TDDE01 - Lab 3

David Forslöf - davfo018, assignment 3
Martin Friberg - marfr370, assignment 1
Filip Frenning - filfr933, assignment 2

2020/12/13

1 Assignment 1.

Two files with data were used in order to predict temperatures for a specific location based on measurements from a number of other locations. The files contain information about weather stations and temperature measurements in the stations at different days and times. The data have been kindly provided by the Swedish Meteorological and Hydrological Institute (SMHI).

To decide the smoothing factors, e.g. the H-values, kernels were plotted for the different metrics. Since the kernels were supposed to be Gaussian, the appropriate function was used for the kernel which in this case is

$$k = \frac{1}{e^(u^2)}$$

where u is

$$\frac{x - x_n}{h}$$

The Kernels generated can be seen below.

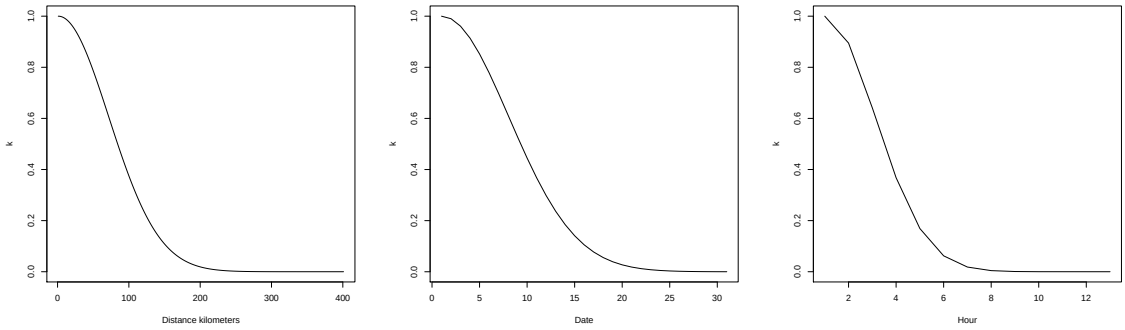

Figure 1: Kernels for the different metrics

By plotting the kernels with some different smoothing factors, it was concluded that reasonable values for the smoothing factors are: 100 000 for distance, 10 for the dates and 3 for the hours. These values resulted in that measurements from locations further away than approximately 20km, from dates that differ more than 20 days and from times that differ more than approximately 6 hours will not have a significant effect when we estimate the temperatures for our selected location during a specific date.

When predicting the temperatures, measurements posterior to the date of prediction and measurements on the same day were filtered out since those are not supposed to exist if we are to be realistic. An assumption was drawn that measurements on the same day are not to be considered even though they are prior to the hour that the temperature is predicted for.

A latitude value of 60.7256 and a longitude value of 15.0167 were chosen for the prediction in order to represent the village Leksand. The date chosen for the prediction was 2013-07-15.

Two different predictions of the temperatures were made, where one prediction was based on the sum of the three different kernels and one prediction was based on the product of the three kernels. The results of the predictions can be seen in the graphs below.

