Computer Lab 8

Thomas Zhang

2015-12-11

Assignment 1

We want to classify the e-mails as spam or not spam using the word and character frequencies and the uninterrupted capital lettering data found in the file `spambaseshort.csv`. To this end we choose to train a simple perceptron model using 70% of the data available. The settings to be used for the model are exactly as described in the lab instructions. We train two models, one with `set.seed(7235)` and one with `set.seed(846)` and find the misclassification rate of the test data in each case.

```
## [1] "Misclassification rate for test data: 0.0797"
```

```
## [1] "Misclassification rate for test data: 0.101"
```

We see that the simple perceptron model works well for this data set, with not very bad misclassification rates for the two seeds. Let us now compare the test misclassification rates with those produced by classification performed by a logistic regression model classifier.

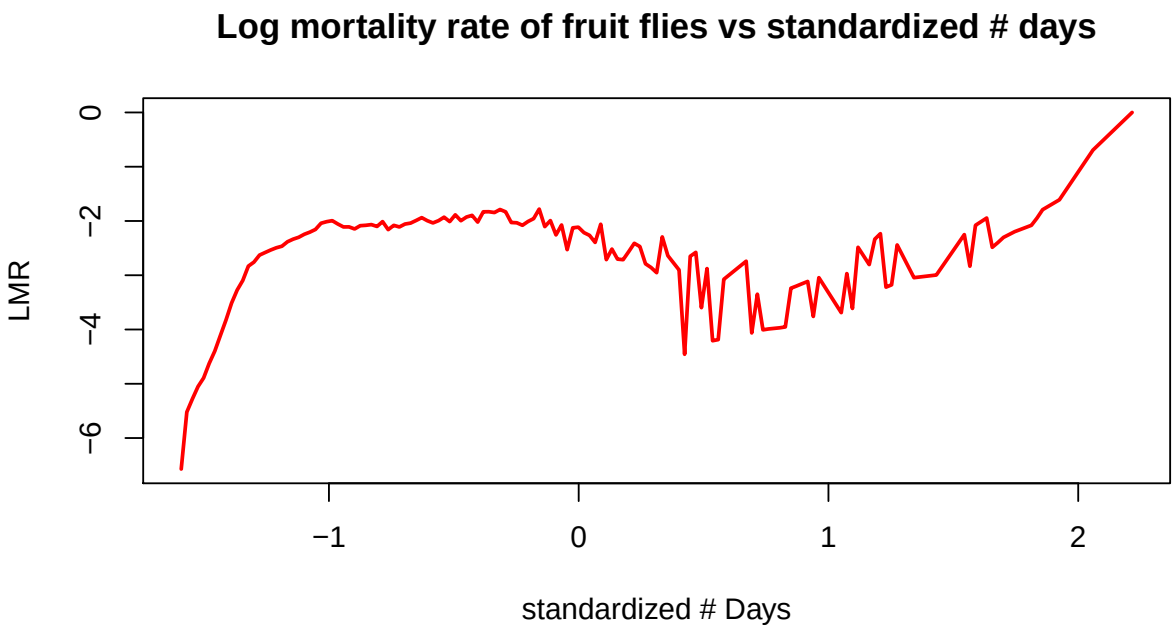
```
##          testresult
## testspam -1  1
##       -1 71 10
##       1   7 50
```

```
## [1] "Logistic regression misclassification rate: 0.123"
```

From my vantagepoint, it looks as if the simple perceptron model performs slightly better than the logistic regression model in terms of misclassification rate.

