata,threshold= my_threshold)
cat( paste( colnames(data)[as.numeric(a[,1])][1:10], collapse='\n' ))


#2
library(glmnet)

#Checking best lambda by cv
mycv<- cv.glmnet(x = as.matrix(t(x)), y = as.factor(y), family = "binomial", alpha = 0.5)
plot(mycv)
lambda_min<- mycv$lambda.min

bestcv_model<- glmnet(x = as.matrix(t(x)), y = as.factor(y), family = "binomial", alpha = 0.5, lambda = lambda_min)
coef(bestcv_model)
elastic_pred<-predict(bestcv_model, newx = as.matrix(t(x_t)), type ="class")
table(elastic_pred= elastic_pred, Observed_values = test$Conference )

####2b
```

```r
#install.packages("kernlab")
library(kernlab)


res<-ksvm(Conference~., data = train, kernel = "vanilladot")
my_pred<- round(predict(res, test[,-ncol(test)]), digits =0)
table( prediction = my_pred, True = y_t )

#3
namesdata<-names(data)

pvalue<- integer(0)

names(data0)
for(i in 1:(length(namesdata)-1)){
  formula<-  as.formula(paste(namesdata[i],"~ Conference"))
  pvalue[i]<-t.test(formula = formula, data = data, alternative="two.sided")$p.value

}
ordered_pvalues<-pvalue[order(pvalue)]
plot(ordered_pvalues)

alpha <- 0.05
chosingvalues<- alpha*1:(length(namesdata)-1)/(length(namesdata)-1)
final<- ifelse(ordered_pvalues<chosingvalues, ordered_pvalues,NA)



myframe<- data.frame(names = namesdata[order(pvalue)], pvalues = ordered_pvalues, decision =
    final )

myframe_res<- myframe[complete.cases(myframe),1:2]
```

Assignment 2

```r

###Assignment 2
##Asignment 2

set.seed(1234567890)
spam <- read.csv2("C:/Users/M/Desktop/Statistics and Data Mining Master/Semester 1/Second part
    of the semester/intro to machine learning/lab3 block 2/spambase.csv")
ind <- sample(1:nrow(spam))
spam <- spam[ind,c(1:48,58)]

gaussian_h <- 1
beta  <-  c(0,-0.05,0,-0.05) # Your value here
  M   <-  c(500, 500, 20, 20) # Your value here
  N   <- 500 # number of training points



mySVMfunction <- function(M, beta){

  b<-0

  mydata <- model.matrix(Spam~., data = spam)
  mydata <- mydata[,-1] # taking out the last column of proof
  spam[,"Spam"]<-2*spam[ncol(spam)]-1 # last colum from spam or not by one or 0
  myspamvect <- spam[,all.vars(Spam~.)[1]]


##my function to calculate the euclidean distance between two vectors
  distancevectors<- function(vector1, vector2){
    if(length(vector1)==length(vector2)){
      myresult<- sqrt(sum((vector1-vector2)**2))
      return(myresult)
      }
    else{(stop("vectors are not of the same length"))
    }
  }

  #gaussian kernel function just to pass the eucliedean distance
  gaussian_k <- function(x, h = gaussian_h) { # Gaussian kernel h = 1
    gaussiandist<- exp(-(x**2/(2*h^2)))   #h = 2*var(x), where variance = 1
    return(gaussiandist)
    }

  ##calculating y
  SVM <- function(sv,i) { # SVM on point i with support vectors sv

    vector_dist<- as.matrix(dist(mydata[c(i,sv),]))[-1,1]#euclidean distance
```

```
    kernel_distance<-gaussian_k(x=vector_dist) #one vector distance point between both data
    y<-   1*myspamvect[sv]*kernel_distance+b #a = 1
    return(sum(y))

    }

  errors <- 1
  errorrate <- vector(length = N)
  errorrate[1] <- 1
  sv <- c(1)

  for(i in 2:N) {
    # Your code here
    y_i<-SVM(sv = sv, i = i)

    if(myspamvect[i]*y_i<0){##if t_i*y<= 0, then error
      errors<- errors+1


    if(myspamvect[i]*y_i<beta){##if t_i*y<= Beta, then
      sv[length(sv) + 1] <- i

    if(length(sv) > M){
      res_temp <- lapply(seq_along(1:length(sv)), function(j){
        svm_temp <- SVM(sv = sv, i = sv[j])
        itself <- 1*myspamvect[sv[j]]*
          gaussian_k(x = dist(mydata[c(sv[j],sv[j]),]))
        res_out <- myspamvect[sv[j]]*(svm_temp-itself)
        return(res_out)

      })
      sv <- sv[-which.max(res_temp)]
      }
        }
      }
    errorrate[i] <- errors / i
  }
  return(list(errorrate,sv))
}


res1 <- mySVMfunction(M = 500, beta =  0)
res2 <- mySVMfunction(M = 500, beta = -0.05)
res3 <- mySVMfunction(M = 20, beta = 0)
res4 <- mySVMfunction(M = 20, beta = -0.05)

sv1<-res1[[2]]
sv2<-res2[[2]]
sv3<-res3[[2]]
sv4<-res4[[2]]

res_1<-res1[[1]]
res_2<-res2[[1]]
res_3<-res3[[1]]
res_4<-res4[[1]]

res_1==res_2
myframe<- data.frame(res_1, res_2, res_3, res_4)
library(ggplot2)
ggplot2::ggplot(myframe)+
          geom_line(aes(x= 1:N, y = res_1, color = "M = 500, beta = 0"))+
          geom_line(aes(x= 1:N, y = res_2, color = "M = 500, beta = -0.05"))+
          geom_line(aes(x= 1:N, y = res_3, color = "M = 20, beta = 0"))+
          geom_line(aes(x= 1:N, y = res_4, color = "M = 20, beta = -0.05"))+
          ylab("error rates")+  theme_bw()



