732A99/732A68/ TDDE01 Machine Learning

Computer lab 3 block 1

Johannes Hedström, Mikael Montén & Siddhesh Sreedar

STIMA
Department of Computer and Information Science
Linköpings universitet

2023-12-17

Contents

1 Contribution of work

Mikael did assignment 1, Siddhesh did assignment 3, Johannes did assignment 2.

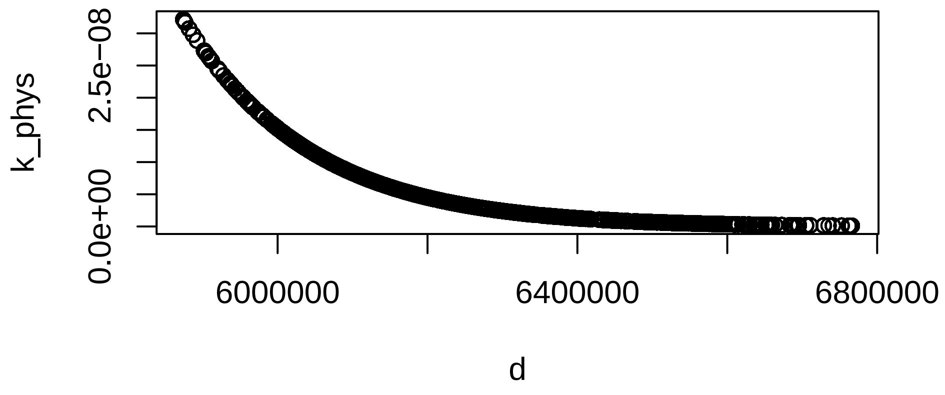
2 Assignment 1 - Kernels

Implement a kernel method to predict the hourly temperatures for a date and place in Sweden. To do so, you are provided with the files stations.csv and temps50k.csv. 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). You are asked to provide a temperature forecast for a date and place in Sweden. The forecast should consist of the predicted temperatures from 4 am to 24 pm in an interval of 2 hours. Use a kernel that is the sum of three Gaussian kernels:

- The first to account for the physical distance from a station to the point of interest. For this purpose, use the function distHaversine from the R package geosphere.
- The second to account for the distance between the day a temperature measurement was made and the day of interest.
- The third to account for the distance between the hour of the day a temperature measurement was made and the hour of interest

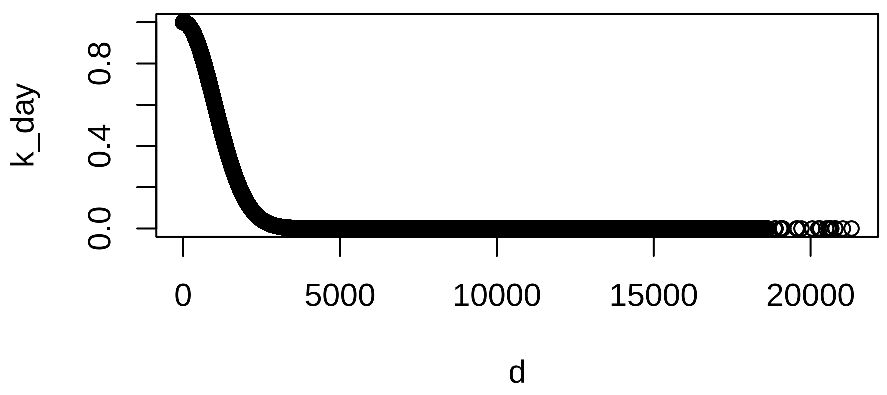
Choose an appropriate smoothing coefficient or width for each of the three kernels above. No cross-validation should be used. Instead, choose manually a width that gives large kernel values to closer points and small values to distant points. Show this with a plot of the kernel value as a function of distance. Help: Note that the file temps50k.csv may contain temperature measurements that are posterior to the day and hour of your forecast. You must filter such measurements out, i.e. they cannot be used to compute the forecast. Finally, repeat the exercise above by combining the three kernels into one by multiplying them, instead of summing them up. Compare the results obtained in both cases and elaborate on why they may differ. The only R package that is allowed to solve this assignment is the geosphere package (specifically, the function distHaversine). Feel free to use the template below to solve the assignment.

