Group 1 - Block 1 - lab 1 - Group Report - 732A99 - Machine Learning

24 November 2019

# Assignment 1 - Spam classification with nearest neighbors

*The data file spambase.xlsx contains information about the frequency of various words, characters etc for a total of 2740 e-mails. Furthermore, these e-mails have been manually classified as spams (spam = 1) or regular e-mails (spam = 0).*

## 1.

*Import the data into R and divide it into training and test sets (50%/50%) by using the following code:*

Importing data:

Dividing the data into training and test sets (50/50):

## 2.

*Use logistic regression (functions glm(), predict()) to classify the training and test data by the classification principle:*

$$\hat{Y} = 1\ if\ p(Y = 1|X) > 0.5,\ otherwise\ \hat{Y} = 0 \\$$

*and report the confusion matrices (use table()) and the misclassification rates for training and test data. Analyse the obtained results.*

Run the model:

spam\_logistic <- glm(train$Spam~., family = binomial, data = train)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

predicted <- predict(spam\_logistic, test, type = "response")  
predicted\_class <- as.numeric(predicted > 0.5)  
  
# confusion matrix  
table(predicted\_class, test$Spam)

##   
## predicted\_class 0 1  
## 0 808 92  
## 1 143 327

The misclassification rates are as follows:

print((92 / (808 + 92))\*100)

## [1] 10.22222

print((143 / (143 + 327))\*100)

## [1] 30.42553

Mail is approximately 10% of the time misclassified as spam. Spam is approximately 30% misclassfied as regular mail. These results are quite bad, imagine if you lose 1 in every 10 mails because it is wrongly labelled as spam. Furthermore, the filter is not great because it still let’s 3 out of 10 spam mails through, thus not really solving the problem.

## 3.

*Use logistic regression to classify the test data by the classification principle:*

$$\hat{Y} = 1\ if\ p(Y = 1|X) > 0.8,\ otherwise\ \hat{Y} = 0 \\$$

*and report the confusion matrices (use table()) and the misclassification rates for training and test data. Compare the results. What effect did the new rule have?* Classify on a 0.8 split:

##   
## predicted\_class 0 1  
## 0 931 314  
## 1 20 105

## [1] 25.22088

## [1] 16

With the new model the misclassification of true mail as spam increased to 25% compared to the 10% in the previous model. On the other hand, the model became better in detecting spam. Now it correctly classifies spam as spam 84% of the time, with a 16% misclassification rate compared to the previous 30%. So the new model and the old model had different stenghts.

## 4.

*Use standard classifier kknn() with K=30 from package kknn, report the the misclassification rates for the training and test data and compare the results with step 2.*

##   
## predicted\_class 0 1  
## 0 702 180  
## 1 249 239

## [1] 13.33333

## [1] 51.02459

The misclassification rate of true mails classified as spam is approximately 13%. Thus this rate is only slightly worse than the misclassification rate of 10% in the model of step 2. On the otherhand, this K-nearest neighbor with K = 30 model performs terrible when it comes to classifing spam as spam. It has a misclassification rate of approximately 51%. Meaning 1 in every 2 spam mails gets mistaken for a normal email, making this filter pretty useless.

# 5.

##   
## predicted\_class 0 1  
## 0 644 185  
## 1 307 234

## [1] 22.31604

## [1] 56.74677

Compared to the model with K = 30, this model performs worse. It classifies mail as spam in 22% of the cases and spam as mail in over 56% of the cases. An obvious answer would be that this model is more sensitive to changes in the data, because the prediction depends only on 1 neighbor instead of a majority in the case of K= 30.

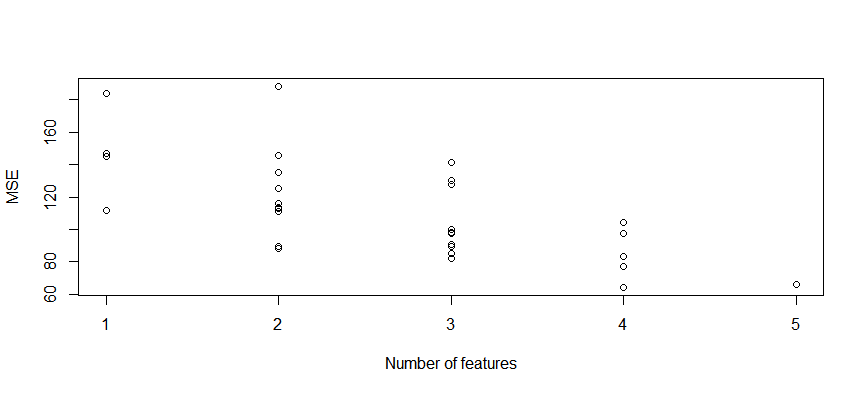
# Assignment 3 - Feature selection by cross-validation in a linear model

