Vikgu708
Viktor Gustafsson

Lab 1

Exercise 1

2. Result for threshold = 0.5

Train-data

```
Miss rate: 16%

    real
pred   0   1
   0 804  93
   1 127 346
```

Test-data

```
Miss rate: 17%

     real
pred   0   1
   0 808  92
   1 143 327
```

The result shows a slightly higher miss rate for test data, which is not surprising as the testdata will naturally deviate in distribution slightly from train data.

That the miss rates are similar for test and train indicates a reasonable bias/variance tradeoff, as the model seems to classify test data similarly to train data. This is not surprising, as the model is linear and thus not very complex. A higher complexity would result in higher variance.

3. Result for threshold 0.8

Train-data

```
Miss rate: 25%

     real
pred   0   1
   0 921 333
   1  10 106
```

Test-data

```
Miss rate: 24.4%

     real
pred   0   1
   0 931 314
   1  20 105
```

It is clear that a higher threshold results in a worse prediction rate. This because a threshold of 0.8 implies a loss function that penalises False positives more than false negatives. This is then also reflected in the result, as FPR<<FNR. The tradeof, is that the miss rate is noticeably larger than a threshold that does not prioritize any prediction.

Vikgu708
Viktor Gustafsson

4. KKN Classifier K=30

As the results are to be compared to logistic regression with threshold 0.5, the same threshold will be used for the following prediction using a KKN classifier with k=30.

Train-data

```
Miss rate: 47.2%

     real
pred   0   1
   0 580 276
   1 371 143
```

Test-data

```
Miss rate: 31.3%
     real
pred   0   1
   0 702 180
   1 249 239
```

The KNN model performs much worse than the logistic regression. This is due to the curse of dimensionality. As the observation of features are 48-dimensional, a measure of distance is less valid to classify data. This because the average distance to a neighbour increases linearly with each dimension, which means that there is less possibility to segment data into separate clusters.

5. KNN Classifier K=1

Train-data

```
Miss rate: 0%

     real
pred   0   1
   0 931   0
   1   0 439
```

Test-data

```
Miss rate: 35.9%

     real
pred   0   1
   0 644 185
   1 307 234
```

That K=1 for train has a 0-miss rate makes sense, as the same point is found for each observation and are thus classified as the label for themselves. This is naturally better than K=30, where the training data will classify on points nearby, and not on the points own label, and is thus subject to misclassification when points are nearby the bounds of two clusters. The miss rate for test data is like K of 30, which is not unexpected since the curse of dimensionality still applies.

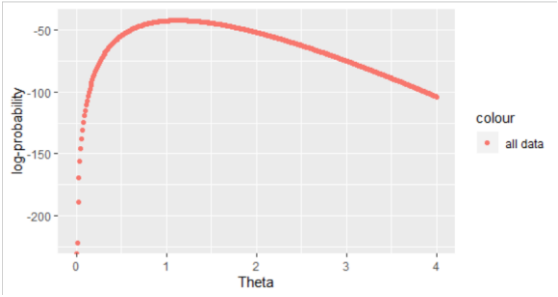
Vikgu708
Viktor Gustafsson

Exercise 2

2.

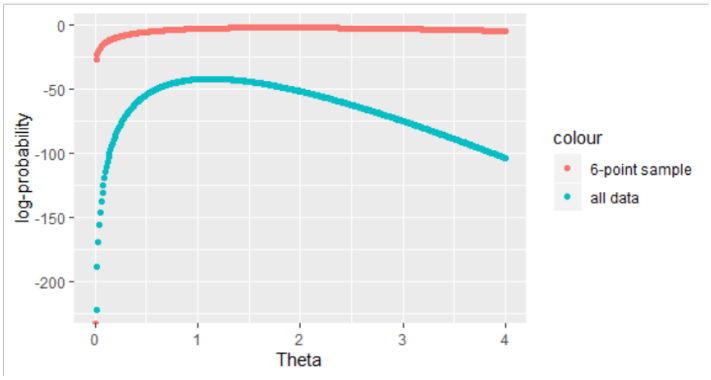
Loglikelihood is derived from log of likelihood for all observed values for exponential function. It is then implemented in the following function

```
loglike = function(vector,theta){
  n= nrow(vector)
  return (n*log(theta) - theta*sum(vector))
}
```



Maximum Likelihood is found at theta=1.126 (as maximum likelihood is equivalent as maximising loglikelihood)

3.

