Lab 8 - Introduction to Machine Learning

Akshaya Swuroupikka Balasubramanian, Gustav Sternelöv, Araya Eamrurksiri and Yixuan Xu

11 December 2015

Assignment 1

1.1

A function that uses the simple perceptron algorithm is implemented with the following code:

```
### Assignment 1
## 1.1
# "lambda" is the learning rate, "stopcond" the parameter for the stop condition
spl <- function(x, y, lambda, weights, stopcond){
  yHat <- 0
  MisClass1 <- 0
  MisClass2 <- 1
  # Loop that runs until stopping condition is met.
  while(abs(MisClass2 - MisClass1) > stopcond){
    for(i in 1:nrow(x)){
      # The weights are multiplied with i:th row
      linComb <- sum(weights * x[i, ])
      # Uses the sign function to classify the observation
      yHat[i] <- sign(linComb)
      # Updates the weights
      weights <- weights + lambda*(y[i] - yHat[i]) * x[i, ]
    }
    # Calculates the misclassification value
    # The misclass value for the latest iteration is saved for comparison with
    # the next misclass value
    MisClass2 <- MisClass1
    confMat <- table(yHat, y)
    MisClass1 <- 1 - sum(diag(confMat)) / sum(confMat)
  }
  RetU <- list(Wt = weights, miscTr = MisClass1)
  return(RetU)
}
# The classify the test set with the obtained model the following code is used
testSPL <- spl(x=inputs, y=train[,58], lambda=0.1, weights=init_weights, stopcond=0.01)
yHatTest <- 0
for(i in 1:nrow(testInput)){
  # Uses the weights returned by the function on every row
  linCombTest <- sum(testSPL$Wt * testInput[i,])
  yHatTest[i] <- sign(linCombTest)
}
confMatTest <- table(yHatTest, test$Spam)
MisClass3 <- 1 - sum(diag(confMatTest)) / sum(confMatTest)
```

1.2

Load the data into R and split it into a training and test set partitioning 70/30 using the seed 12345.

1.3

The initial weights are obtained by simulating from the normal distribution. These weights are used together with the train data set to fit a simple perceptron model. Regarding the other parameters the learning rate λ is set to *0.1* and the stopping condition is set to *0.01*.

1.4

The fitted model in *1.3* is used to classify the observations in the test set. Then, the misclassification rate is computed to be 11.59 %. With such a low percentage of misclassified observations the fitted model seem to be a good fit to this data.

1.5

Step *1.3* and *1.4* is repeated, but with another seed. This time the misclassification rate when classifying the test data is computed to be 8.69 %. Then, again, the conclusion is that the fitted model is a good fit to this data set.

1.6

For the variable *Spam* the class *-1* is changed to *0*. Then a logistic regression is fitted where *Spam* is the target variable. The model is fitted on train data and evaluated by predicting the class for the observations in the test set. A confusion matrix of the predicted values against the observed values is created and can be seen below. Since 17 of the 138 observations is misclassified the obtained misclassification rate is 12.3 %.

```
##          fitted
## test.Spam  0  1
##         0 71 10
##         1  7 50
```