plot(res_1, x = 1:N, ylim=c(0,1), type="l", col = "red", xlab = "N",
      ylab = "Error Rates")
lines(res_2, x = 1:N, ylim=c(0.4,1), type="l", col = "blue")
lines(res_3, x = 1:N, ylim=c(0.4,1), type="l", col = "green")
lines(res_4, x = 1:N, ylim=c(0.4,1), type="l", col = "black")
 legend("topright", c("M = 500, beta = 0","M = 500, beta = -0.05",
                      "M = 20, beta = 0","M = 20, beta = -0.05"),
        lty = c(1,1),
        col = c("red","blue","green","black"))
```

732A95: Lab 4 block 1
Introduction to Machine Learning

Emma Wallentinsson, Zaida Liendeborg, Carles Sans

December 4, 2016

Assignment 1 - Uncertainty estimation

On assignment 1 we have been asked to use the State.csv file which contains per capita state and local public expenditures and associated state demographic and economic characteristics, as well as the variables:

- MET: Percentage of population living in standard metropolitan areas.
- EX: Per capita state and local public expenditures ($)

1.1

In this section data must be reordered according to the increasing value of MET and a plot of EX versus MET must be carried out.

From Figure 1 below, it can be seen that there is a non-linear relationship between the EX variable (per capita state and local expenditure) and MET (% of population living in metropolitan areas), because of the curve pattern in the data. A non-linear regression model could be appropriate to fit the data.

Given the high variance of our data distribution, a deterministic model (those models that does not explain the error of data) cannot be used. For that reason, a probabilistic model which tries to capture the variance of the model inside some confidence bands is suggested.

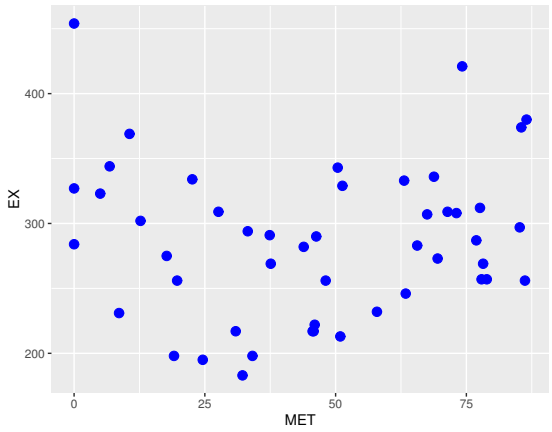


Figure 1: Scatter plot of EX vs MET.

1.2

In this section the whole data set ordered by increase in MET is going to be used to fit a regression tree model with target EX and feature MET in which the number of the leaves is selected by cross-validation using the whole data set.

To do so, the tree function from the library tree has been used having in each leave at least 8 observations (with minsize $\bar{8}$). To find the optimal tree size, the deviance measure is plotted against number of leaves from the cross validation in figure 2.

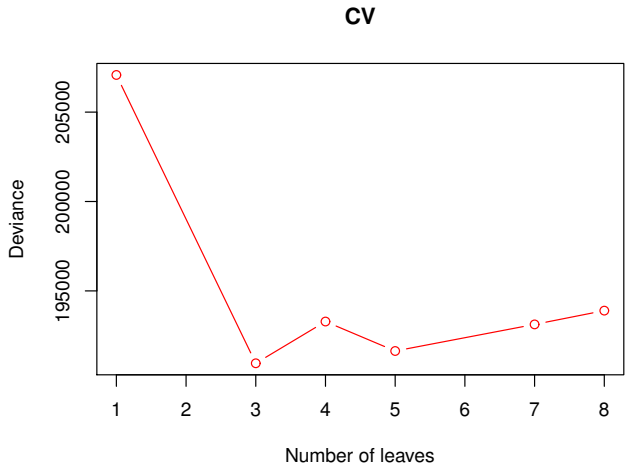


Figure 2: Deviance score from cross validation.

We want the lowest score on the deviance, the best tree is the one of size 3, and a plot of the optimal tree is provided below(see figure 3). It can also be seen on it the different variables used to split the data and the decision rules.

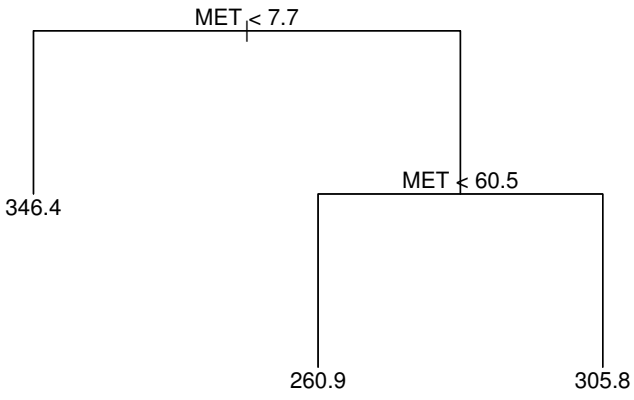


Figure 3: Regression tree with three leaves.

In the next step, the fitted values from the regression tree are plotted together with the original values from figure 1. The result is shown below in figure 4.

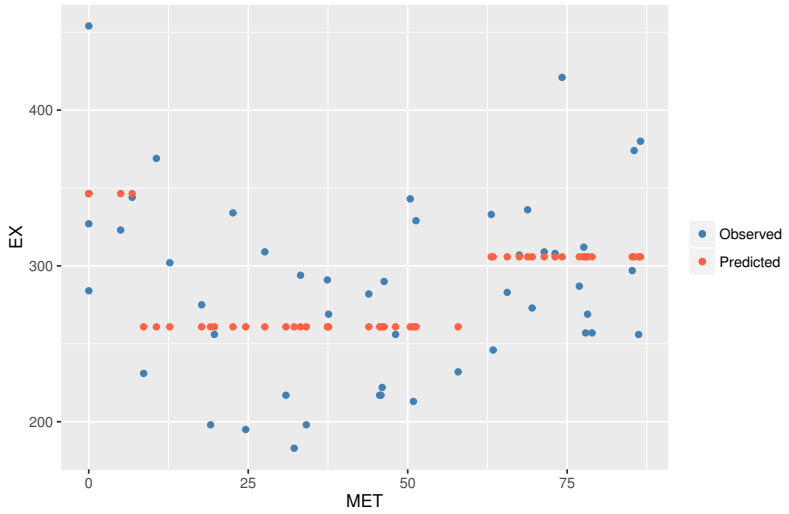


Figure 4: Observed and predicted values.

From figure 4, we see that the predicted values lies in straight lines between some intervals on the x-axis. This is because the predicted values are the mean value of the observations inside the terminal leaf. When we read the values from the tree in figure 3, it can be seen that when **MET** is below 7.7 units, **EX** the predicted value is 346. The same principle applies when **MET** is between 7.7 and 60.5 units, the predicted values of EX are then 260.9. When **MET** is higher than 60.5, the predicted values for **EX** are 308.8 units.

To analyze the model further, a histogram over the residuals is plotted in figure 5.

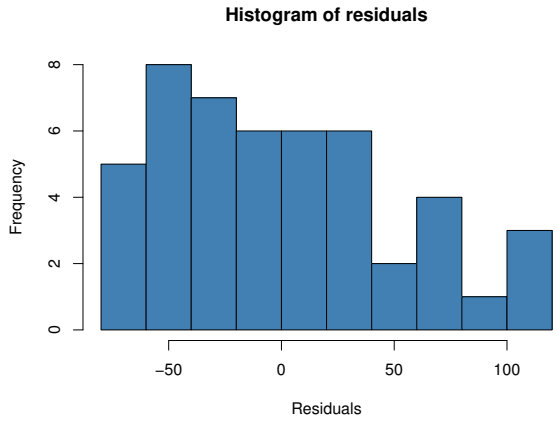


Figure 5: Histogram over residuals.

The residuals in figure 5 seems to follow an exponential distribution since there exists skewness to the right. Moreover, it can be seen that many observations have large residuals. It would be good to evaluate whether other models would give smaller residuals.

1.3

In this section the 95% confidence bands for the regression tree model from step 2 must be provided by using a non-parametric bootstrap.

Non-parametric bootstrap is used when the distribution and the variance of the response variable is not known. Then, by creating samples of the same length by sampling the data with replacement, its sample mean and its sample variance can be estimated.

In this case, the **boot** function has been used to get 1000 bootstrap samples. The **envelope** function was used to provide with the 95% confidence interval to analyze the uncertainty of the estimators. (see figure 6 below).

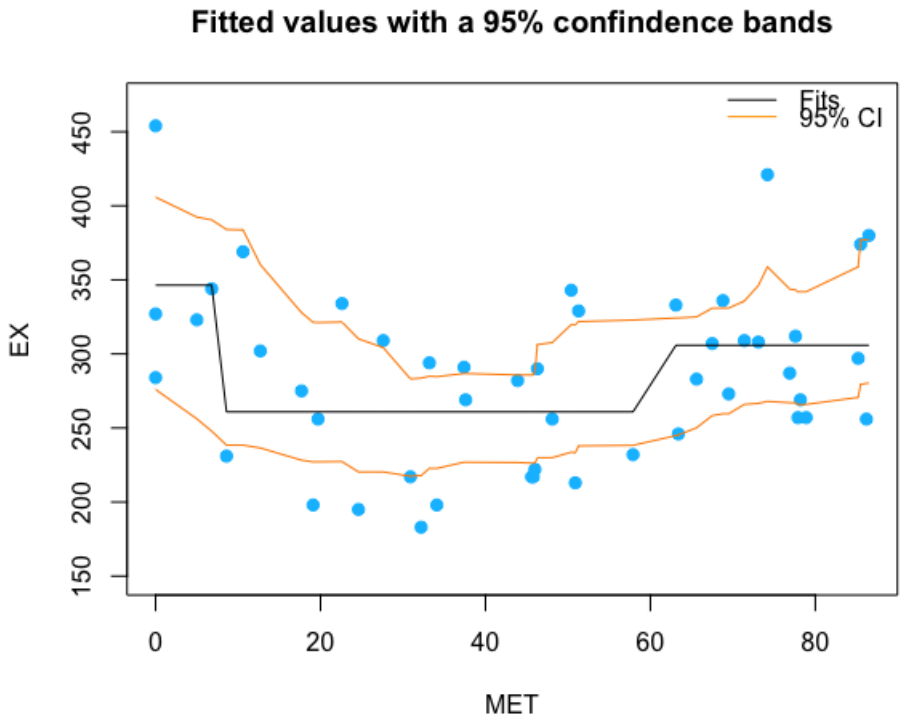


Figure 6: Observed and predicted values with 95% confidence bands.

In figure 6, the blue dots are observed values and the black line represents the fitted ones. As previously mentioned, the orange lines are the 95 % confidence interval calculated from 1000 non-parametric bootstrap samples. As seen in the plot, the bands are narrow and very rigid. The band is very bumpy due to the fact that different bootstrap samples gave different standard errors. Confidence intervals describe the uncertainty of the estimators on how well the true mean is determined. It gives a range for the expected value (mean) of the response variable EX for many different bootstrap samples of MET. Looking at the plot, it seems like that the result from the regression tree model is reliable.

1.4

In this section a parametric bootstrap is used to fit a 95 % prediction and confidence band from 10 000 bootstrap samples for the same model as in 1.3. It is assumed that data is $Y \sim N(\mu_i, \sigma^2)$, where μ_i is the expected value or the mean of the observations in the terminal leaves and σ^2 is the variance. The result is visualized in figure 7.

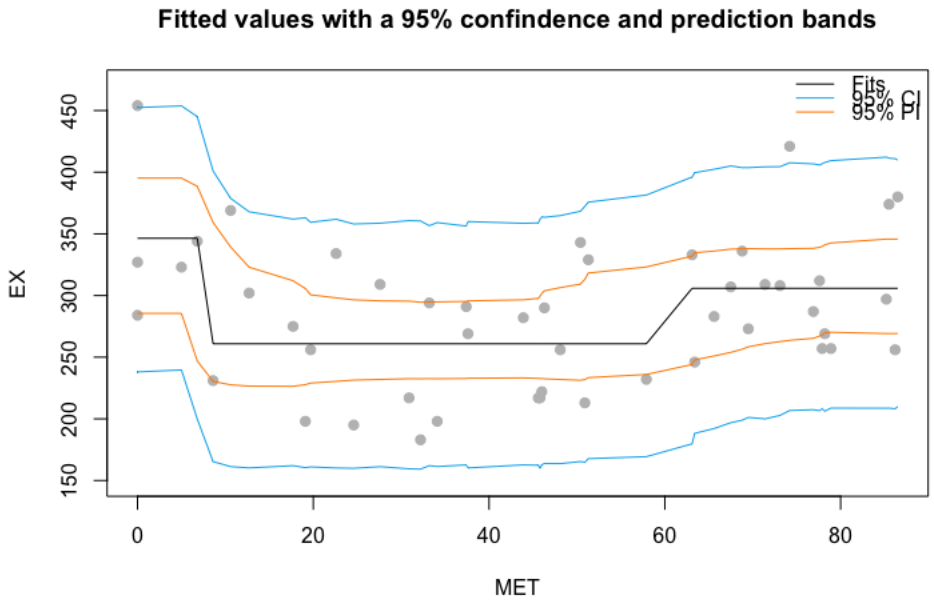


Figure 7

The confidence interval represented by the blue lines are much wider compared with the confidence interval obtained from the non-parametric bootstrapping. The prediction band which is represented by the orange lines is also very narrow compared to the confidence band. In theory, the prediction band should be wider than the confidence band because it accounts for both the uncertainty of the individual point estimates and the variability of the estimates around the predicted mean. For this reason, the result from non-parametric bootstrapping is not reliable.

Choosing 95% prediction interval is expected to give a 5% marginal error, so 5% of the data should be outside the prediction bands. In this case, there are more than 5% of the data that lies outside the prediction interval.

1.5

From the residuals in figure 5, it's concluded that they are not normally distributed but rather more exponentially distributed. The parametric bootstrap assumes that the response variable is normally distributed that is why we got a very high uncertainty on its estimates. It is therefore concluded that the non-parametric bootstrap gave better result and is more reliable.

Assignment 2 - Principal components

In this assignment, the data ***NIRspectra.xls*** is used. The data contains near-infrared spectra (features) which is to be used to predict the level of viscosity of the fuels.

2.1

A standard PCA is conducted for the features, and a scree plot is provided to see how much of the variation is explained by each of the principal components. Principal components is used mainly to reduced the amount of predictors but at the same time not dropping information from these predictors. Principal components projects the data into new direction, where variation of the data is as high as possible. The components are orthogonal and uncorrelated to each other. A scree

plot is provided below in figure 8 to see how much of the variation is captured by each principal component.

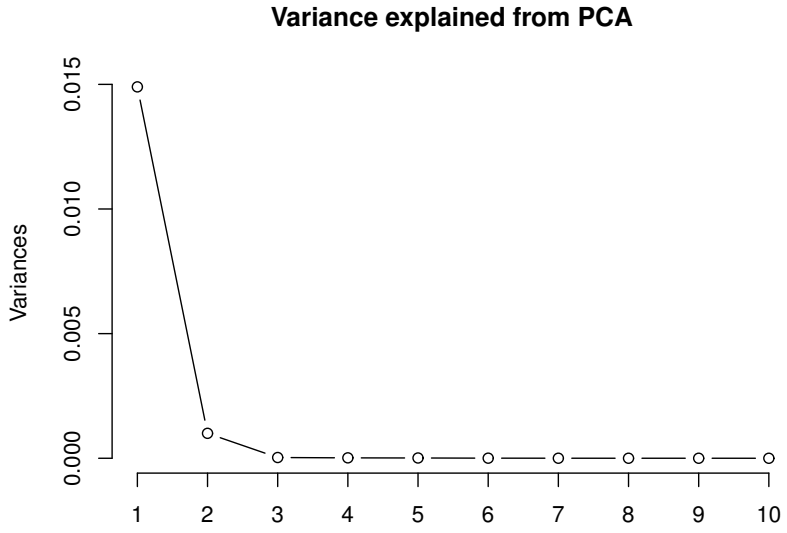


Figure 8: Screeplot of principal components.

From figure 8, we see that the first principal component explains a high amount of the variation in the data. To select the number of components which explains at least 99 % of the variation we use the summary output below (just showing PC1 - PC9).