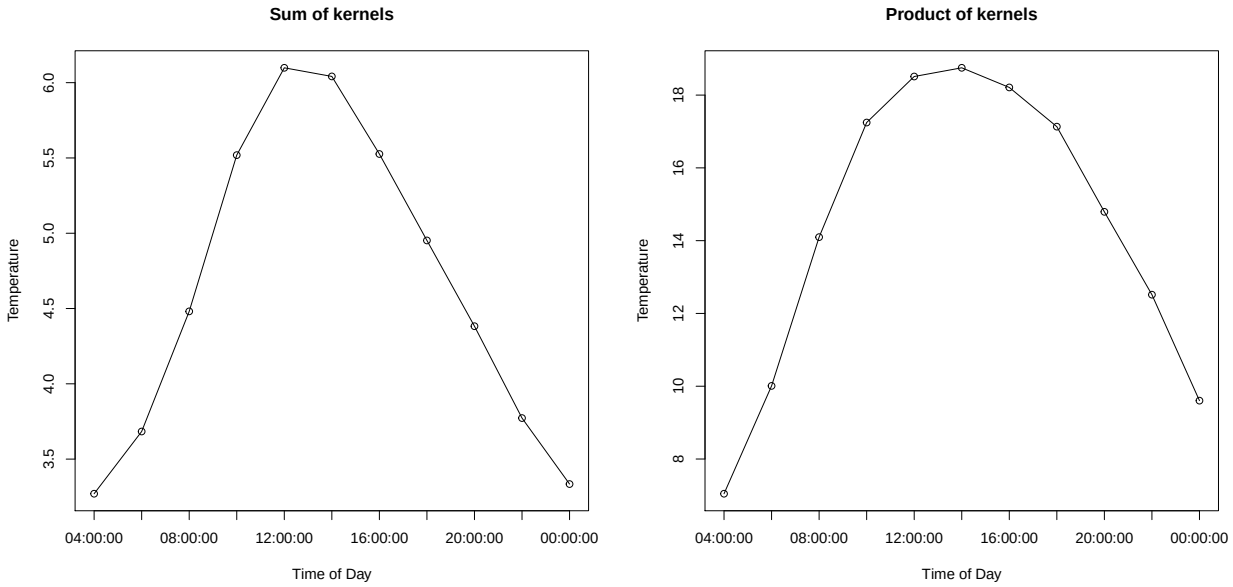

Figure 2: Temperature predictions based on the sum and the product of the kernels

The sum of the kernels predicts the temperatures way worse than the product of the kernels. This is due to that the sum of the kernels basically only need one of the three kernels to have a relatively high value in order to consider the measurement as a relevant one, whereas the product of the kernels actually looks to that all the values of a measurement for the kernels needs to have a relatively large value in order to make a difference in the prediction of the temperature.

2 Assignment 2.

2.1 Question 1

The fourth filter, filter 3, is the one with the lowest misclassification error. The prediction here is made on the test data but since we fit the model on the train, validation and test data combined we cannot use this. We found out optimal value of C by checking the validation error on the fitted training data. When predicting on the test data and checking the error, we have to first train and fit the model on only the training and validation data.

The filter0 is not chosen since that is using only the training data to fit the model. By also comparing the err_va and err0 we see that they are the same since we perform the exact same fitting and predicting data sets.

The third filter, filter 1, is predicting using the test data which is good. But the fitted filter, just as filter0, is only fitting the model using the training data. Therefore this filter is not an accurate depiction since we want to include all data that is not test data in the fitting.

We therefore choose filter2 to return to the user, since it is trained and fitted using both training and validation data and is predicting on the test data. We then get the correct model and filter.

2.2 Question 2

The estimate of the generalization error of the filter returned is err2, since we use the test data to make the prediction on the model, i.e. the unseen data. This prediction is the so called out-of-sample prediction. By doing this and then evaluating the error we get the so called out-of-sample error, which is also called the generalization error. In this case the generalization error for filter2 is err2 which is equal to 0.08364544.

3 Assignment 3.

The assignment was to train a neural network to learn the trigonometric sinus curve, that is predict $\sin(x)$ from x. First, 500 points were at random sampled uniformly in the interval $[0, 10]$. From this dataset, 25 points were used for training and the rest for testing. For the algorithm 2 layers were chosen, with 5 hidden layers in the first layer and 7 hidden layers in the second. The neural network is presented below.