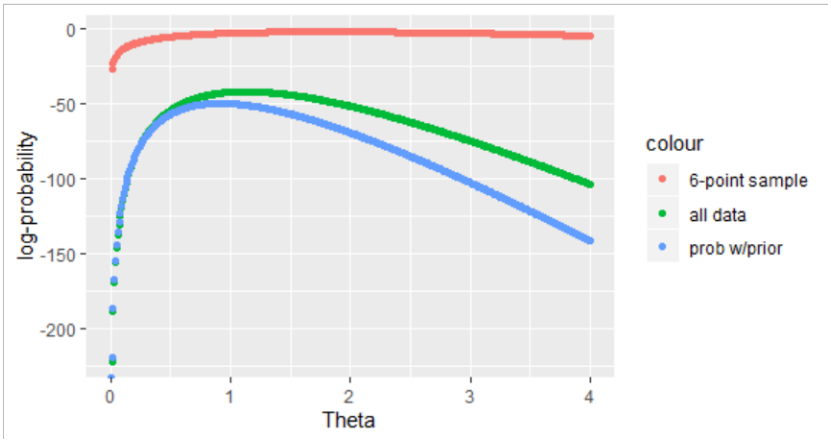


The reliability for the 6-point sample is much worse, as a smaller sample has less total information about the distribution. We can see this as the range for the likelihood for the 6-point data is much narrower than the range for likelihood on the entire dataset.

For the 6-point sample, theta for maximum likelihood is equal to 1.785. A larger theta value means that smaller values are likelier to occur, which means that the 6_point sample probably has a lower average value than the rest of the dataset.

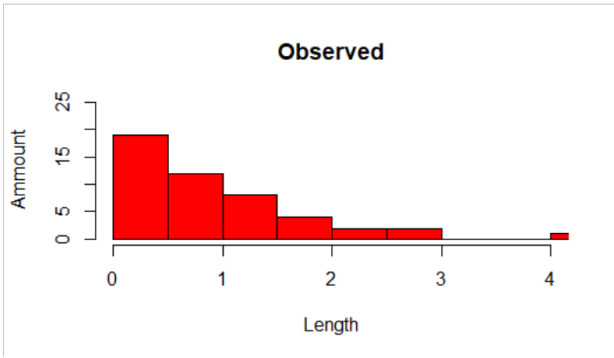
Vikgu708
Viktor Gustafsson

4.

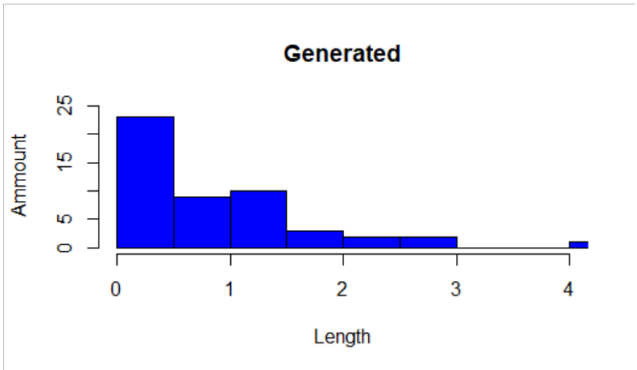


Optimize function in R gives us the maximum value for the prior adjusted log probability as theta = 0.912. Theta is smaller than in part 2 because of the prior distribution with relatively high lambda value that will give a higher likelihood for smaller theta values.

5.