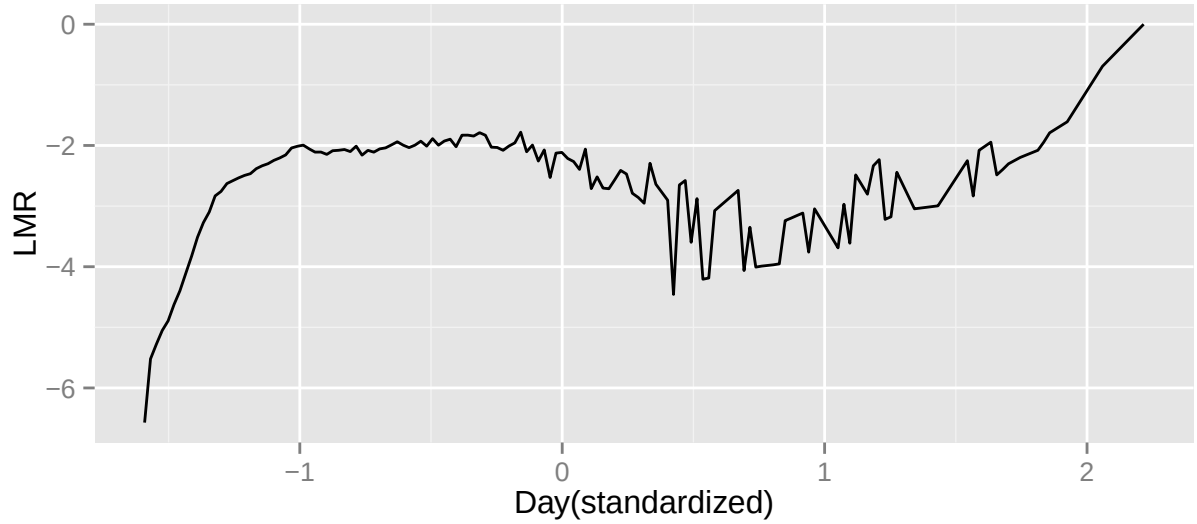
1.7

The two first models are almost identical simple perceptron models, the difference between the models is how the initial weights is set in the beginning. A comparison of these models gives that the second one are a little bit better since it has a lower misclassification rate for the test set (8.59 % vs 11.59 %). The third model was logistic regression model and with this model a misclassification rate of 12.3 % was obtained. That is the highest misclassification rate of all the fitted models. The simple perceptron models seem to be better fits, and of the three fitted models the second simple perceptron model gave the best result. Moreover, it can be seen that the value of initial weights have an effect on the performance of the model.

Assignment 2

2.1-2.2

The studied data set contains information about the mortality rate for fruit flies for each day. The data comes from a study where the theory that the mortality rates (probability of dying per unit time) of many organisms increase at an exponential rate was tested. The variable *LMR*, that is the logarithm of the variable *Rate*, is plotted against the variable *Day*(standardized).



The relationship between *LMR* and *Day* is quite different for different periods. During the first days the mortality rate is increasing rapidly. Then, during the middle of the observed time period, the mortality rate is constant a short while before it decreases a little. At the end of the observed time period the mortality rate increases, not as fast as in the beginning but still rather fast.

2.3

A multilayer perceptron network with a single hidden neuron, *tanh* as the activation function ,a maximum number of iterations that is $1 * 10^6$ and a treshold for stopping the fitting of the model at *0.1*.

a)