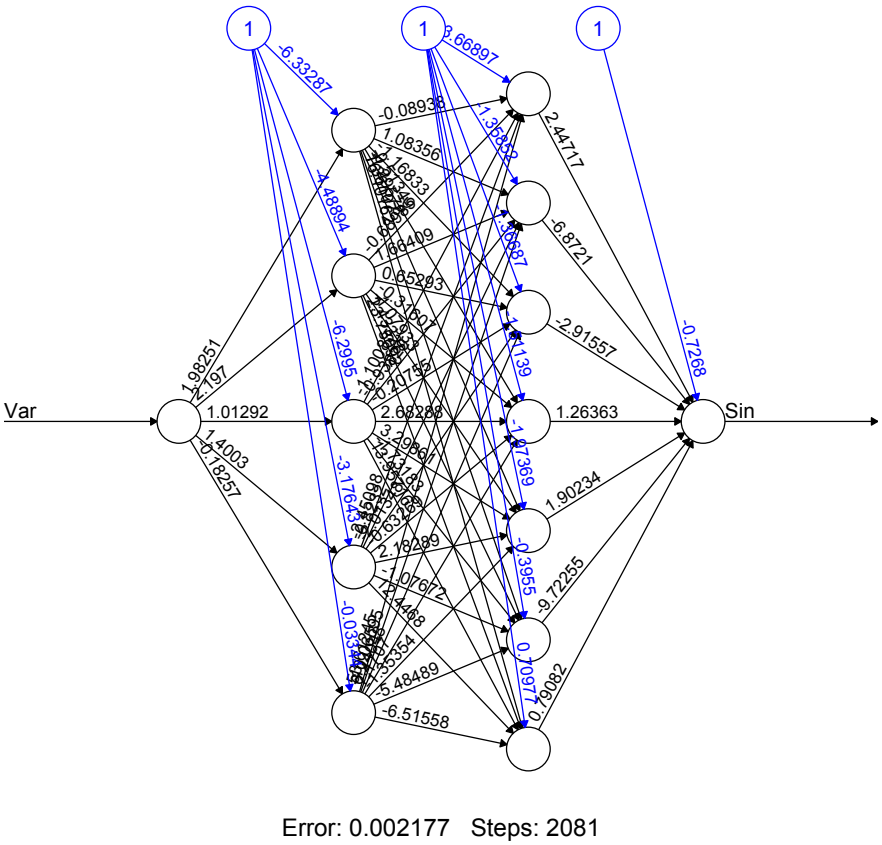

Figure 3: Neural Network

A prediction was made on the test data which produced the following graph. We can see that the network somewhat accurately predicts the sinus curve in the interval $[0, 10]$.

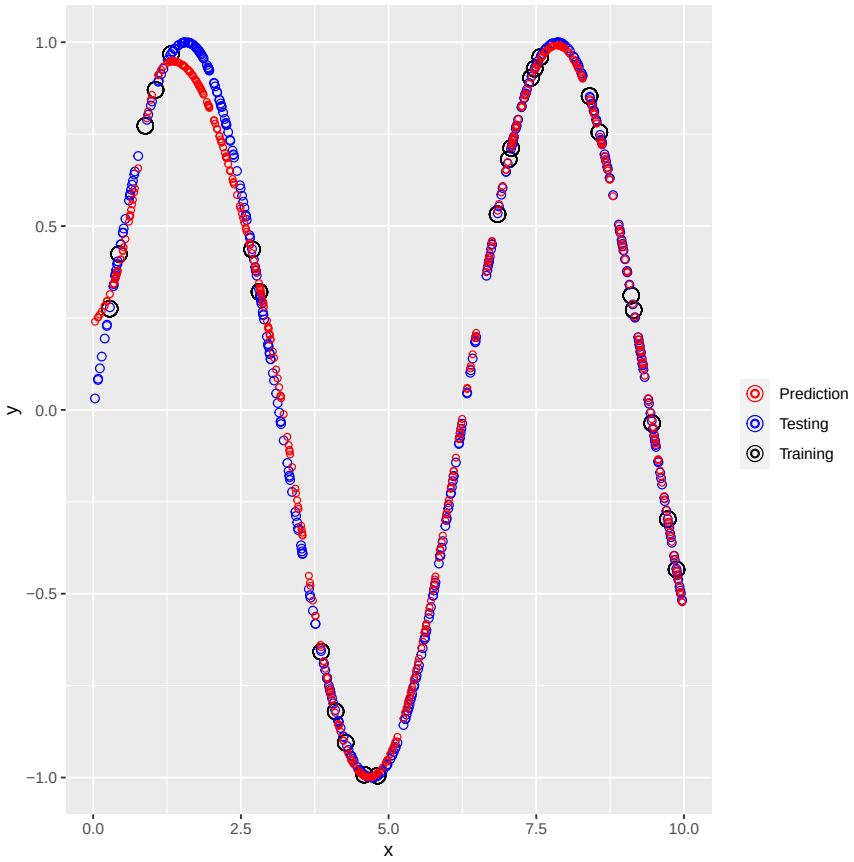

Figure 4: Predicted values on the interval $[0, 10]$

Thereafter, another dataset which contained 500 points was at random sampled uniformly in the interval $[0, 20]$. A prediction was then made on this validation set which produced the graph below. As seen in the graph, the model was not successful at predicting the sinus curve on the interval $[10, 20]$ since it was only trained to see the interval $[0, 10]$. In other words, a supervised model is not good at extrapolating outside the training set.