```
Importance of components:
                          PC1     PC2      PC3      PC4      PC5      PC6     PC7       PC8
Standard deviation     0.1221 0.03162 0.005435 0.004011 0.003303 0.001985 0.00119 0.0007058
Proportion of Variance 0.9333 0.06263 0.001850 0.001010 0.000680 0.000250 0.00009 0.0000300
Cumulative Proportion  0.9333 0.99596 0.997810 0.998820 0.999500 0.999750 0.99983 0.9998700
```

From the output abowe, the first PCA explains 93 % of the variation, and together with PC2, these two components explains 99% of the variation. Therefore the first two components were selected for the analysis.

The principal components are expressed by the formula:

$$Z_1 = 0.1099X_{750} + 0.1097X_{752} + \ldots + 0.0829X_{998} + 0.0811X_{1000}$$
$$Z_2 = -0.0339X_{750} - 0.03423X_{752} + \ldots + 0.3435_{X998} + 0.3108X_{1000}$$

These two principal components are uncorrelated. PC1 projects the data where the variance is maximized. PC2 is the direction of the projected data which variation is maximized after the variation in PC1 was removed. The first principal components have positive loadings which can be interpreted as the weighted mean value of the viscosity level. The PC2 have both negative and positive loadings which can be interpreted as differences between the high and low levels of viscosity.

A plot of the calculated scores from PC1 and PC2 is also provided. These are calculated by substituting the values of the variables for every observation on the equation above. The corresponding score plot is shown below in figure 9.

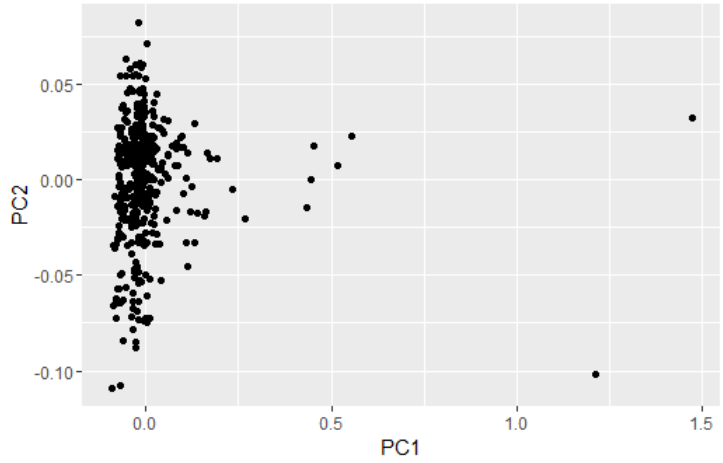
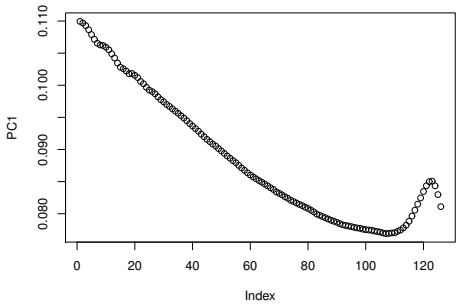


Figure 9: Scores of PC1 vs Pc2.

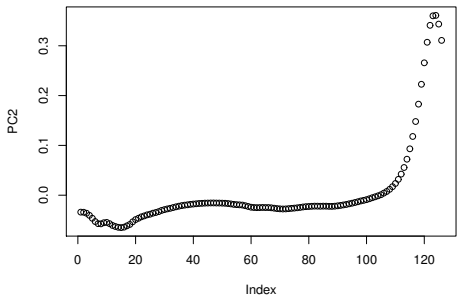
Since the data is centered, the mean level of viscosity is zero. Almost all the fuels have the same mean level of viscosity, as they scattered around zero. However, there are few fuels that deviates a lot from most of the others. These are the ones found to the right of the mean, having very high viscosity level. As per the PC2, fuels found in or near zero, do not differ or has very little difference on their viscosity level. Since, do not have more detailed information about the spectra, it is hard to interpret the dimension PC2.

2.2

Trace plots from PC1 and PC2 are presented below in figure 10a and 10b.



(a) Loadings of PC1.



(b) Loadings of PC2.

Figure 10: PCA loadings.

It can be seen that PC1 is explained similarly by all features, going smoothly down on the last ones. On the other hand, PC2 was mainly explained by the last features as almost all the feature got loadings close to zero except for the last 13 features.