50 generated values using theta = 1.126199



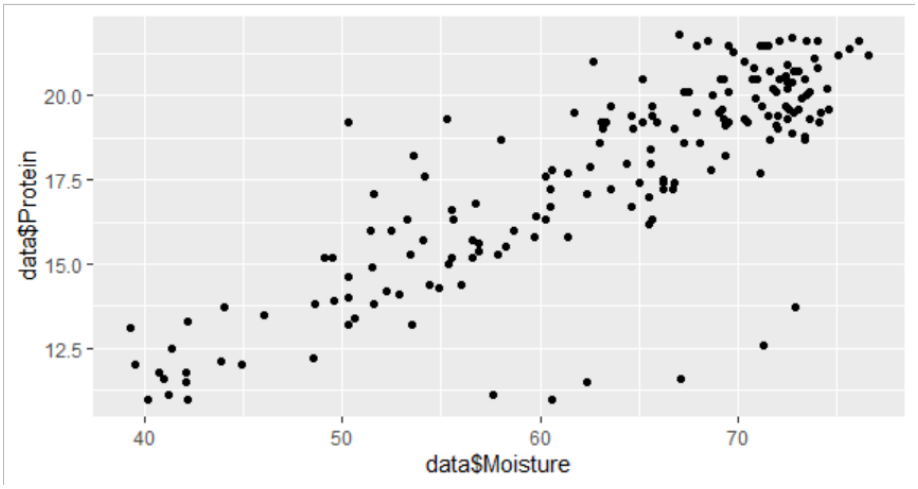
As we can see, the distributions are similar. This implies that the estimated theta value found with MLE is like the original distribution for the observed values, as the generated values share similar sizes in each column of the histogram. The deviations can most likely be attributed to randomness.

Vikgu708
Viktor Gustafsson

Exercise 4.

1.

A basic prediction can be found with a linear model, although with quite a large spread I.E error. The correlation between the variables seems to be linear just by observing the linear increase in protein by moisture shown by the graph.

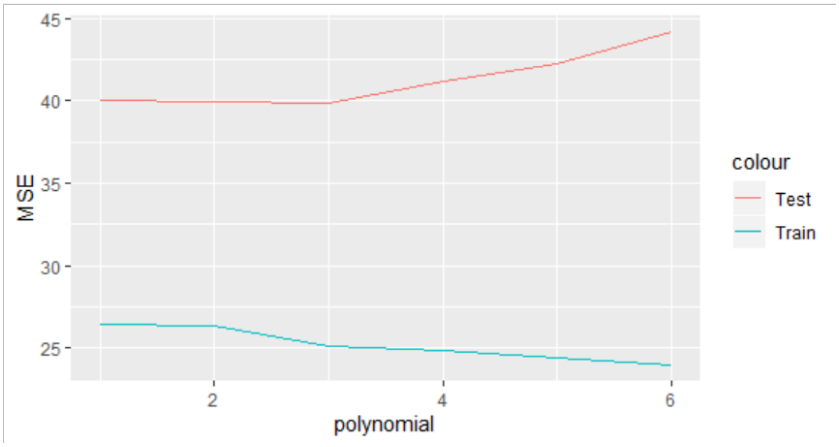


2.

MSE as this is the measure to minimize over when maximising maximum likelihood for a gaussian distribution. $M_i = N(c_0 + \sum_{k=1}^{i} c_i x^i, \sigma)$

3.

The plot for MSE for test and train data is shown below.



This matches with the theory that increasing model complexity will produce less MSE for training set, and an increasing MSE for test set. The bias error is decreased with model complexity, as an increase in complexity enables the model to rely more on data than on initial assumptions of model. The variance error is increased with model complexity, as the variance in the model itself is increased.

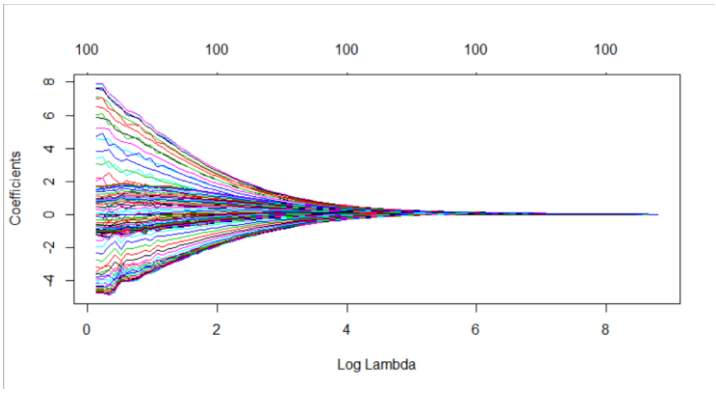
For training data, the increase in variance do not produce larger error as the variance is attributed to the data itself. The variance does however increase error rate in test data, as the model is fitted more and more closely to the initial training data, implying overfitting. This means that the increase in error from variance eventually overtakes the increase in error from bias. This is observed to happen after polynomial of 2 in the model.

Vikgu708
Viktor Gustafsson

4.