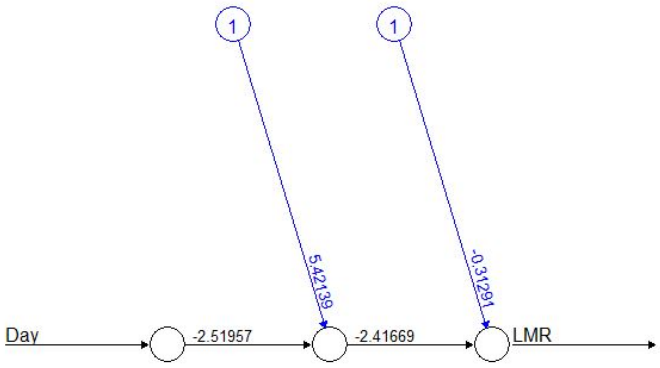
The obtained model is

$$y = -0.31291 - 2.41669Z$$

where

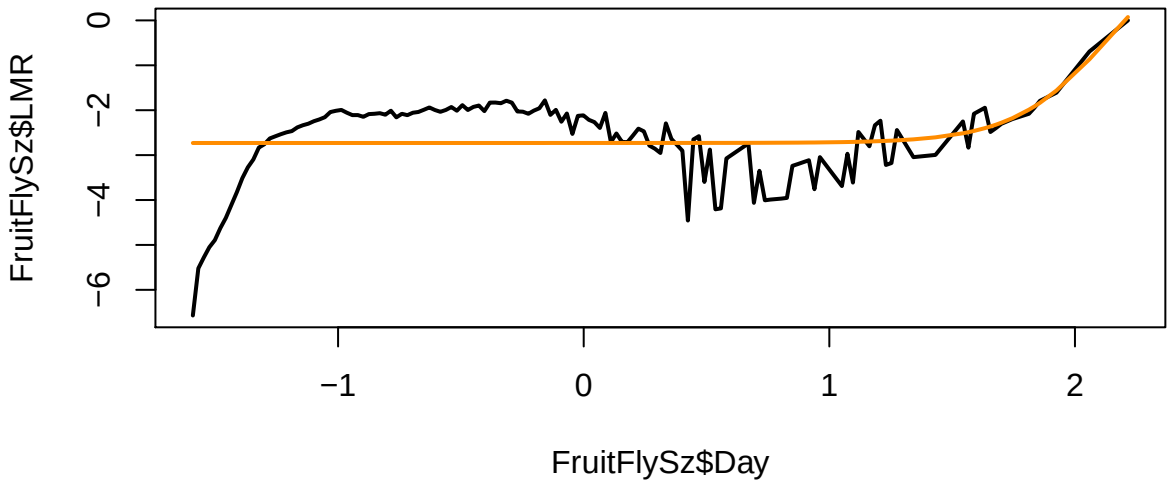
$$Z = tanh(5.42139 - 2.51957X)$$

A plot visualizing the network's weights.



b)

The fitted values obtained are visualized with the orange line and compared against the observed values (the black line).



It can be easily seen that model1, a Multilayer Perceptron network with one hidden neuron, is not a good model to fit with this data. The fitted line does not follow the observed data at all. Only a few data is captured by the model. Thus, it can be concluded that this model performs rather bad.

2.4

Again, a multilayer perceptron model is fitted. The new model has the same settings as the model in *2.3*, except that this model has two hidden neurons instead of one.

a)