```
> tail(PC2, 13)
      X976       X978       X980       X982       X984       X986       X988
0.1075628  0.1332918  0.1614746  0.1921053  0.2232881  0.2534943  0.2803218
      X990       X992       X994       X996       X998      X1000
0.3014615  0.3161506  0.3238344  0.3251783  0.3198321  0.3074829
```

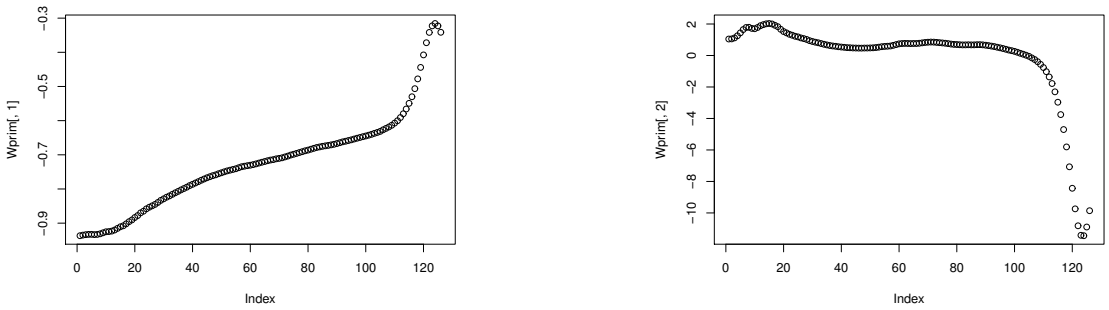
2.3

Now, an independent component analysis is performed (ICA), where the number of components is selected to be two from the previous findings in the PCA. When ICA is performed, the assumption is that the variables are non-normal and independent of each other.

The components are calculated by maximizing the independence, and this is done by maximizing the non normality.

a)

The *W'* is computed by multiplying the pre- whitening matrix *K*, which has projected the data into two principal components, with the estimated un-mixing matrix *W*. The trace plots from *W'* is shown below in figure 11a and 11b.



(a) Traceplot of W'1.

(b) Traceplot of W'2.

Figure 11: Traceplots of W'.

The plots in figure 11, is the independent components from the ICA. Compared to the trace plots from the PCA in figure 10a and 10b, we see that the first ICA component is not explained by the first features, more by the last. The second ICA component is explained by the first features, on the contrary to PC2.

b)

The plot of the scores from the ICA is presented below in figure 12.

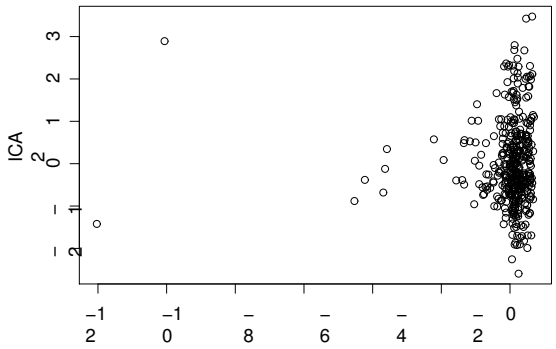


Figure 12: Scores from ICA.

From the score and the trace plots it can be seen that similar results obtained as in PCA. This time the first independent component explained by mainly the last features as in PCA whereas the second component explained by almost all the features. The score plot seems to be almost equal as the one in PCA but mirrored. For that, no new conclusions has been extracted from our data. There might be no hidden factors that affect the level of viscosity of the fuels.

2.4

A principal component regression is performed where the number of components is selected by cross validation. A PCR is a regression that uses the principal components to estimate the coefficients, where the principal components that explains the highest variance is used in the model.

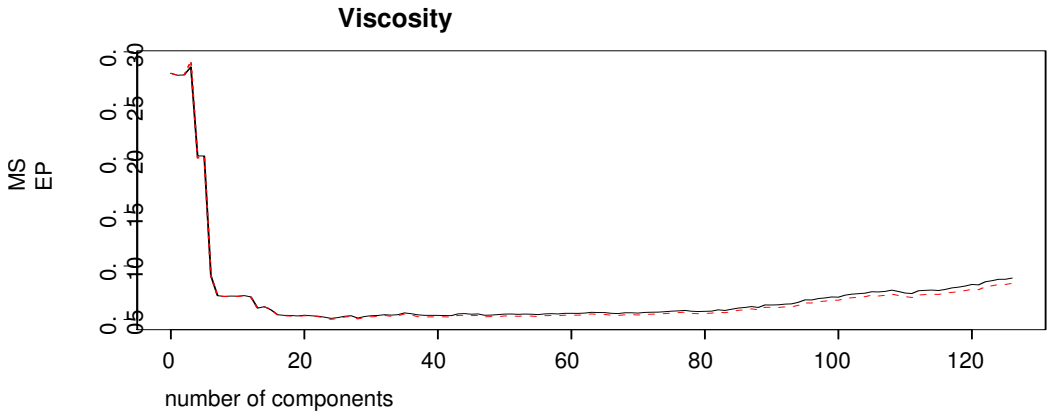


Figure 13: MSEP vs number of components.

In figure 13, the mean squared error of the predictions are plotted against the number of components. We want the lowest value on the squared error for our model and by looking at the graph, we could see the the lowest value is found after 20 components. Examining the cross-validation RMSEP from 20 components, we could see that model having 24 components got the lowest value.