## 1.

*Implement an R function that performs feature selection (best subset selection) in linear regression by using k-fold cross-validation without using any specialized function like lm() (use only basic R functions). Your function should depend on:* • *X: matrix containing X measurements* • *Y: vector containing Y measurements* • *Nfolds: number of folds in the cross-validation* *You may assume in your code that matrix X has 5 columns. The function should plot the CV scores computed for various feature subsets against the number of features, and it should also return the optimal subset of features and the corresponding cross-validation (CV) score. Before splitting into folds, the data should be permuted, and the seed 12345 should be used for that purpose.*

# 2.

*Test your function on data set swiss available in the standard R repository. Fertility should be Y and all other variables should be X. Nfolds should be 5. Report the resulting plot and interpret it. Report the optimal subset of features and comment whether it is reasonable that these specific features have largest impact on the target.*



## $CV  
## [1] 64.48673  
##   
## $Features  
## [1] 1 0 1 1 1

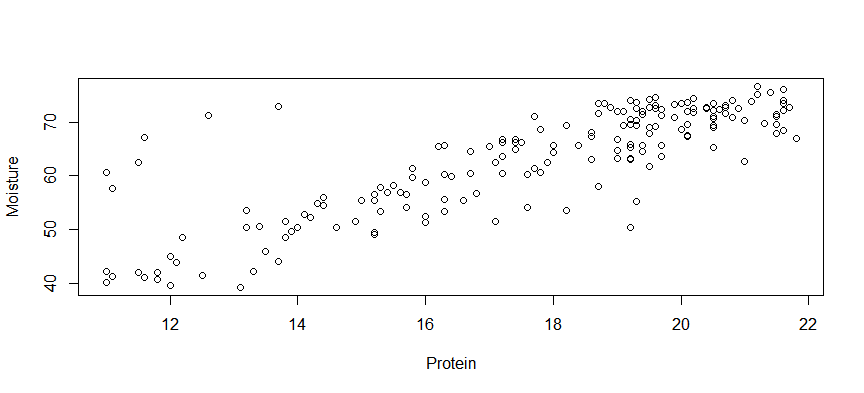
(Code and figures in the appendix) \* On the X axis of the plot are shown the number of features used at a time. The Y axis shows the minimal Mean Squared Error (MSE) obtained in the different folds. \* As we increase the number of features used in our model, the MSE reduces. \* We can also see in the 1 variable case, that there should be specifically one feature that reduces drastically the MSE. \* Another interesting characteristic of this plot, is that the min MSE was obtained with 4 features, instead of 5.

The best result was obtained using features 1 (Agriculture), 3 (Education), 4 (Catholic) and 5 (Infant Mortality). This result, makes sense for variables 1, 3 and 4 as these variables are related with education and lifestyle. But variable 5 does not make sense, because infant mortality should not have any effect on fertility.

# Assignment 4 - Linear regression and regularization

*The Excel file tecator.xlsx contains the results of study aimed to investigate whether a near infrared absorbance spectrum can be used to predict the fat content of samples of meat. For each meat sample the data consists of a 100 channel spectrum of absorbance records and the levels of moisture (water), fat and protein. The absorbance is -log10 of the transmittance measured by the spectrometer. The moisture, fat and protein are determined by analytic chemistry.*

## 1.

*Import data to R and create a plot of Moisture versus Protein. Do you think that these data are described well by a linear model?* Plot of Protein vs Moisture:  
 There is a clear linear relationship between Protein and Moisture and it could be described well by a linear model.

**Consider model in which Moisture is normally distributed, and the expected Moisture is a polynomial function of Protein including the polynomial terms up to power .**  
Probabilistic model for :

$$M\_i \sim N(\sum\_{j=0}^{i} w\_j x^j, \; \sigma^2),\; i \in \{1, 2, 3, 4, 5, 6\} \\$$

We use mean of squared error (MSE) because we want to make our model to be as close as possible to the training data . So the lower the value of MSE, the better the model fits the training data. That way, we can select the best model by looking at the lowest MSE value.  
MSE for :

According to the plot, if we look at MSE values on training set, is the best model. However, if we look at the validation MSE, has the smallest errors. This can be explained by bias-variance trade-off. Models with higher degree of polynomial have lower bias but higher variance, so the model would overfit the training data and resulting in higher prediction errror when presented with validation data. If the models have lower degree of polynomial, it would underfit the data and have higher training MSE. However, due to the lower variance, they generalize better than more complex models hence the lower validation MSE.

**Variable selection of a linear model in which Fat is response and Channel1-Channel100 are predictors by using stepAIC:**

There are 63 variables selected out of 100 variables.

**Ridge regression:**

**LASSO:**

Based on two plots above, we can see that the ridge regression still use all of 63 variables as the value of increased, although the value of the coefficients got smaller but not reaching 0. In the LASSO model, some coefficients reached to 0 even when is still not relatively big. When or , there were 10 variables left that have nonzero coefficient. In conclusion, LASSO models use only several variables as the value of grows while ridge regression models use all variables.