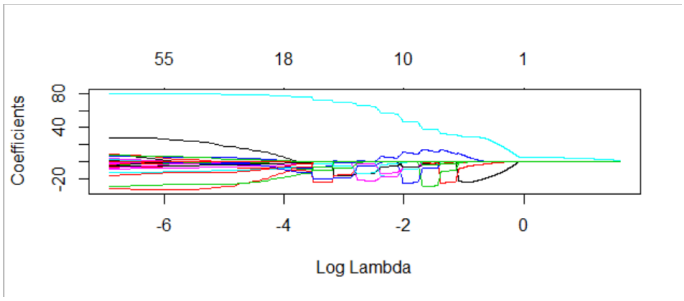
36 variables were removed after the backwards choosing stepAIC. This means that many variables are not necessary to compute the response variable fat and thus suggests high covariance for these removed variables and the kept ones. Thus only 64 variables are needed to predict the output in similar quality to keeping all variables.

5.



With ridge regression, the coefficients will decrease close to linearly with an increase in log(lambda). Basically, as the minimization function is penalized with the square of each coefficient, along with the squared prediction error, the coefficients will be increased for larger lambda values (where lambda is the factor for penalizing square of coefficients).

6.

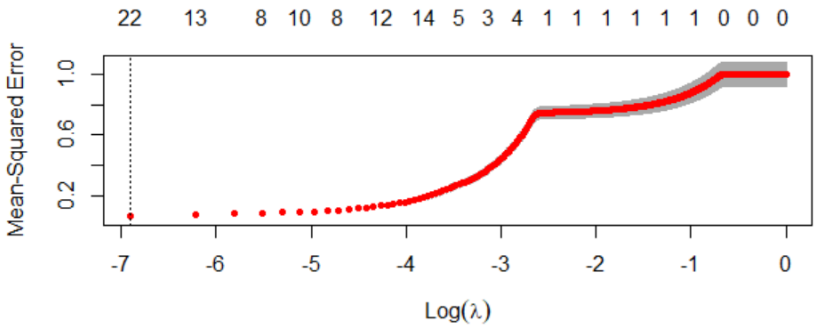


Lasso regression deviates largely from ridge regression as Lasso tends to quickly set coefficients to zero, while ridge simply minimizes the variables continuously. This can be seen in the graph, where most values are set to zero when lambda =1. This is since the minimization function is penalized with the absolute value of the coefficients, and thus there is a high likelihood of the minimization function's gradient to tangent coefficients' axis.

7.

After running cv.glmnet, the optimal lambda was found to be 0. The MSE was found to increase with each variable removed. This leads to all variables being kept. This suggests that even with all variables, the model do not overfit during the folds in cross validation. This suggests that the model cannot overfit on the data. It is noted however that MSE is only increased slightly with increasing lambda and this suggests, like the AIC suggests, that many variables can be removed without sacrificing much in goodness of fit.

Vikgu708
Viktor Gustafsson



8.

Stepwise AIC removed 36 variables, while cross validation kept every variable. I hypnotize that this is due to that AIC penalizes amount of variables by definition, while cross validation method with different lambdas penalizes amount of variables depending on lambda, and will result in worse MSE if the model is not overfitting as said before and a low lambda will in this case be picked.

Vikgu708
Viktor Gustafsson

Appendix

Code for exercise 1

```
# Loading

library("readxl")

library("kknn")

# read file



#Calculates new confussion matrix

confussion = function(pred,real,name){

 pred <- as.numeric(pred>threshold)

 length(pred)

 length(real)

 cat("-----------------------------------------\n")

 cat("\n\n STATS for ")

 cat(name)

 conf = table(pred,real)

 cat("\n\n Confussion matrix \n\n ")

 print(conf)

 cat("\n")


 mcr = 1-sum(diag(conf))/sum(conf)

 cat(paste("MissRatio; ", mcr))

 cat("\n")

}

data <- read_excel("spambase.xlsx")


#split data 50/50 training and testing

n=dim(data)[1]

set.seed(12345)

id=sample(1:n, floor(n*0.5))
```

Vikgu708
Viktor Gustafsson

```
train=data[id,]

test=data[-id,]


threshold=0.5

#Train linear regression model on train-data

logistic <- glm(Spam ~ . ,data=train,family="binomial")


#Get stats for traindata

pred_train <- predict(logistic,train,type="response")

confussion(pred_train, train$Spam, "train")


#get stats for testdata

pred_test <- predict(logistic,test,type="response")

confussion(pred_test,test$Spam,"test")


knnModel = kknn(Spam~.,train,test, k = 30)

knnpred_test = fitted(knnModel)

confussion(knnpred_test,test$Spam, "### knnTest ###")


knnModel = kknn(Spam~.,train,train, k = 30)

knnpred_train = fitted(knnModel)

confussion(knnpred_train,train$Spam, "### knn Train ###")
```