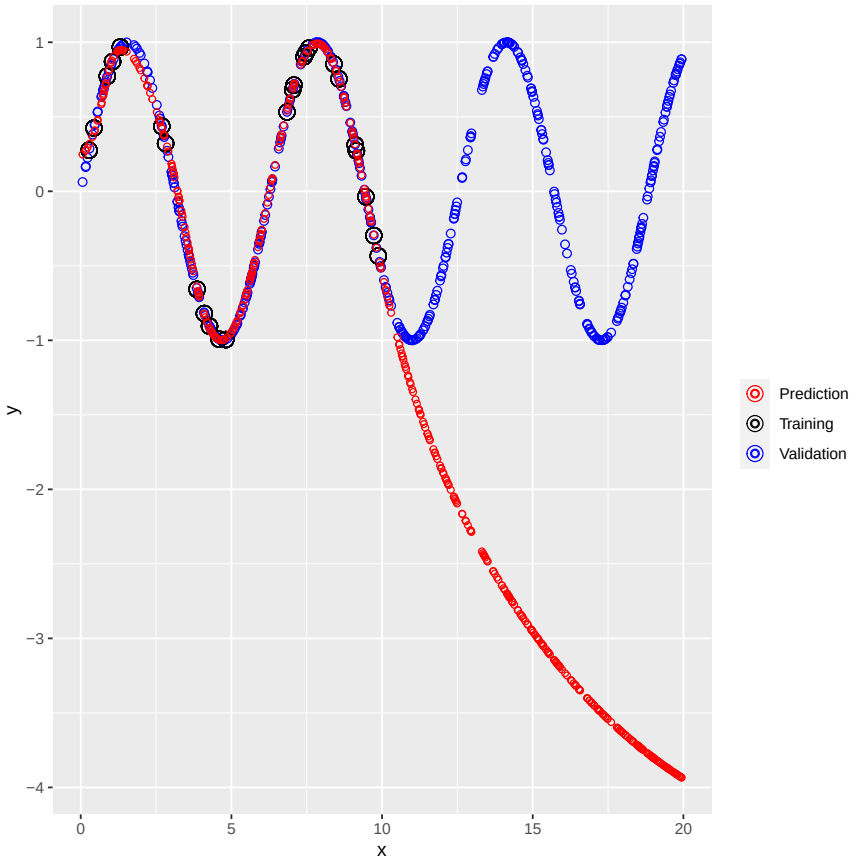

Figure 5: Predicted values on the interval $[0, 20]$

A model was then trained on a dataset containing 500 points that was random sampled uniformly in the interval $[0, 10]$ and then predicted on a similar dataset on the interval $[0, 20]$. However, this time the neural network was trained to predict x from $\sin(x)$. The network did not converge with the earlier proposed parameters for number of hidden layers and number of layers, so this was reduced to 6 hidden layers with 1 layer. The model is in this case completely unable to predict x from $\sin(x)$ which is expected, since for a single input values there are multiple output that are correct.