The obtained model is

$$y = -1.84151 - 3.97184Z_1 + 3.30129Z_2$$

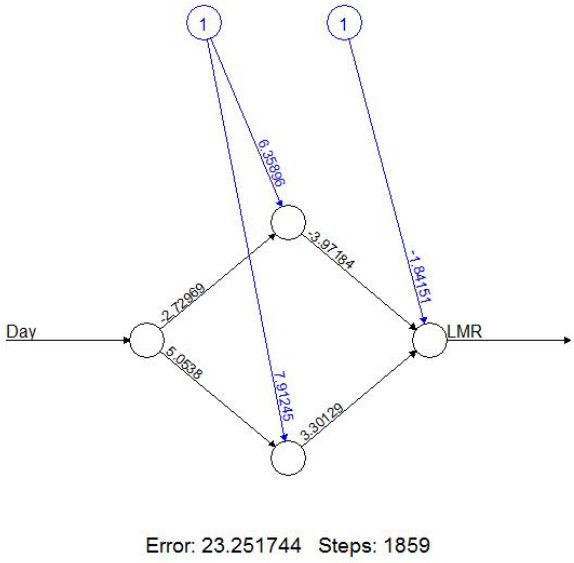
where

$$Z_1 = tanh(6.35896 - 2.72969X)$$

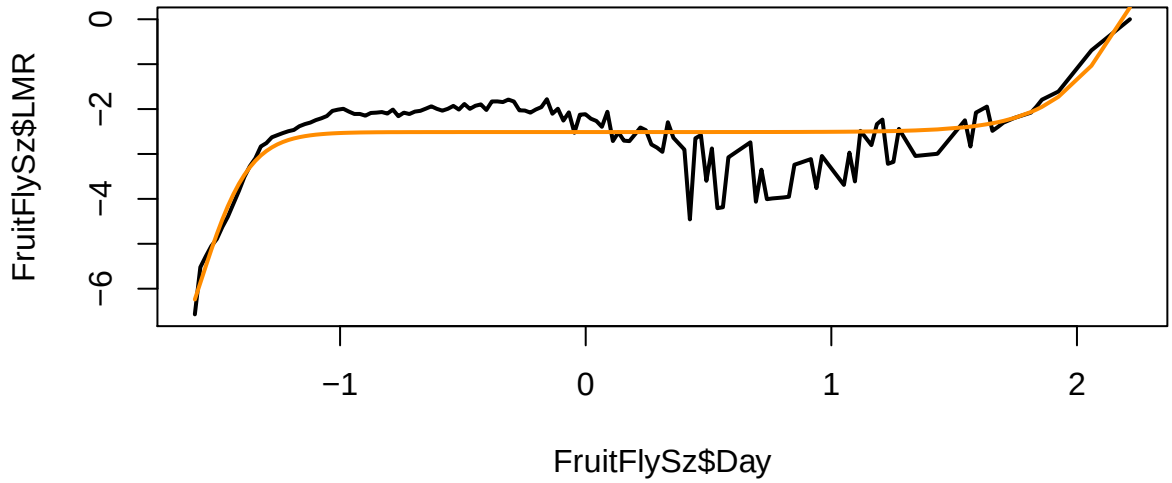
and

$$Z_2 = tanh(7.91245 + 5.053798X)$$

A plot visualizing the network's weights.



b)

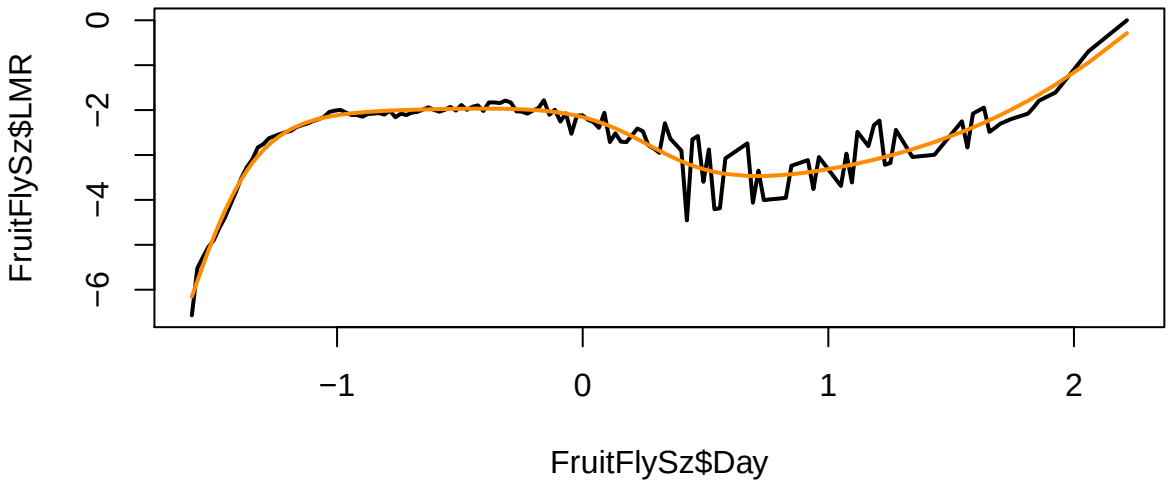


The figure shows that model2, a Multilayer Perceptron network with two hidden neurons, performs much better than the previous model using with one hidden neuron. The fitted line from this model can capture the observed data at the beginning of the plot rather well compare with the model1 which has a straight line. However, this model appears to be less effective when fitted with the obseved data in the middle of the plot, it can not capture the underlying trend of the observed data and instead produce a straight line.