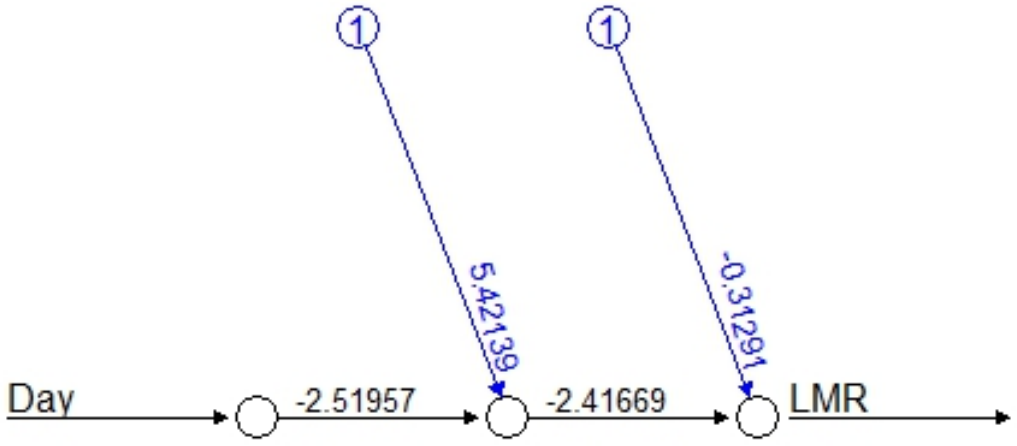
Assignment 2

We wish to perform nonlinear MLP neural network regression of Log mortality rate of fruitflies upon standardized day variable in the dataset `mortality.csv` using the package `neuralnet`. The line plot of the data looks like this:

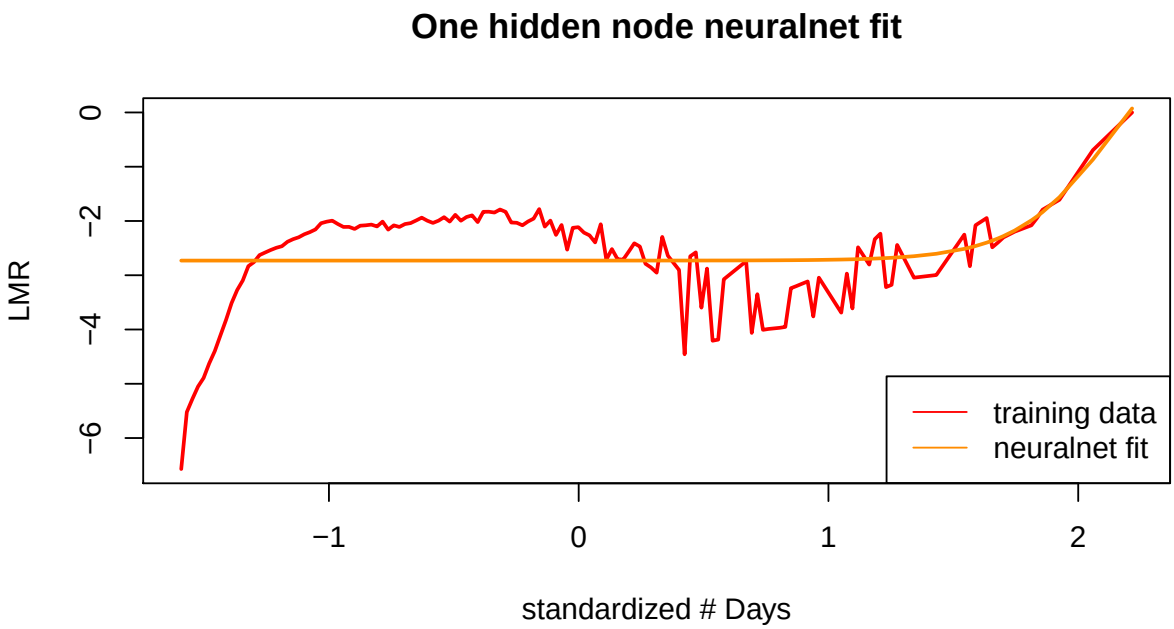


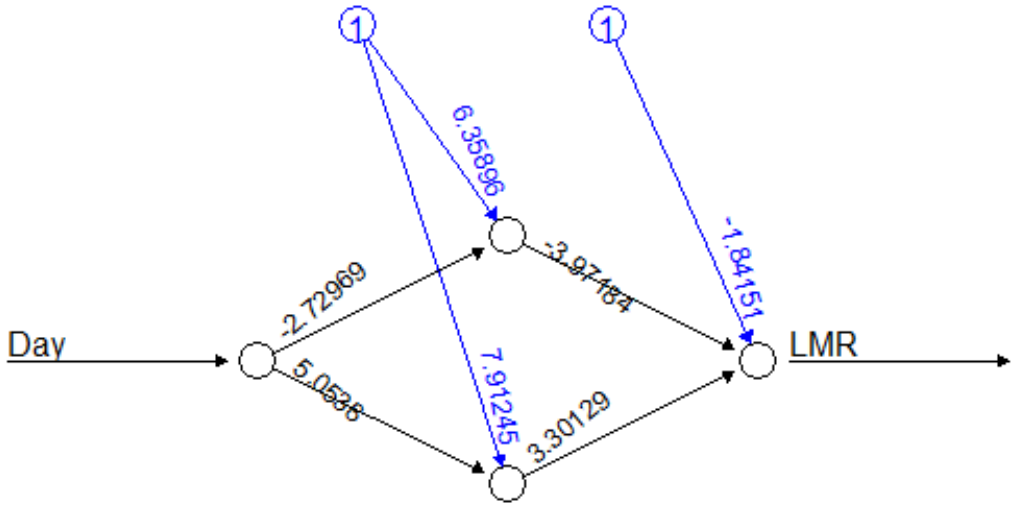
A first impression suggests that the regression fit indeed will not be linear, but rather non-linear.

We now fit neural networks with one, two, four and five hidden nodes, all using `set.seed(7235)`, and investigate their fits, relative SSE and steps taken.