```
1       20 comps  21 comps  22 comps  23 comps 24 comps
2 CV      0.2324    0.2312    0.2305    0.2284    0.2259
3 adjCV   0.2313    0.2303    0.2295    0.2273    0.2239
```

To strengthen the analysis above the PRESS value was also examined, 24 components gave the lowest PRESS value. PRESS value is a measure to express the prediction ability of a model. Therefore, it is reasonable to select 24 components in doing the PCR.

Appendix

Code for assignment 1

```
##Assignment 1
state<-read.csv2("state.csv", sep=";")

# 1.1
ord<-state[order(state$MET),] #order map MET

ggplot(ord,aes(MET,EX)) + geom_point(colour="blue", size=3)

#1.2


regmod<- tree(EX ~ MET, data=ord,control= tree.control(nobs=48,minsize=8))#tree reg
set.seed(12345)
krosstree<-cv.tree(regmod)## cross validation
#plotting deviance vs number of leaves
plot(krosstree$size, krosstree$dev, type="b", col="red", ylab="Deviance", xlab="Number of
    leaves", main="CV")

finalTree<-prune.tree(regmod, best=3) #pruning to 3 leaves
Yfit<-predict(finalTree, newdata=ord) #new predictions from best tree

#plot tree
plot(finalTree)
text(finalTree, pretty=0)

#plot observed and predicted values
ggplot() + geom_point(aes(y=ord$EX, x=ord$MET, color="Observed")) +
  geom_point(aes(y=Yfit, x=ord$MET, color="Predicted")) +
  scale_color_manual("", values=c("Observed"="steelblue", "Predicted"="tomato")) +
  labs(x="MET", y="EX")

#plot histogram
hist(residuals(finalTree), col="steelblue", main="Histogram of residuals", xlab="Residuals")


## ass 1.3
library(boot)

# computing bootstrap samples
f<- function(data, ind){
  set.seed(12345)
  data1<- data[ind,]# extract bootstrap sample
  res<- tree(formula= EX ~ MET, data=data1,
          control=tree.control(nobs=48, minsize = 8)) #fit linear model
  best<- prune.tree(res, best=3)
  pred<- predict(best,newdata=data)
  return(pred)
}

res_nonpar<- boot(state_sort, f, R=1000)

e<- envelope(res_nonpar) #compute confidence bands
plot(x=state_sort$MET, y=state_sort$EX, pch=19, col="deepskyblue",
     ylim=c(150,470), xlab="MET", ylab="EX",
     main="Fitted values with a 95% confindence bands")
points(state_sort$MET, st_tree_pred, type="l") #plot fitted line
#plot cofidence bands
points(state_sort$MET, e$point[2,], type="l", col="darkorange")
points(state_sort$MET, e$point[1,], type="l", col="darkorange")
legend("topright", legend=c("Fits", "95% CI"), lty=1,
       col=c("black", "orange"), bg="white", cex=1, bty="n")


# 1.4
# Prediction Interval
f_par<- function(data){
  model<- tree(formula= EX ~ MET, data=data,
               control=tree.control(nobs=48, minsize = 8)) #fit reg tree model
  best<- prune.tree(model, best=3)
  pred<- predict(best, newdata=data)
  n<- length(data$EX)
  pred_ran <- rnorm(n, pred, sd(residuals(best)))
  return(pred_ran)
}

mle<- best_tree
randgen<- function(data, mle) {
  data1<- data.frame(EX=data$EX, MET=data$MET)
  n<- length(data1$EX)
  #generate new EX
  data1$EX<- rnorm(n, predict(mle, newdata=data1), sd(residuals(mle)))
```