**Cross-validation to find the optimal LASSO model:**

The value of chosen is which is the same as using all 63 variables. In conclusion, the best model for predicting Fat is by using 63 variables that were selected by stepAIC. As seen on the plot, the value of MSE increased as the value of increased too.

# Appendix

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##   
## predicted\_class 0 1  
## 0 808 92  
## 1 143 327

## [1] 10.22222

## [1] 30.42553

##   
## predicted\_class 0 1  
## 0 931 314  
## 1 20 105

## [1] 25.22088

## [1] 16

##   
## predicted\_class 0 1  
## 0 702 180  
## 1 249 239

## [1] 13.33333

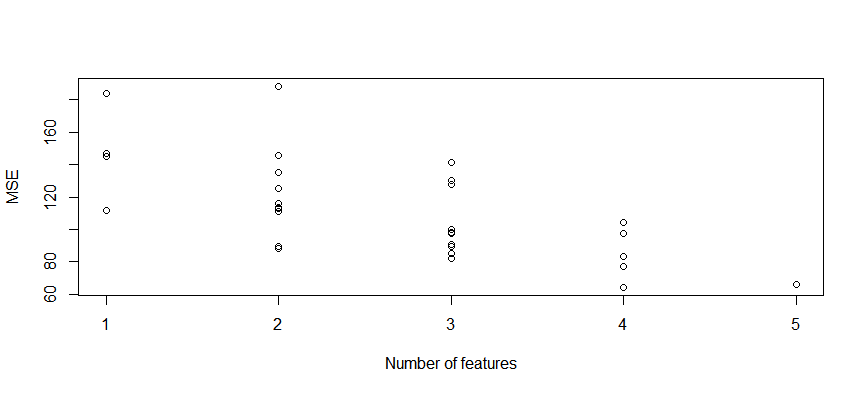
## [1] 51.02459

##   
## predicted\_class 0 1  
## 0 644 185  
## 1 307 234

## [1] 22.31604

## [1] 56.74677

#ASSIGNMENT 3  
  
data(swiss)  
  
#linear regression  
mylin=function(X,Y, Xpred){  
 Xpred1 = cbind(1,Xpred)  
 #NotMISSING: check formulas for linear regression and compute beta  
 x1 = cbind(1,X)  
 beta <- ((solve(t(x1) %\*% x1)) %\*% t(x1)) %\*% (Y)  
 Res = Xpred1 %\*% beta  
 return(Res)  
}  
  
myCV=function(X,Y,Nfolds){  
 n=length(Y)  
 p=ncol(X)  
 set.seed(12345)  
 ind=sample(n,n)  
 X1=X[ind,]  
 Y1=Y[ind]  
 sF=floor(n/Nfolds)  
 MSE=numeric(2^p-1)  
 Nfeat=numeric(2^p-1)  
 Features=list()  
 curr=0  
   
 #we assume 5 features.  
   
 for (f1 in 0:1)  
 for (f2 in 0:1)  
 for(f3 in 0:1)  
 for(f4 in 0:1)  
 for(f5 in 0:1){  
 model= c(f1,f2,f3,f4,f5)  
 if (sum(model)==0) next()  
 SSE=0  
   
 for (k in 1:Nfolds){  
 #not MISSING: compute which indices should belong to current fold  
 test\_index <- if (k == Nfolds){  
 (sF\*(k-1)+1):n  
 }   
 else{  
 (sF\*(k-1)+1):((k \* sF))  
 }  
 #MISSING: implement cross-validation for model with features in "model" and iteration i.  
 #MISSING: Get the predicted values for fold 'k', Ypred, and the original values for folf 'k', Yp.  
 if(sum(model)==1){  
 Ypred <- mylin(X[,which(model==1)][-test\_index], Y[-test\_index], X[,which(model==1)][test\_index])  
 }  
 else{  
 Ypred <- mylin(X[,which(model==1)][-test\_index,], Y[-test\_index], X[,which(model==1)][test\_index,])  
 }  
 Yp <- Y[test\_index]  
 SSE=SSE+sum((Ypred-Yp)^2)  
 }  
 curr=curr+1  
 MSE[curr]=SSE/n  
 Nfeat[curr]=sum(model)  
 Features[[curr]]=model  
   
 }  
   
 #MISSING: plot MSE against number of features  
 nr\_features <- c()  
 for (i in 1:length(Features)){  
 nr\_features[i] <- sum(Features[[i]])   
 }  
 plot(nr\_features, MSE, xlab = "Number of features")  
 #plot End  
 i=which.min(MSE)  
 return(list(CV=MSE[i], Features=Features[[i]]))  
}  
  
  
myCV(as.matrix(swiss[,2:6]), swiss[[1]], 5)



## $CV  
## [1] 64.48673  
##   
## $Features  
## [1] 1 0 1 1 1