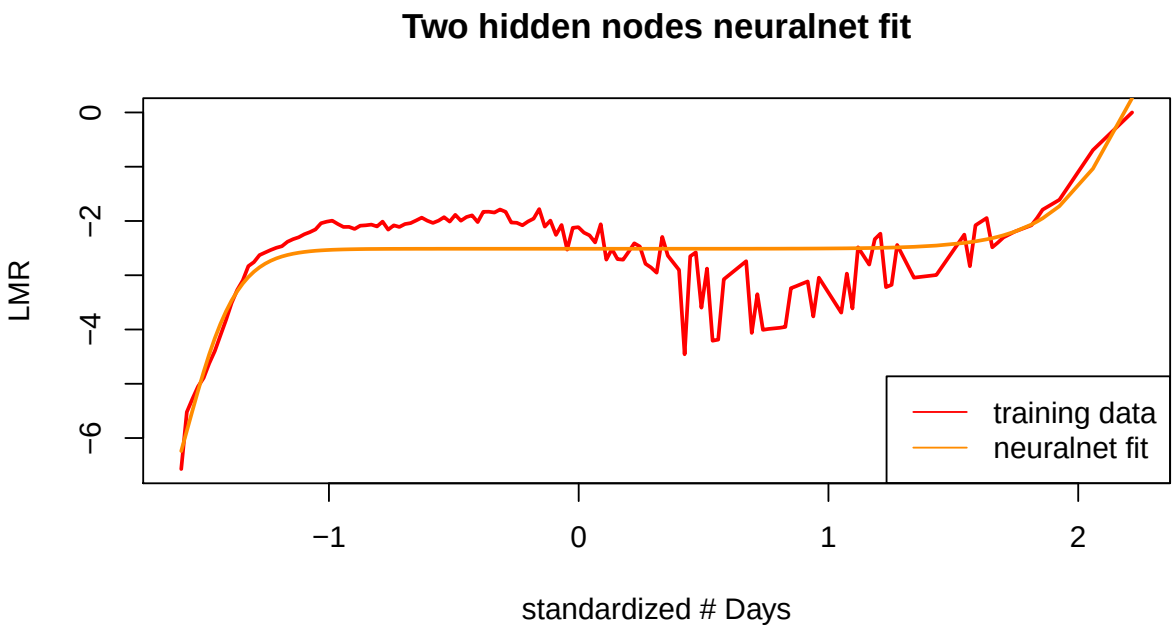


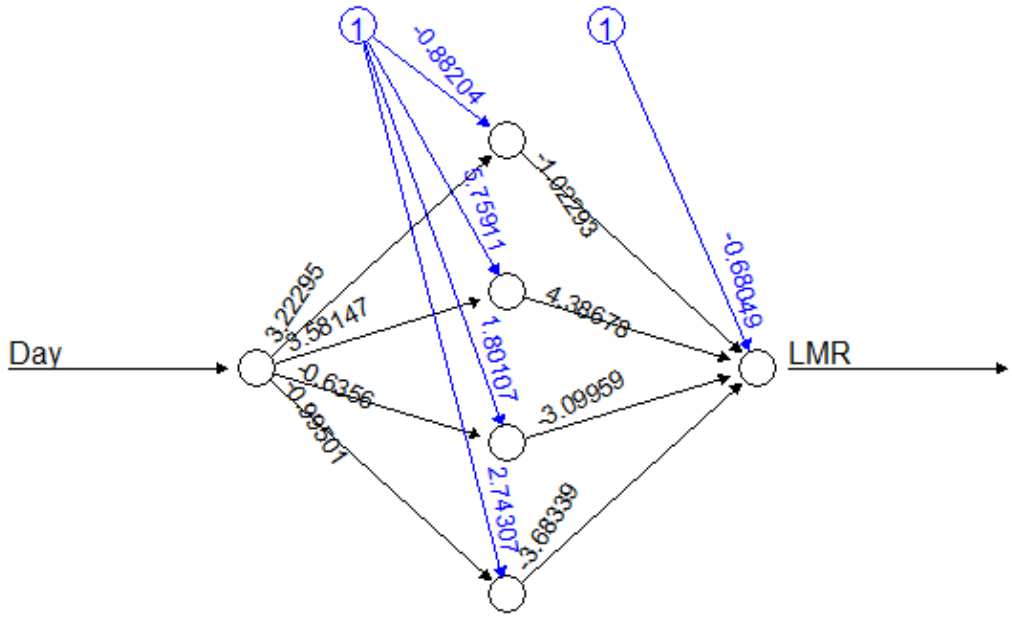
Error: 49.703647 Steps: 607

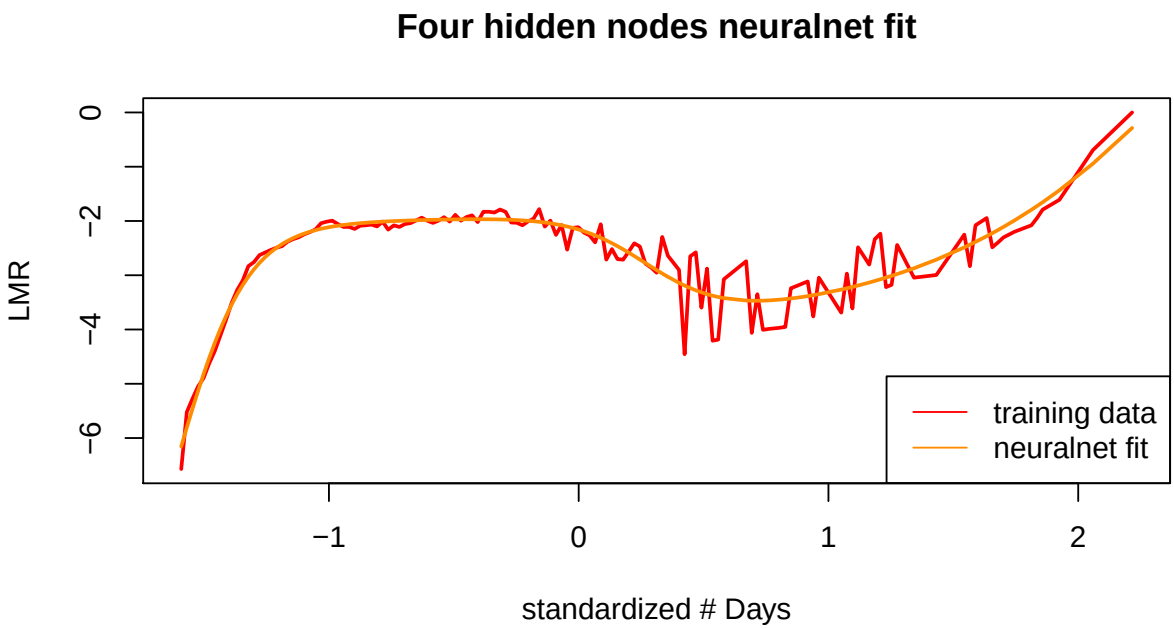


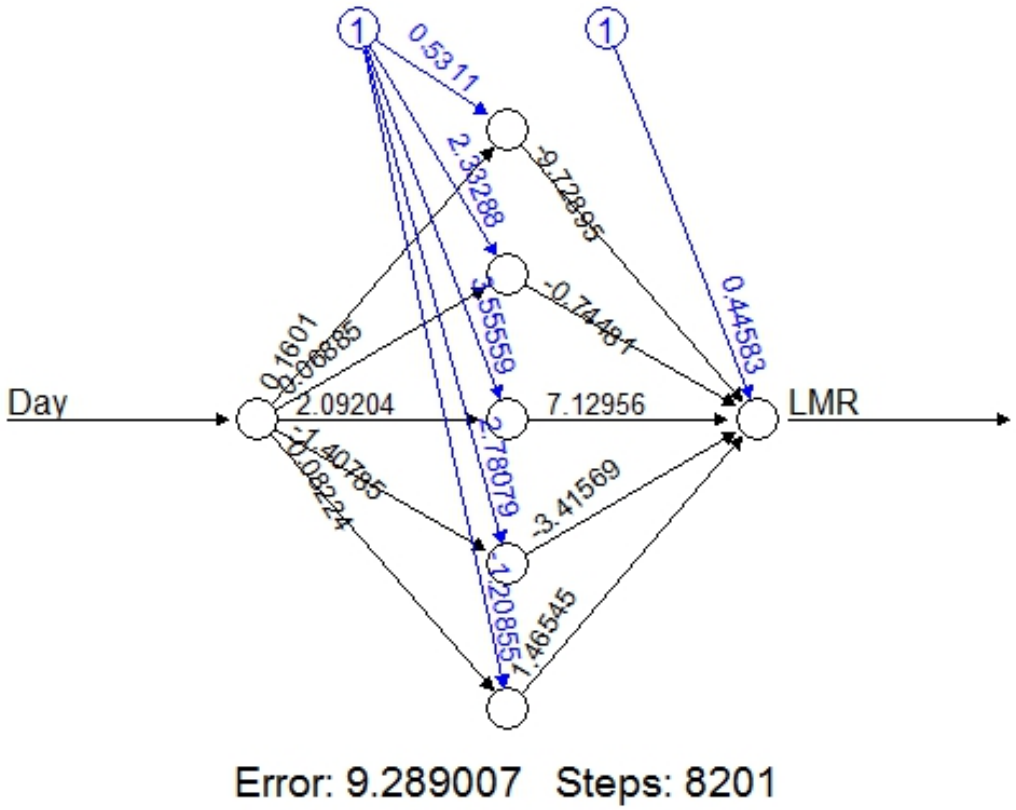


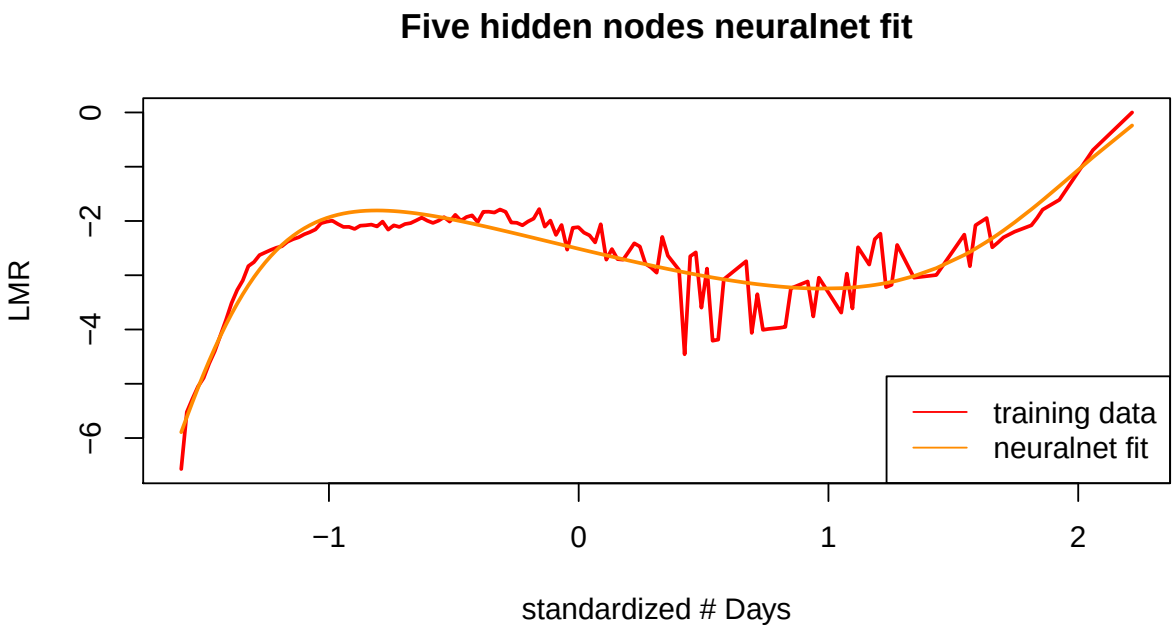
Error: 23.251744 Steps: 1859


Two hidden nodes neuralnet fit
LMR
0
−2
−4
−6
−1
0
1
2
training data
neuralnet fit
standardized # Days


1
1
-0.88204
-1.02293
5.75911
-0.68049
3.22295
3.58147
4.39678
1.80107
Day
-0.6356
-0.99501
3.09959
LMR
2.74307
-3.68339
Error: 6.212279 Steps: 2355


Four hidden nodes neuralnet fit
LMR
0
−2
−4
−6
−1
0
1
2
training data
neuralnet fit
standardized # Days


1
1
0.5311
2.33288
-9.72895
-0.74481
0.44583
0.1601
-0.06885
3.55559
Day
2.09204
7.12956
LMR
-1.40785
-0.08224
2.78079
-3.41569
-1.20855
1.46545
Error: 9.289007 Steps: 8201



We clearly see that the more nodes, the smaller the SSE becomes up to four hidden nodes. The neural network fit becomes overfitted with more hidden nodes than four. We also see that the number of steps taken increase with number of hidden nodes. It appears as if the first two models do not produce good fits, while the last two models do produce good fits. To me, the fit using four nodes looks like the best fit (it also has the best SSE). I'd venture to say that more hidden nodes lead to better ability to fit non-linearities in the fitted curve.