2.0.1 Physical distance kernel



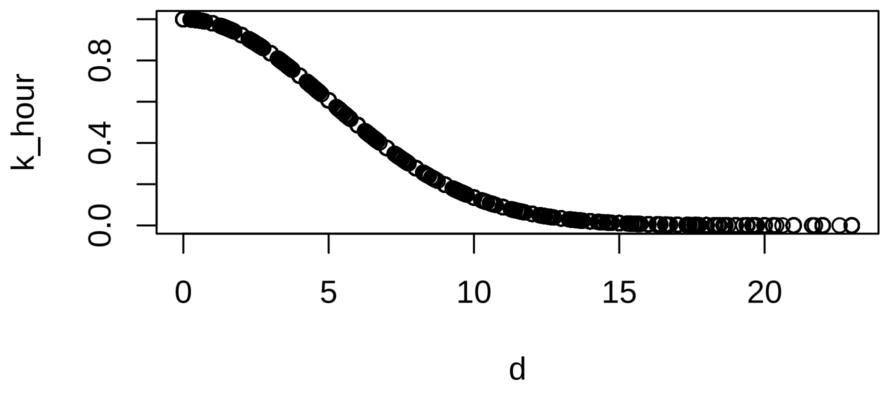
$L = 1000000$ seems to give good results for the physical distance kernel.

2.0.2 Day distance kernel



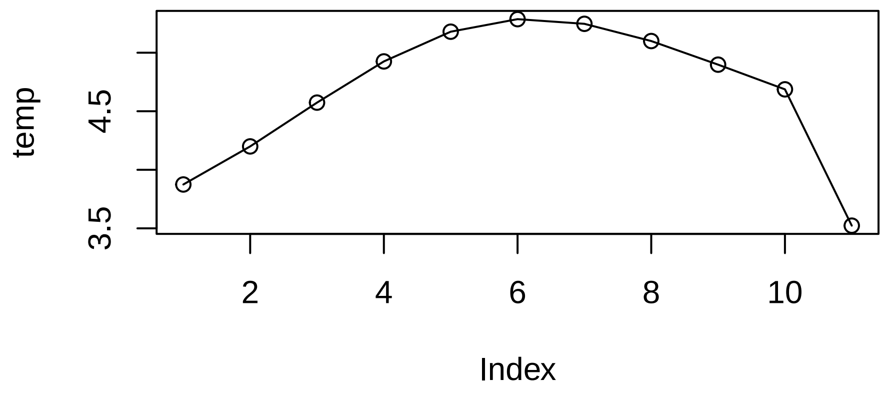
$L = 1000$ gives good results.

2.0.3 Hours distance kernel



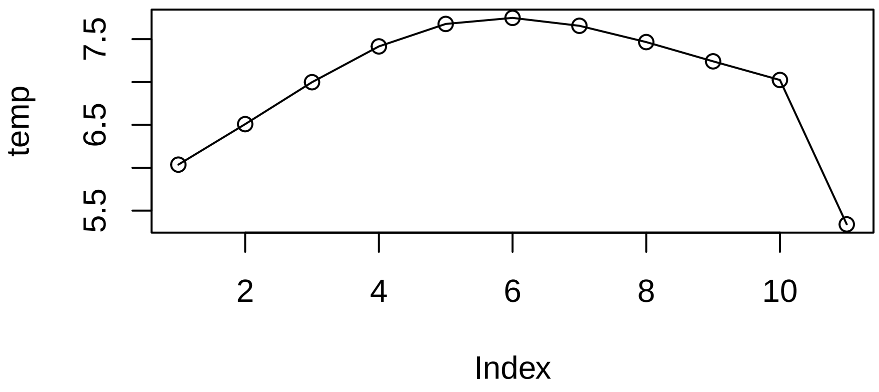
$L = 5$ gives good result for the hour kernel.