Vikgu708
Viktor Gustafsson

Code for exercise 2

```
# Loading

library("readxl")

library("kknn")

library("ggplot2")

# read file

data <- read_excel("machines.xlsx")


#loglikelihood for a vector of observations and its theta value

loglike = function(vector,theta){

  n = length(vector)

  return (n*log(theta) - theta*sum(vector))

}


#loglike with prior distrubution for theta

bayesian = function(const,theta,vector){

  prior = (log(const)-const*theta)

  return (prior + loglike(vector,theta))

}


# inverse of CDF for exponential distrubution

getLength = function(prob,theta){

  return( -log(1-prob)/theta )

}


thetaX = seq(0,4,0.01)

#sample 6 first points

data_6 = data$Length[1:6]


#find theta with highest corresponding likelihood for loglike and loglike with prior

theta1 = optimize(loglike,c(0,100),vector=data$Length,maximum="true")$maximum
```

Vikgu708
Viktor Gustafsson

```
theta2 = optimize(bayesian,c(0,100),const=10,vector=data$Length,maximum="true")$maximum


#p: loglike on all observations p2: loglike on 6 first observations p3: loglike w/ prior on all observation

p <- qplot(thetaX,loglike(data$Length,thetaX),col="all data",xlab = "Theta", ylab ="log-probability")

p2 <- geom_point(aes(thetaX,loglike(data_6,thetaX),col="6-point sample"))

p3 <- geom_point(aes(thetaX,bayesian(10,thetaX,data$Length),col="prob w/prior"))

p+p3


set.seed(12345)


#histogram over observed vals

h1 <- hist(data$Length,xlim=c(0,4),ylim=c(0,25),main="Observed",xlab="Length",ylab="Ammount",
col="red")

#generate new vals with optimized theta

newVals = getLength(runif(50, 0, 1),theta1)

#histogram over new vals

h2 <-
hist(newVals,xlim=c(0,4),ylim=c(0,25),main="Generated",xlab="Length",ylab="Ammount",col="blue")

h1

h2
```

Vikgu708
Viktor Gustafsson

Code for exercise 4

```
# Loading

library("readxl")

library("ggplot2")

library("MASS")

library("glmnet")



data <- read_excel("tecator.xlsx")



makePolynomialModels = function(maxPol,train,test){

 trainData = data.frame("Moisture" = train$Moisture)

 testData = data.frame("Moisture" = test$Moisture)

 mse_train = c()

 mse_test = c()

 for(i in 1:maxPol){

  trainData[sprintf("Protein %d",i)]=train$Protein^i

  testData[sprintf("Protein %d",i)]=test$Protein^i

  model <- glm(Moisture ~ . ,data=trainData,family="gaussian")

  pred_train <- predict(model,trainData,type="response")

  pred_test <- predict(model,testData,type="response")

  mse_train[i] = sum((trainData$Moisture-pred_train)^2)/nrow(trainData)

  mse_test[i] = sum((testData$Moisture-pred_test)^2)/nrow(testData)

  #cat(sprintf("\n MSE %d Test: %f Train: %f",i,mse_train[i],mse_test[i]))

 }

 p <-
qplot(x=seq(1,maxPol,1),y=mse_train,color="Train",ylab="MSE",xlab="polynomial",geom="line")

 p2 <- geom_line(aes(seq(1,maxPol,1),mse_test,color="Test"))

 return(p + p2 )


}
```

Vikgu708
Viktor Gustafsson

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


makePolynomialModels(6,train,test)

model <- glm(Fat~ . -Sample -Protein - Moisture,data=data,family="gaussian")
modelAIC = stepAIC(model,trace=FALSE,direction="backward")
length(modelAIC$coefficients)

X = as.matrix(scale(data[,2:101]))
Y = scale(data$Fat)

model = cv.glmnet(x=X,y=Y,alpha=0,family="gaussian",lambda= seq(0,1,0.001),nfolds=10)
plot(model)

model$lambda.min
```