Appendix - R-Code

```
data <- read.csv2("C:/Users/Dator/Documents/R_HW/ML-lab-1/data/spambaseshort.csv")
truespam <- data[,ncol(data)]
data <- data[,-ncol(data)]
data <- scale(data)

set.seed(12345)
index <- sample(1:nrow(data),floor(0.7 * nrow(data)))
train <- data[index,]
test <- data[-index,]
train_truespam <- truespam[index]
test_truespam <- truespam[-index]

simple_perceptron_classif <- function(data,truespam,seed){

  data <- cbind(data,rep(1,nrow(data)))
  set.seed(seed)
  weights <- rnorm(ncol(data))
  output <- rep(0,nrow(data))
```

```
  for(i in 1:nrow(data)){
    for(j in 1:(ncol(data))){
      output[i] <- output[i] + weights[j] * data[i,j]
    }
    output[i] <- sign(output[i])
  }

  for(j in 1:(ncol(data))){
    for(i in 1:nrow(data)){
      weights[j] <- (truespam[i]- output[i]) * 0.1 * data[i,j] + weights[j]
    }
  }
  misclass <- 1 - length(which(output == truespam)) / nrow(data)
  prevmisclass <- 1
  counter <- 1



  while(prevmisclass - misclass > 0.01){
    prevmisclass <- misclass
    for(i in 1:nrow(data)){
      for(j in 1:(ncol(data))){
        output[i] <- output[i] + weights[j] * data[i,j]
      }
      output[i] <- sign(output[i])
    }

    for(j in 1:(ncol(data))){
      for(i in 1:nrow(data)){
        weights[j] <- (truespam[i]- output[i]) * 0.1 * data[i,j] + weights[j]
      }
    }
    misclass <- 1 - length(which(output == truespam)) / nrow(data)
    counter <- counter + 1
  }

  return(list(weights=weights,
              misclassification_rate=misclass,counter = counter))
}


simple_percept_exec <- function(testdata,truespam,weights){
  testdata <- cbind(testdata,rep(1,nrow(testdata)))
  output <- rep(0,nrow(testdata))
  for(i in 1:nrow(testdata)){
    for(j in 1:ncol(testdata)){
      output[i] <- output[i] + weights[j] * testdata[i,j]
    }
    output[i] <- sign(output[i])
  }
  misclass <- 1 - length(which(output == truespam)) / nrow(testdata)
  return(paste("Misclassification rate for test data:",signif(misclass,3)))
}
```

```
firstlist <- simple_perceptron_classif(train,train_truespam,7235)
simple_percept_exec(test,test_truespam,firstlist$weights)

secondlist <- simple_perceptron_classif(train,train_truespam,846)
simple_percept_exec(test,test_truespam,secondlist$weights)
data <- read.csv2("C:/Users/Dator/Documents/R_HW/ML-lab-1/data/spambaseshort.csv")
data$Spam[data$Spam < 0] <- 0
set.seed(12345)
index <- sample(1:nrow(data),floor(0.7 * nrow(data)))
train <- data[index,]
test <- data[-index,]
testspam <- test[,ncol(test)]
testspam[testspam == 0] <- -1
test <- test[,-ncol(test)]


logisticreg <- glm(Spam ~., family = binomial(link="logit"), data = train)
testresult <- predict(logisticreg,test)
testresult <- sign(testresult)
table(testspam,testresult)
paste("Logistic regression misclassification rate:",
      signif(1 - length(which(testresult == testspam)) / length(testspam),3))
library(neuralnet)
mort <- read.csv2("C:/Users/Dator/Documents/R_HW/ML-lab-1/data/mortality.csv")
stdday <- scale(mort[,1])
mort[,1] <- stdday
plot(LMR ~ Day, data = mort, type="l",col="red",lwd=2,
     main="Log mortality rate of fruit flies vs standardized # days",
     xlab="standardized # Days")
set.seed(7235)
model1 <- neuralnet(LMR ~ Day, data = mort, hidden = 1,
                    act.fct = "tanh", stepmax = 1e6, threshold = 0.1)

set.seed(7235)
model2 <- neuralnet(LMR ~ Day, data = mort, hidden = 2,
                    act.fct = "tanh", stepmax = 1e6, threshold = 0.1)

set.seed(7235)
model3 <- neuralnet(LMR ~ Day, data = mort, hidden = 4,
                    act.fct = "tanh", stepmax = 1e6, threshold = 0.1)

set.seed(7235)
model4 <- neuralnet(LMR ~ Day, data = mort, hidden = 5,
                    act.fct = "tanh", stepmax = 1e6, threshold = 0.1)

plot(LMR ~ Day, data = mort, type="l",col="red",lwd=2,
     main="One hidden node neuralnet fit",
     xlab="standardized # Days")
points(x=mort$Day, unlist(model1$net.result), col="darkorange", type="l",
       lwd=2)
legend("bottomright",legend = c("training data","neuralnet fit"),
       lty = c(1,1),col=c("red","darkorange"))
plot(LMR ~ Day, data = mort, type="l",col="red",lwd=2,
```

```
     main="Two hidden nodes neuralnet fit",
     xlab="standardized # Days")
points(x=mort$Day, unlist(model2$net.result), col="darkorange", type="l",
       lwd=2)
legend("bottomright",legend = c("training data","neuralnet fit"),
       lty = c(1,1),col=c("red","darkorange"))
plot(LMR ~ Day, data = mort, type="l",col="red",lwd=2,
     main="Four hidden nodes neuralnet fit",
     xlab="standardized # Days")
points(x=mort$Day, unlist(model3$net.result), col="darkorange", type="l",
       lwd=2)
legend("bottomright",legend = c("training data","neuralnet fit"),
       lty = c(1,1),col=c("red","darkorange"))
plot(LMR ~ Day, data = mort, type="l",col="red",lwd=2,
     main="Five hidden nodes neuralnet fit",
     xlab="standardized # Days")
points(x=mort$Day, unlist(model4$net.result), col="darkorange", type="l",
       lwd=2)
legend("bottomright",legend = c("training data","neuralnet fit"),
       lty = c(1,1),col=c("red","darkorange"))
## NA
```