2.0.4 Results and comparison



```
##        Time Temperature
## 1  04:00:00    3.874531
## 2  06:00:00    4.199762
## 3  08:00:00    4.573691
## 4  10:00:00    4.925723
## 5  12:00:00    5.179530
## 6  14:00:00    5.286536
## 7  16:00:00    5.246694
## 8  18:00:00    5.099618
## 9  20:00:00    4.898059
## 10 22:00:00    4.687061
## 11 00:00:00    3.523048
```

Above is the predicted temperature when using a kernel that is the sum of all the different kernels. It has an increasing trend for the first half of the graph, i.e. from 04:00 AM til about 2:00 PM. Temperature maxes out at this time as well at 5.28 degrees celsius.



```
##        Time Temperature
## 1  04:00:00    6.036249
## 2  06:00:00    6.508820
## 3  08:00:00    6.998141
## 4  10:00:00    7.415566
## 5  12:00:00    7.677242
## 6  14:00:00    7.747944
## 7  16:00:00    7.655767
## 8  18:00:00    7.465763
## 9  20:00:00    7.241403
## 10 22:00:00    7.023618
## 11 00:00:00    5.339459
```

The above shows the kernel that is a product of the three computed kernels. It follows the same general shape as the summed kernel but has a slightly larger magnitude in its values. It also shows a rising trend for the first half of the day, and then decreases. Max temperature is at 2:00 pm and is predicted at 7.75 celsius.

Both kernels has a similarly predicted shape alas behavior of the temperature over the day, but product kernel is generally larger. This is likely due to a few reasons that relate to that a product of values is more sensitive to extreme values or other conditions that affect the overall result more than an addition or summed values would do. This can relate to outliers, which can disproprotionetely affect the product, or the multiplication of the kernels values that result in larger differences if they dont differ alot numerically unless multiplicated, i.e. 1+0.1 versus 1*0.1.

3 Assignment 2 - SVM

The code in the file Lab3Block1 2021 SVMs St.R performs SVM model selection to classify the spam dataset. To do so, the code uses the function ksvm from the R package kernlab, which also includes the spam dataset. All the SVM models to select from use the radial basis function kernel (also known as Gaussian) with a width of 0.05. The C parameter varies between the models. Run the code in the file Lab3Block1 2021 SVMs St.R and answer the following questions.

3.1 1

Which filter do you return to the user ? filter0, filter1, filter2 or filter3? Why?

```
## [1] 0.0675
```

```
## [1] 0.08489388
```

```
## [1] 0.082397
```

```
## [1] 0.02122347
```

Filter2 is trained on training and validation data, and gets a lower score on the test data than filter0/1(they are the same filter, the error is calculated on on different data). And as filter3 is trained on all data then tested on a part of it, its not the generalized or validation error that's calculated. So we could chose filter2 as it got the lowest error on the test data 0.082397 vs filter0/1 with 0.08489388. So one could think in this case it was better to just use the train / test split as filter0 underfitted the model a bit when only using the training data and with more data, filter 2 was able to improve the predictions on the test data. But as the c parameter is validated from testing different c values on the training data and validated on the validation data, which also includes in the training in filter2, a bias is included. Therefore filter1(same as filter0) should be returned to the user.

3.2 2

What is the estimate of the generalization error of the filter returned to the user? err0, err1, err2 or err3? Why?

Table 1: Generalization error of the returned filter(1)

x
0.0848939

As filter1 is the returned to the user and err1 is for the testdata, its the Generalized error for the returned filter.