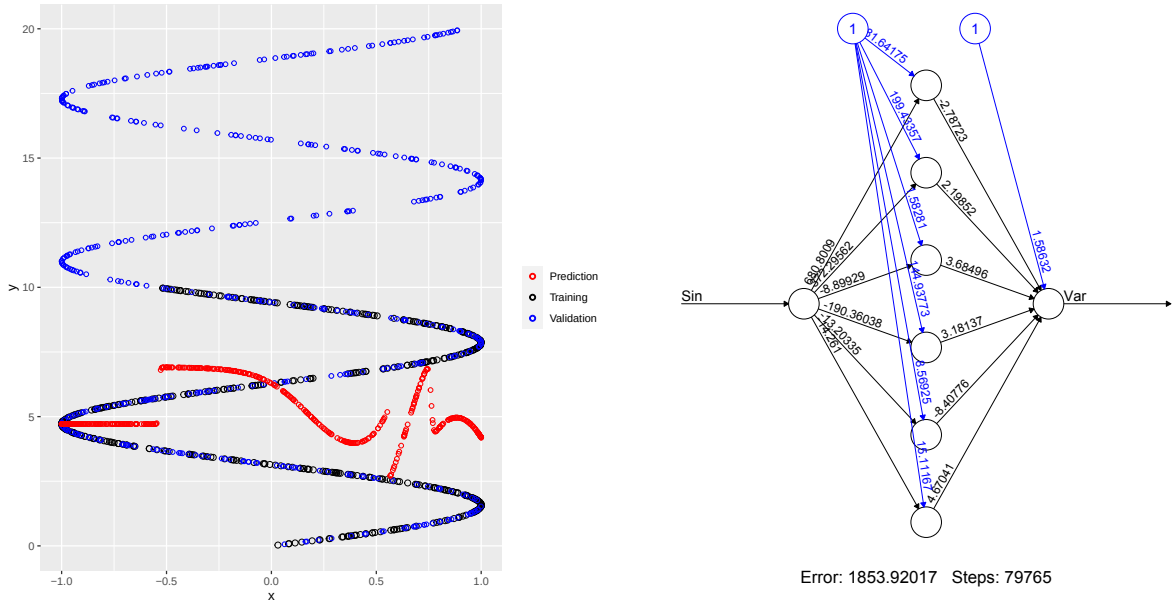

Figure 6: Predicted values and the neural network

4 Appendix

4.1 Code for Assignment 1

```
setwd("~/Desktop/TDDE01/tdde01/lab3/ass1")
set.seed(1234567890)
library(geosphere)

## We are not to use measurements from the same day at all basically (according to lab assistant)
################################################
###############   DISTANCE      ################
#  Distance between days takes into account only the days
#(not considering the year). For instance, 9 dec of one year is
#expected to be very much similar to 9 dec of the previous/next year.
#So they would be equal
################################################

# H-values based on plot below
#########
# H-values (smoothing factors) are based on "educated guesses, e.g. we plot the kernel functions
# for different H-values and use the H-values that gives us a good kernel
#########
#The smoothing factors makes measurements count to a limit of ca. double the amount of the smoothing fa
h_distance <- 100000
h_date <- 10
h_time <- 3
H_vals <- data.frame(h_distance, h_date, h_time)

#These values can be set to what ever (needs to be a location though)
# Location for the forecast -> Leksand
latitude <- 60.7256
longitude <- 15.0167


# Target date # date <- "2013-11-04" # The date to predict (up to the students)
date <- "2013-07-15"
targets <- data.frame(latitude, longitude, date)


#These files contain information about weather stations and temperature measurements in
#the stations at different days and times. The data have been kindly provided by the
#Swedish Meteorological and Hydrological Institute (SMHI).
stations = read.csv("stations.csv", fileEncoding="latin1")
temps = read.csv("temps50k.csv")
st = merge(stations,temps,by="station_number")

# times <- c("04:00:00", "06:00:00", ..., "24:00:00")
times <- c("04:00:00", "06:00:00", "08:00:00", "10:00:00", "12:00:00", "14:00:00", "16:00:00", "18:00:0

# Plot k against distance to find good h-values, e.g these values are just references to see how far aw
distance <- seq(0,400000,1000) #(default is meters -> plotting 40km) e.g our parameters that are return
date_diff <- seq(0,30,1)
time_diff <- seq(0,12,1)
```

```r
#######    Gaussian kernel: k(u) = exp(- u 2) where       is the Euclidean norm. #####
###### distance in this function resembles the distance to the location we want to forecast
###### Plotting to decide reasonable h-values
# Plot weights of Distance kernel
plotKernalDistance <- function(distances, h) {
  u <- distances/h
  k <- 1/exp(u^2) #same as exp(-u^2) but more intuitive
  pdf('Distance_Kernel.pdf')
  plot(k, type="l", xlab = "Distance kilometers")

  dev.off()
}

# Plot weights of Date kernel
plotKernalDate <- function(date_diff, h) {
  u <- date_diff/h
  k <- 1/exp(u^2)
  pdf('Date_Kernel.pdf')
  plot(k, type="l", xlab = "Date")
  dev.off()
}

# Plot weights of Hour kernel
plotKernalHour <- function(time_diff, h) {
  u <- time_diff/h
  k <- 1/exp(u^2)
  pdf('Hour_Kernel.pdf')
  plot(k, type="l", xlab = "Hour")
  dev.off()
}


plotKernalDistance(distance, H_vals$h_distance)
plotKernalDate(date_diff, H_vals$h_date)