```
  return(data1)
}

set.seed(12345)
res_par<-boot(state_sort, statistic=f_par, R=10000,
         mle = best_tree , ran.gen=randgen, sim="parametric")

e_par<- envelope(res_par) #compute prediction bands
plot(x=state_sort$MET, y=state_sort$EX, pch=19, col="grey",
     ylim=c(150,470), xlab="MET", ylab="EX",
     main="Fitted values with a 95% confindence and prediction bands")
points(state_sort$MET, st_tree_pred, type="l") #plot fitted line
#plot cofidence bands
points(state_sort$MET, e_par$point[2,], type="l", col="deepskyblue2")
points(state_sort$MET, e_par$point[1,], type="l", col="deepskyblue2")
legend("topright", legend=c("Fits", "95% CI", "95% PI"), lty=1,
       col=c("black", "deepskyblue2", "darkorange"), bg="white", cex=1, bty="n")


# Confidence Interval

fpar_CI<- function(data){
  model<- tree(formula= EX ~ MET, data=data,
               control=tree.control(nobs=48, minsize = 8))
  best<- prune.tree(model, best=3)
  pred<- predict(best, newdata=data)
  return(pred)
}

set.seed(12345)
respar_CI<- boot(state_sort, statistic=fpar_CI, R=10000,
         mle=mle, ran.gen=randgen, sim="parametric")

epar_CI<- envelope(respar_CI)

#plot cofidence bands
points(state_sort$MET, epar_CI$point[2,], type="l", col="darkorange")
points(state_sort$MET, epar_CI$point[1,], type="l", col="darkorange")
```

Code for assignment 2

```
nirspectra<- read.csv2("NIRSpectra.csv")
features<-nirspectra[,-ncol(nirspectra)] #remove the target variable viscosity
nircomp<-prcomp(features) #PCA

summary(snircomp)
screeplot(nircomp, type="l", main = "Variance explained from PCA")
##Carles score plot
res <- prcomp(data[,-ncol(data)])
summary(res) # it is seen that just the first two components stand for more than 99%

#alternatively one can check eigenvalues, which are the square roots of the eigen values
lambda <- res$sdev^2
#proportion of variation of eigenvalues over hundred
proportion_variation<- as.numeric(sprintf("%2.3f",lambda/sum(lambda)*100))
sum(proportion_variation[1:2])##99% of the variance with the first 2 variables
##Screenplot to visualize it
screeplot(res,main = "")

library(ggplot2)
# The scores are meant to be the output x in the prcomp result
scores1 <- res$x[,1]
scores2 <- res$x[,2]

##ggplot of the scores of PC1, PC2 coordinates
ggplot(data= data.frame(scores1= scores1, scores2= scores2)) +
  geom_point((aes(x = scores1, y =scores2))) +
  xlab("scores1")+ ylab("scores2")

plot(ett,tva, ylab="PC2", xlab="PC1")

##2.2
plot(nircomp$rotation[,1], ylab="PC1") #ca 110 forsta var contributar till pc1
plot(nircomp$rotation[,2],ylab="PC2") #typ de sista contributar till pc2

##### 2.3
#
install.packages("fastICA")
library(fastICA)
set.seed(12345)
ica<- fastICA(features, n.comp=2, alg.typ = "parallel", fun="logcosh", alpha=1, method="R", row
    .norm = FALSE,
              maxit = 200, tol=0.0001, verbose = TRUE)
```

```
## a
Wprim<- ica$K %*% ica$W
plot(Wprim[,1])
plot(Wprim[,2])
##b
plot(ica$S[,1], ica$S[,2], xlab="ICA1", ylab="ICA2")
plot(ica$S)

##2.4

library(pls)

set.seed(12345)
pcr.fit<-pcr(Viscosity ~., data=nirspectra, validation="CV")
summary(pcr.fit)
validationplot(pcr.fit,val.type="MSEP")
```

732A95: Lab 5 block 1
Introduction to Machine Learning

Emma Wallentinsson, Zaida Liendeborg, Carles Sans Fuentes

December 12, 2016

Kernel methods and support vector machines

The task is to predict hourly temperature for a certain date and place in Sweden, with time intervals $4, 6, 8, \ldots, 24$, using kernel methods. The datafiles used are **stations.csv** and **temps50k.csv** which contains measured temperatures for different stations at different times and days, provided by SMHI. There are 50,000 observations in the data. There are three different gaussian kernels that were produced:

- The distance between a chosen station in Sweden and all other stations.
- The distance between the day the temperature was measured and a chosen day of a year.
- The distace between the hour of the day the temperature was measured and the time of interest.

To predict the temperature for time of interest, the three gaussian kernels above were combined. The kernels are computed by the formula:

$$K(x, x') = e^{-\frac{|x-x'|^2}{2\sigma^2}}$$

Where x is our point of interest, and x' is another point. When applying this, we get 50 000 different values for $K(x, x')$ due to the number of observations in the data. To calculate the expected value of our prediction, the following formula is used:

$$y^* = \frac{\sum_{i=1}^{N}(K(x, x_i')y_i)}{\sum_{i=1}^{N} K(x, x_i')}$$

1. Kernel for the station distances

The first kernel computed was the distances of Örberga station, found in Östergötland, with longitude = 14.826 and latitude = 58.4274 from all other stations. By using the formula given above, and setting its distance to Swedish miles, the kernel weight was computed as 1391.14 and the predicted temperature for the Örberga station is 6.287 degrees.

2. Kernel for the day distances

Following the same procedure as above, the kernel for the distance was computed between the day of interest, which is 2013-11-04, and the other days in the datafile. The kernel weight is 1750.042 and the predicted temperature for the chosen day is 4.401 degrees.

3. Kernel for different time intervals

There are 11 different time of interest to be used to compute the distances between them and the observed data. These time interval is from $4, 6, 8, \ldots, 24$. Following the same procedure, we get kernel weights for respective time interval:

```
1 13856.391 13832.761 11082.548  9244.326 10378.640 10344.911 12700.446 14708.791 13354.079
      10038.077  6003.440
```

Using these weights, we get the predicted temperature as:

```
1 3.172895 3.659227 4.527639 5.892712 6.576999 6.457490 5.767727 5.147736 4.674724 4.314486
      4.011356
```

4. Combination of three kernels

The three kernels above does not really say much about the temperature alone. We want a kernel to predict the temperatures with respect to distance between stations, days and hours combined. To do this, we can average the three kernels above to make a new kernel:

$$K_{temp}(x, x') = K_{dist}(x, x') + K_{days}(x, x') + K_{hours}(x, x')$$

The predicted temperatures from this new kernel is as follows:

```
1 4.772045 4.924128 5.238103 5.587153 5.768699 5.737876 5.574897 5.382387 5.246064 5.203754
      5.251991
```

The predicted temperatures from this kernel function is plotted in figure 1.

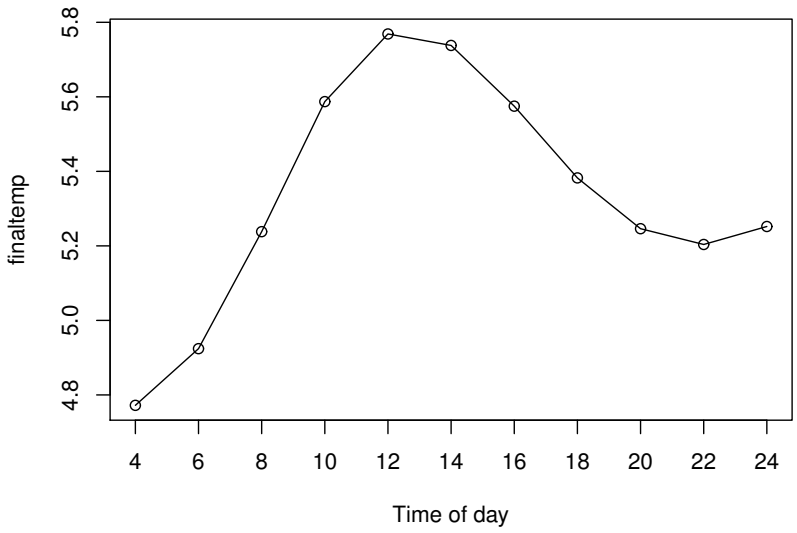


Figure 1: Predicted temperatures.

So, from figure 1, we see the predicted temperatures over the day 2013-11-04, for a place that has the coordinates *longitude =14.826* and *latitude = 58.4274*. When inspecting the coordinates, it came forward that the place for the measuring station is placed here in Östergötland. From

SMHI:s webpage, you can see that the average temperature for november 2013 in Östergötland were around 4-6 degrees, so the resulting plot is reasonable.

The true temperature for that day is given below. It can be seen that the prediction for that day is not bad at all in this case (it predicts 4.9 for time 6:00:00 and the true one is 4.9).

```
station_number station_name measurement_height latitude longitude
16462          85390     Kvarn Mo                  2  58.6277   15.4372
          readings_from         readings_to elevation       date     time
16462 2010-03-01 00:00:00 2016-10-01 08:00:00        72 2013-11-04 06:00:00
      air_temperature quality
16462             4.9       G
```

Nevertheless, considering the three kernel variables as independent does not make much sense in our case and it biases our results down. The intuition behind this fact is that by taking time and day separately, all temperatures from that day or that time in Sweden affect our predictor equally. Another way to say is, that by considering date and time separated to the station, dates and times that are far from weather stations will have the same value as those who are near, and thus it will affect the results negatively. For that, those who are nearby and hence more valuable for our prediction are not used. A better prediction for it could be weighing each variable according to cross-validation. For instance, giving more importance to stations that are nearby and less to time, assuming different Kernel models (e.g. Cauchy or Epanechnikov kernel). Another approach is by creating dependence between the variables and so by the kernels.

Appendix-Code

```
1
2 set.seed(1234567890)
3 library(geosphere)
4 stations <- read.csv("stations.csv")
5 temps <- read.csv("temps50k.csv")
6 st <- merge(stations,temps,by="station_number")
7
8 a <- 58.4274 # latitude
9 b <- 14.826  #longitude
```

```
89   }
90 }
91 ##bindging kernels and temps
92 predframehour<-cbind(karnelshours, st[,11])
93
94 #pred
95 weights<- c()
96 temp <- vector(length=length(times))
97 for (j in 1:11){
98   weights<-sum(karnelshours[,j]) #weights for time j
99   temp[j]<- sum(predframehour[,12]*predframehour[,j])/ weights
100 }
101
102
103 #############combining all three kernels.
104 finaltemp<-vector(length=11)
105 for ( i in 1:11){
106   vikt<-sum(karnelshours[,i]+karnels+karnelstime)
107   finaltemp[i]<-(sum(karnelshours[,i]*predframehour[,12]+karnels*predframehour[,12]+karnelstime
        *predframehour[,12]))/vikt
108 }
109 finaltemp
110 plot(finaltemp, type="o", xaxt="n", xlab="Time of day")
111 axis(1, at=1:11, labels=seq(04,24,2))
```

```r


longlat<-matrix(c(st[,1],st[,5], st[,4]), ncol=3)#matrix of longitude and lat
dist0<-matrix(0,ncol=2,nrow=50000)

for (i in 1:50000){
  long<-longlat[i,][2] #longitud
  lat<-longlat[i,][3] #latitud
  dist0[i,][1]<-distHaversine(c(long,lat), c(b,a))/10000 # distans i mil
  dist0[i,][2]<-longlat[i,][1] #station nummer
}

varians<-var(dist0[,1]) #varians of distances
karnels<-numeric()
for (i in 1:50000){
  avst<-dist0[i,1]
  karnels[i]<-exp(-(avst**2)/varians*2) #gaus kernels
}

predframe<-matrix(c(st[,11],karnels), ncol=2) #df with kernels and temperature

namnare<-sum(karnels)  #denominator
sum(predframe[,1]*predframe[,2])/ namnare #predicted


######kernel for day distance

st[,"date"]<-as.Date(st[,"date"]) #making to date format
preddate<-date  # exemplet for date 2013-11-07

disttime<-numeric()
for (i in 1:50000){
  dy<-difftime(preddate, st[i,"date"], units = "days")#dist in days
  disttime[i]<-dy[[1]] #the number of days
}
disttime<-abs(disttime)#dist in absolute value
varianstime<-var(disttime) #varians in distance
karnelstime<-numeric()
for (i in 1:50000){   #kernel for days
  avst<-disttime[i]
  karnelstime[i]<-exp(-(avst**2)/varianstime*2)
}
##yhat
predframetid<-matrix(c(st[,11],karnelstime), ncol=2) #df with temp and kernels
namnaretid<-sum(karnelstime) #denominator
sum(predframetid[,1]*predframetid[,2])/ namnaretid # predicted temp

######kernel for hour distance
times <- c("04:00:00", "06:00:00", "08:00:00", "10:00:00", "12:00:00", "14:00:00",
           "16:00:00", "18:00:00", "20:00:00", "22:00:00", "24:00:00")

times<- as.character(times)
##hours from midnight
obstime<-as.difftime(as.character(st$time), format="%H:%M:%S",units="hours")
wishtime<-as.difftime(as.character(times), format="%H:%M:%S", units="hours")


disthours<- matrix(0,ncol=11,nrow=50000)
#differences
for (i in 1:50000){
  for (j in 1:11){
    dif<- (obstime[i] - wishtime[j])

    disthours[i,j]<- dif
  }
}
head(disthours)
#absoult distance
disthours<-abs(disthours)
head(disthours)