2.5

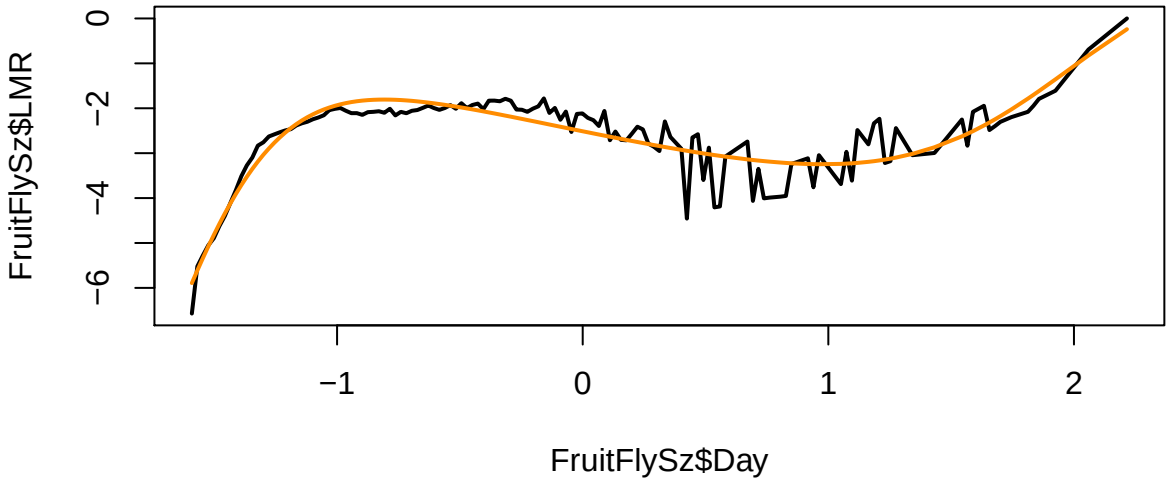
Four hidden neurons

Error:6.21 Number of iterations:2355



Five hidden neurons

Error:9.29 Number of iterations:8201



Comparison of the models

Both models are good fits to the data set, especially the first model with four hidden neurons seem to be a really good fit. When comparing the model in *2.4* with two hidden neurons and the model with four hidden neurons it is evident that a more complex model is a better fit to this data. This visual conclusion is also confirmed by the number of iterations and the error which is lower for the model with four hidden neurons. However, when a even more complex model with five hidden neurons is fitted it does not result in a better fit.

According to what have been done so far for this assignment, one noticeable thing is that an increase in model complexity provides more ability for the model to fit with data. The increase in complexity between the network also makes the model more flexible and allows the model to perfectly fit with the observed data. However, the model can become too complex if the number of hidden neurons is too high. In this case this limit seem to be four hidden neurons, when this number of neurons is exceeded a worse model is obtained. That is because in some points it would be too difficult to train all data.