plotKernalHour(time_diff, H_vals$h_time)

# Filter dates in data posterior to target date
filterDataByDate <- function(data, date) {
  #### Drop rows with date > selected date
  filtered <-data[!(as.Date(data$date) > as.Date(date)),]
  return (filtered)
}


#Data that we are allowed to use for our prediction
FilteredData_date=filterDataByDate(st, date)

# Gaussian Kernel for distance
Kernel_distance <- function(data, target, h) {
  distanceDifference <- distHaversine(data.frame(data$longitude, data$latitude), target)
  u <- distanceDifference/h
  k <- 1/exp(u^2)
```

```
  return(k)
}

# Gaussian Kernel for day
Kernel_day <- function(data, date1, h) {
  dayDifference <-as.numeric(as.Date(data$date) - as.Date(date1), unit="days")
  dayDifference <- abs(dayDifference)
  u <- dayDifference/h
  k <- 1/exp(u^2)
  return(k)
}

# Gaussian Kernel for hours
Kernel_hours <- function(data, time, h) {

  timeDifference <- as.numeric(difftime(strptime(data$time , format = "%H:%M:%S"),
                        strptime(time , format = "%H:%M:%S")))


  timeDifference = timeDifference/3600
  for(i in 1:length(timeDifference)){
    if (timeDifference[i]<(-12)){
      timeDifference[i]<-timeDifference[i] %%24
    }
    else if (timeDifference[i]>12){
      timeDifference[i]<-abs(timeDifference[i] - 24)
    }
  }

  u <- timeDifference/h
  k <- 1/exp(u^2)
  return(k)
}


# Multiplication and Summation of kernels
Est_temp <- function(data, target, smoothH, times) {
  #Basically a vector of values between 0-1 depending on the location of the data we are comparing to.
  k_distance <- Kernel_distance(data, c(target$longitude[1], target$latitude[1]), smoothH$h_distance[1])

  #Basically a vector of values between 0-1 depending on the date of the data we are comparing to. The
  k_day <- Kernel_day(data, target$date[1], smoothH$h_date[1])

  #Constructing vectors that we can store the temperature values based on the summation and the product
  temp_sum <- vector(length = length(times))
  temp_mult <- vector(length = length(times))

  for (i in 1:length(times)){
    #Basically a vector of values between 0-1 depending on the time of the data we are comparing to. Th
    k_hour <- Kernel_hours(data, times[i], smoothH$h_time[1])

    #Summation of kernels
    k_tot_sum <- k_distance + k_day + k_hour
```

```
    #Product of kernels
    k_tot_mult <- k_distance * k_day * k_hour

    #Vectors of temperature values between 4am to 24pm
    #Kernel-weighted average over all the points slide 8, 3a
    temp_sum[i] <- sum(k_tot_sum %*% data$air_temperature)/sum(k_tot_sum)
    temp_mult[i] <- sum(k_tot_mult %*% data$air_temperature)/sum(k_tot_mult)
  }
  return(list(temp_sum = temp_sum, temp_mult = temp_mult))
}


Estimated_temps <- Est_temp(FilteredData_date, targets, H_vals, times)

pdf('Kernel_sum.pdf')
plot(Estimated_temps$temp_sum, xaxt = "n", xlab="Time of Day", ylab="Temperature", type="o", main = "Su
axis(1, at=1:length(times), labels=times)
dev.off()

pdf('Kernel_produkt.pdf')
plot(Estimated_temps$temp_mult, xaxt = "n", xlab="Time of Day", ylab="Temperature", type="o", main = "Pr
axis(1, at=1:length(times), labels=times)
dev.off()
```