3.3 3

Once a SVM has been fitted to the training data, a new point is essentially classified according to the sign of a linear combination of the kernel function values between the support vectors and the new point. You are asked to implement this linear combination for filter3. You should make use of the functions alphaindex, coef and b that return the indexes of the support vectors, the linear coefficients for the support vectors, and the negative intercept of the linear combination. See the help file of the kernlab package for more information. You can check if your results are correct by comparing them with the output of the function predict where you set type = "decision". Do so for the first 10 points in the spam dataset. Feel free to use the template provided in the Lab3Block1 2021 SVMs St.R file.

$$\hat{\alpha} = \begin{bmatrix} \hat{\alpha}_1 \\ \hat{\alpha}_2 \\ \vdots \\ \hat{\alpha}_n \end{bmatrix}$$

$$\hat{y}(x_\star) = \hat{\alpha}^T K(X, x_\star)$$

Where $K()$ is the radial basis function kernel.

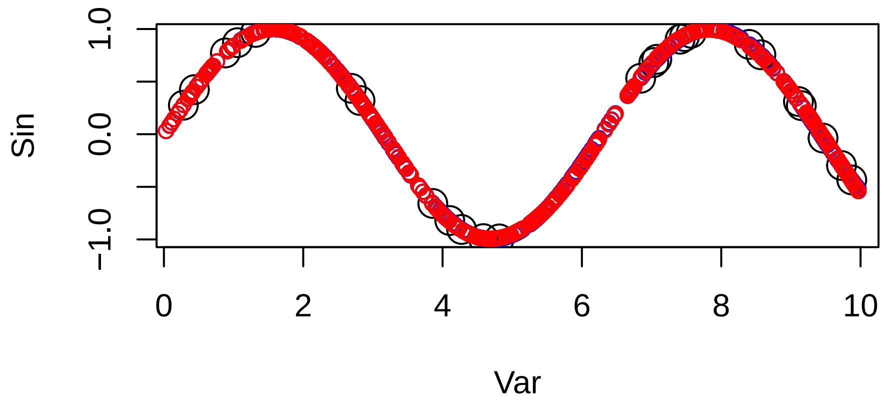
```
##  [1] -1.998999  1.560584  1.000278 -1.756815 -2.669577  1.291312 -1.068444
##  [8] -1.312493  1.000184 -2.208639
```

```
##            [,1]
##  [1,] -1.998999
##  [2,]  1.560584
##  [3,]  1.000278
##  [4,] -1.756815
##  [5,] -2.669577
##  [6,]  1.291312
##  [7,] -1.068444
##  [8,] -1.312493
##  [9,]  1.000184
## [10,] -2.208639
```

We get the same results as the predicted values from filter3

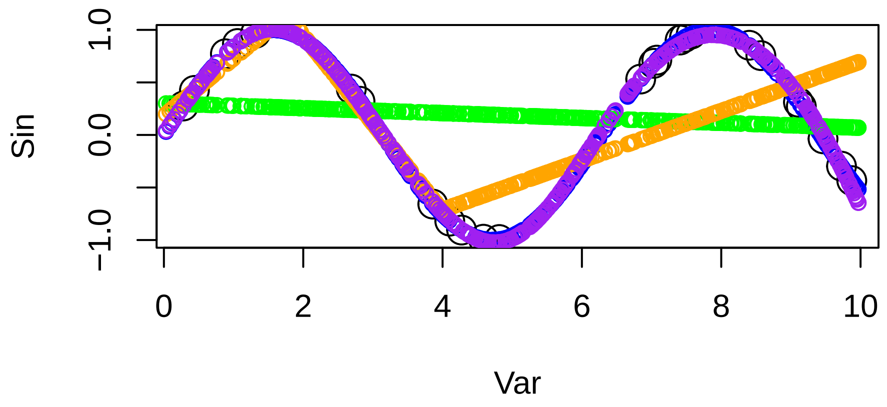
4 Assignment 3 - NN

4.1 1)