summary(disthours)
##kernels
karnelshours<- matrix(0,nrow=50000, ncol=11)
for (j in 1:11){
  skillnad<- c()
  varhours<-var(disthours[,j])
  for(i in 1:50000){
    skillnad[i]<- disthours[i,j]
    karnelshours[i,j]<-exp(-(skillnad[i]**2)/varhours*2) #gaus kernels
```

732A95: Lab 6
Introduction to Machine Learning, Group 3

Zaida Liendeborg, Emma Wallentinsson, Carles Sans Fuentes

December 18, 2016

Assignment 1

In this assignment,we train a neural network with 10 hidden layers to learn the trigonometric sine function. 50 points were sampled from a uniform distribution in the interval [0, 10], and the the sine function is applied to each point. From this data, 25 of the 50 points were used to train the networks and the rest has been used as validation set. The validation set was mainly used to evaluate the gradient descent to avoid overfitting. There are different thresholds evaluated in this case: i/1000 where i = 1...10. The weights of the neural network are randomly initialized from the interval $[-1; 1]$.

To do this implementation, the package **neuralnet** was used. The function neuralnet() is used to train data for different thresholds. Using the function compute(), the neural network was used to predict on the validation data. Then the mean squared errors of the residuals were computed. Typically, when a network is trained, the error first decrease and then start to increase again. In figure 1, it can be seen that the lowest MSE for was obtained from the network which had a threshold of 0.004, after this threshold, the error start to increase again. At the best threshold, we have an error rate of 0.0003400357697.

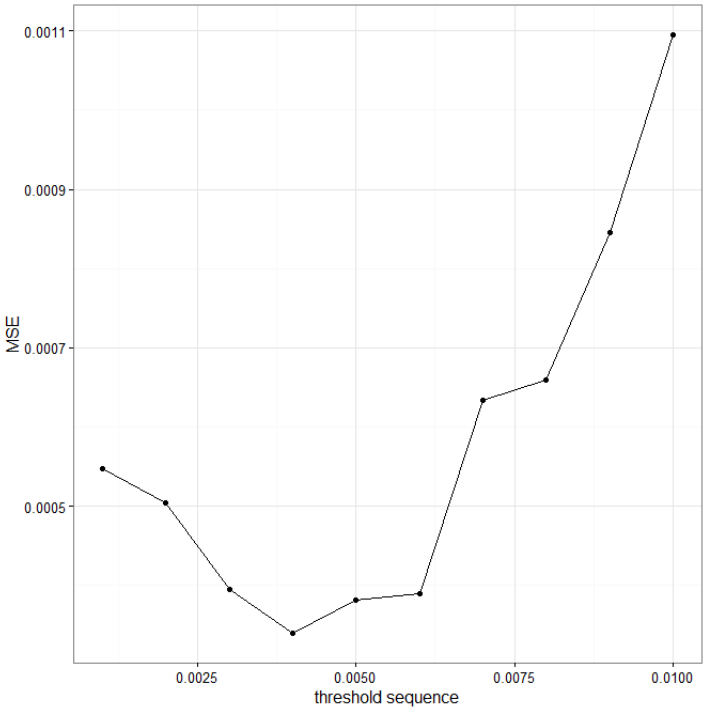


Figure 1: Plot for choosing best threshold considering MSE

The neural network with the chosen threshold is reported in figure 2.

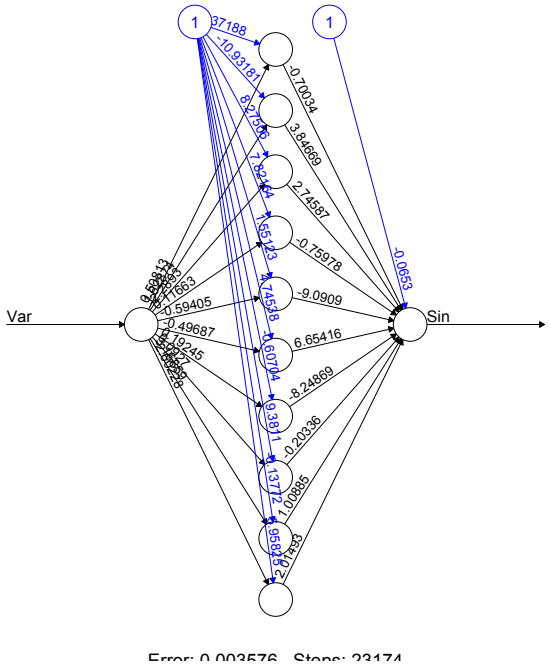


Figure 2: Neural network with 10 hidden layers, threshold = 0.004

The neural network in figure 2, visualizes the input neuron to the left and the hidden neurons in the middle, and the output neuron to the right. The black connections between them are the weights, and the blue are a bias term added in the step. The neural net is generally a black box, so its hard to say anything about the weights and the model.

To evaluate the prediction ability of this neural network, the predicted values from the validation set were plotted against its observed values. This is visualized in figure 3.

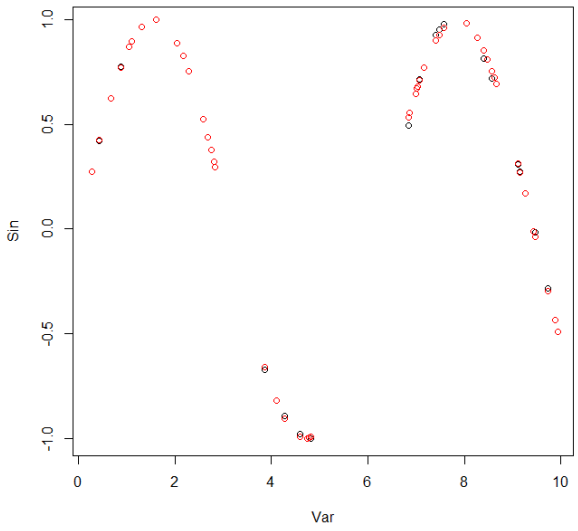


Figure 3: Plot of the prediction of the model (black points) for the best threshold with the original data (red points)

It is seen the that neural network gave a good fit of the sine function as it resembles the sine curve really well.

Appendix

Assignment 1

```r
##R script activity 1
#install.packages("neuralnet")
library(neuralnet)
set.seed(1234567890)


Var <- runif(50, 0, 10)
trva <- data.frame(Var, Sin=sin(Var))
tr <- trva[1:25,] # Training
va <- trva[26:50,] # Validation

# Random initializaiton of the weights in the interval [-1, 1]

start_weights<- runif(50, min = -1, max = 1)


MSE <- integer(0) ## vector of minimum MSE
for(i in 1:10) {
  nn <- neuralnet(
    formula = Sin~Var,
    data = tr, hidden=10, threshold = i/1000, startweights = start_weights )

  pr.nn <- compute(nn, va[,1])

  MSE[i]<- sum((va$Sin- pr.nn$net.result)^2)/nrow(va)      # Your code here
}
library(ggplot2)
ggplot2::ggplot(as.data.frame(MSE))+
  geom_line(aes(x =seq(1/1000,10/1000,1/1000), y = MSE))+
  geom_point(aes(x =seq(1/1000,10/1000,1/1000), y = MSE))+
  xlab("threshold sequence")+ ylab("MSE")+ theme_bw()

index_best_MSE<-which.min(MSE)## Index of minimum position
my_seq <- seq(1/1000,10/1000,1/1000)

plot(nn <- neuralnet(Sin~Var,data = tr, hidden=10,
                     threshold = my_seq[index_best_MSE]))
# Plot of the predictions (black dots) and the data (red dots)
plot(prediction(nn)$rep1)
points(trva, col = "red")
```