Appendix

R-code

```
## ### Assignment 1
## ## 1.1
## # "lambda" is the learning rate, "stopcond" the parameter for the stop condition
## spl <- function(x, y, lambda, weights, stopcond){
##   yHat <- 0
##   MisClass1 <- 0
##   MisClass2 <- 1
```

```
##   # Loop that runs until stopping condition is met.
##   while(abs(MisClass2 - MisClass1) > stopcond){
##     for(i in 1:nrow(x)){
##       # The weights are multiplied with i:th row
##       linComb <- sum(weights * x[i, ])
##       # Uses the sign function to classify the observation
##       yHat[i] <- sign(linComb)
##       # Updates the weights
##       weights <- weights + lambda*(y[i] - yHat[i]) * x[i, ]
##     }
##     # Calculates the misclassification value
##     # The misclass value for the latest iteration is saved for comparison with
##     # the next misclass value
##     MisClass2 <- MisClass1
##     confMat <- table(yHat, y)
##     MisClass1 <- 1 - sum(diag(confMat)) / sum(confMat)
##   }
##   RetU <- list(Wt = weights, miscTr = MisClass1)
##   return(RetU)
## }
## # The classify the test set with the obtained model the following code is used
## testSPL <- spl(x=inputs, y=train[,58], lambda=0.1, weights=init_weights, stopcond=0.01)
## yHatTest <- 0
## for(i in 1:nrow(testInput)){
##   # Uses the weights returned by the function on every row
##   linCombTest <- sum(testSPL$Wt * testInput[i,])
##   yHatTest[i] <- sign(linCombTest)
## }
## confMatTest <- table(yHatTest, test$Spam)
## MisClass3 <- 1 - sum(diag(confMatTest)) / sum(confMatTest)
Spambase <- read.csv2("C:/Users/Gustav/Documents/Machine-Learning/Lab8/spambaseshort.csv", sep=";")
n=dim(Spambase)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.7))
train=Spambase[id,]
test=Spambase[-id,]
## inputs <- data.frame(scale(train[, 1:57], center = TRUE, scale = TRUE))
## inputs <- data.frame(X=rep(1, 322), inputs)
## testInput <- data.frame(scale(test[, 1:57], center = TRUE, scale = TRUE))
## testInput <- data.frame(X=rep(1, 138), testInput)
##
## set.seed(7235)
## init_weights <- rnorm(n = ncol(train))
## testSPL <- spl(x=inputs, y=train[,58], lambda=0.1, weights=init_weights, until=0.01)
## yHatTest <- 0
## for(i in 1:nrow(testInput)){
##   linCombTest <- sum(testSPL$Wt * testInput[i,])
##   yHatTest[i] <- sign(linCombTest)
## }
## confMatTest <- table(yHatTest, test$Spam)
## MisClass3 <- 1 - sum(diag(confMatTest)) / sum(confMatTest)
## MisClass3
## set.seed(846)
```

```
## init_weightsv2 <- rnorm(n = ncol(train))
## testSPLv2 <- spl(x=inputs, y=train[,58], lambda=0.1, weights=init_weightsv2, until=0.01)
## yHatTest <- 0
## for(i in 1:nrow(testInput)){
##   linCombTest <- sum(testSPL$Wt * testInput[i,])
##   yHatTest[i] <- sign(linCombTest)
## }
## confMatTest <- table(yHatTest, test$Spam)
## MisClass4 <- 1 - sum(diag(confMatTest)) / sum(confMatTest)
## MisClass4
train[train==-1] <- 0
test[test==-1] <- 0

logi <- glm(Spam~.,data=train, family=binomial())
pred_logi <- data.frame(test$Spam, fitted=predict(logi, newdata=test, type="response"))
for(i in 1:length(pred_logi[,1])){
  if(pred_logi[i, 2] > 0.5){
    pred_logi[i, 2] = 1
  }else{
    pred_logi[i, 2] = 0
  }
}
confMat <- table(pred_logi)
MisClass <- 1 - sum(diag(confMat)) / sum(confMat)
confMat
### Assignment 2
## 2.1
FruitFly <- read.csv2("C:/Users/Gustav/Documents/Machine-Learning/Lab8/mortality.csv", sep=";")
DaySz <- data.frame(Day =scale(FruitFly$Day, scale = TRUE, center=TRUE))
FruitFlySz <- data.frame(LMR = FruitFly$LMR, Day = DaySz)

## 2.2
library(ggplot2)
qplot(y=LMR,x=Day, data=FruitFlySz, geom = "line", xlab="Day(standardized)",ylab="LMR")
library(neuralnet)
set.seed(7235)
network2_3 <- neuralnet(LMR ~ Day, data = FruitFlySz, hidden = 1, act.fct =
            "tanh", stepmax = 1e6, threshold = 0.1)
plot(network2_3)
plot(y=FruitFlySz$LMR, x=FruitFlySz$Day, type="l", lwd=2)
points(x=FruitFlySz$Day, unlist(network2_3$net.result), col="darkorange", type="l",
       lwd=2)
## 2.4
set.seed(7235)
network2_4 <- neuralnet(LMR ~ Day, data = FruitFlySz, hidden = 2, act.fct =
                          "tanh", stepmax = 1e6, threshold = 0.1)
# a)
plot(network2_4)
# b)
plot(y=FruitFlySz$LMR, x=FruitFlySz$Day, type="l", lwd=2)
points(x=FruitFlySz$Day, unlist(network2_4$net.result), col="darkorange", type="l",
       lwd=2)
## 2.5
```

```
# Four hidden neurons
set.seed(7235)
network2_5a <- neuralnet(LMR ~ Day, data = FruitFlySz, hidden = 4, act.fct =
                         "tanh", stepmax = 1e6, threshold = 0.1)
#plot(network2_5a)
plot(y=FruitFlySz$LMR, x=FruitFlySz$Day, type="l", lwd=2)
points(x=FruitFlySz$Day, unlist(network2_5a$net.result), col="darkorange", type="l",
       lwd=2)
# Five hidden neurons
set.seed(7235)
network2_5b <- neuralnet(LMR ~ Day, data = FruitFlySz, hidden = 5, act.fct =
                          "tanh", stepmax = 1e6, threshold = 0.1)
#plot(network2_5b)
plot(y=FruitFlySz$LMR, x=FruitFlySz$Day, type="l", lwd=2)
points(x=FruitFlySz$Day, unlist(network2_5b$net.result), col="darkorange", type="l",
       lwd=2)
##
```