4.2 Code for Assignment 2

```
library(kernlab)
set.seed(1234567890)

data(spam)

index <- sample(1:4601)
tr <- spam[index[1:3000], ]
va <- spam[index[3001:3800], ]
trva <- spam[index[1:3800], ]
te <- spam[index[3801:4601], ]

by <- 0.3
err_va <- NULL
for(i in seq(by,5,by)){
  filter <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=i)
  mailtype <- predict(filter,va[,-58])
  t <- table(mailtype,va[,58])
  err_va <-c(err_va,(t[1,2]+t[2,1])/sum(t))
}

filter0 <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
mailtype <- predict(filter0,va[,-58])
t <- table(mailtype,va[,58])
err0 <- (t[1,2]+t[2,1])/sum(t)
err0

filter1 <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
mailtype <- predict(filter1,te[,-58])
t <- table(mailtype,te[,58])
err1 <- (t[1,2]+t[2,1])/sum(t)
err1

filter2 <- ksvm(type~.,data=trva,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
mailtype <- predict(filter2,te[,-58])
t <- table(mailtype,te[,58])
err2 <- (t[1,2]+t[2,1])/sum(t)
err2

filter3 <- ksvm(type~.,data=spam,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
mailtype <- predict(filter3,te[,-58])
t <- table(mailtype,te[,58])
err3 <- (t[1,2]+t[2,1])/sum(t)
err3
```

4.3 Code for Assignment 3

```
library(neuralnet)
library(ggplot2)

# TASK 1
set.seed(1234567890)
Var <- runif(500, 0, 10)
sin <- data.frame(Var, Sin=sin(Var))
train <- sin[1:25,]
test <- sin[26:500,]

n_layers=2
n_hidden=c(5,7)

set.seed(1234567890)
winit <- sample(x = seq(from=-1, to=1, by=0.01), size=n_layers)

nn <- neuralnet(Sin ~ ., data=train, hidden=n_hidden, startweights=winit)
pred <- predict(nn, test)

data_frame <- data.frame(x1=train$Var,
                         y1=train$Sin,
                         x2=test$Var,
                         y2=pred[,1],
                         x3=test$Var,
                         y3=test$Sin)

ggplot(data_frame) +
  geom_point(aes(x=x1, y=y1, color='Training'), fill=NA, shape=21, size=4) +
  geom_point(aes(x=x3, y=y3, color='Testing'), fill=NA, shape=21, size=2) +
  geom_point(aes(x=x2, y=y2, color='Prediction'), fill=NA, shape=21) +
  scale_color_manual(values = c("Testing" = 'blue',
                                'Training' = 'black',
                                'Prediction' = 'red')) +
  labs(color="") + xlab("x") + ylab("y")


# TASK 2
set.seed(1234567890)
Var <- runif(500, 0, 20)
valid <- data.frame(Var, Sin=sin(Var))
pred <- predict(nn, valid)

data_frame <- data.frame(x1=valid$Var,
                         y1=valid$Sin,
                         x2=valid$Var,
                         y2=pred[,1],
                         x3=train$Var,
                         y3=train$Sin)


ggplot(data_frame) +
  geom_point(aes(x=x1, y=y1, color='Validation'), fill=NA, shape=21, size=2) +
```

```
  geom_point(aes(x=x3, y=y3, color='Training'), fill=NA, shape=21, size=4) +
  geom_point(aes(x=x2, y=y2, color='Prediction'), fill=NA, shape=21) +
  scale_color_manual(values = c("Validation" = 'blue',
                                'Training' = 'black',
                                'Prediction' = 'red')) +
  labs(color="") + xlab("x") + ylab("y")


# TASK 3
set.seed(1234567890)
Var <- runif(500, 0, 10)
train <- data.frame(Var, Sin=sin(Var))
set.seed(1234567890)
Var <- runif(500, 0, 20)
valid <- data.frame(Var, Sin=sin(Var))

n_layers=1
n_hidden=c(6)

set.seed(1234567890)
winit <- sample(x = seq(from=-1, to=1, by=0.01), size=n_layers)

nn <- neuralnet(Var ~ Sin, data=train, hidden=n_hidden, startweights=winit)
pred <- predict(nn, newdata=valid)

data_frame <- data.frame(x1=valid$Var,
                         y1=valid$Sin,
                         y2=valid$Sin,
                         x2=pred[,1],
                         x3=train$Var,
                         y3=train$Sin)

ggplot(data_frame) +
  geom_point(aes(x=x3, y=y3, color='Training'), fill=NA, shape=21, size=2) +
  geom_point(aes(x=x1, y=y1, color='Validation'), fill=NA, shape=21) +
  geom_point(aes(x=x2, y=y2, color='Prediction'), fill=NA, shape=21) +
  scale_color_manual(values = c("Validation" = 'blue',
                                'Training' = 'black',
                                'Prediction' = 'red')) +
  labs(color="") + xlab("x") + ylab("y")
```