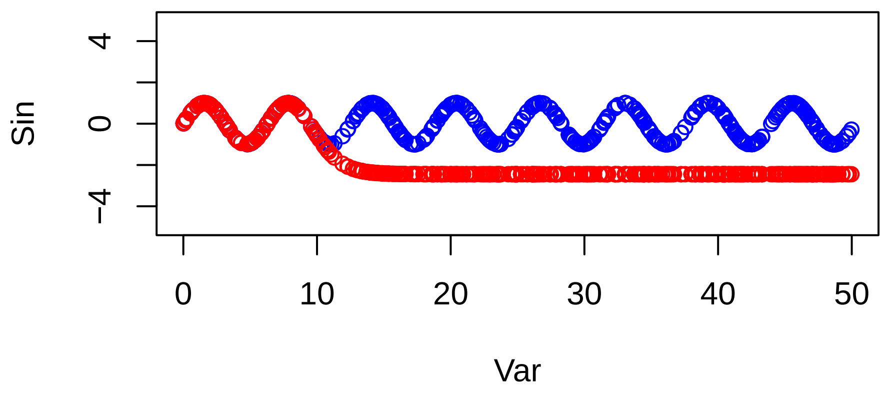
As we can see from the plot, we get very good results, the test data points and predicted sin value for the test data points using the neural network overlap each other suggesting that the model makes good prediction.

4.2 2)



Based on the 3 activation functions, we can see that the softplus activation function does the best and the linear activation function does the worst.

4.3 3)

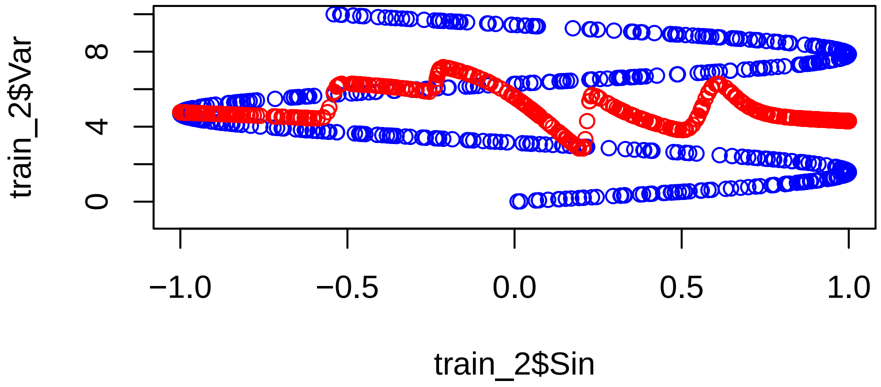


We can see from the plot that the model does well for input values between 0 and 10, and after that it doesnt get any prediction right. The neural network model that we trained was only trained for input values between 0 and 10. While the test data that we fed to this model has input values ranging from 0 and 50 as a result of this we can see that the model doesnt do well.

4.4 4)

We can see after input values of 10, the model predictions becomes a constant value. This is because for that model, the sigmoid activation function would always be close to 1 since the input values for it will be very large for the model. As a result, the in the final layer the prediction will gradually lead to a constant value.

4.5 5)



As we can see, the model doesn't do well to make good predictions. This is because when we take the inverse of the sine function, for a particular input value, we can get multiple possible output values and as a result of this, the model makes bad predictions.

5 Appendix

```
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning=FALSE, fig.width = 5, fig.height = 3, fig.a
set.seed(12345)
# packages
library(glmnet)
library(caret)
library(dplyr)
library(ggplot2)
library(geosphere)

set.seed(1234567890)
stations <- read.csv("stations.csv", fileEncoding = "latin1")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")
set.seed(1234567890)
h_distance <- 1e06
h_date <- 1e03
h_time <- 0.5e01
```

```r
a <- 56.023315 # The point to predict, Ramlösa brunnspark in Helsingborg
b <- 12.744271

date <- as.Date("1999-12-31") # The date to predict, millenium new year
st <- st[as.Date(st$date) <= date,] # Removing obs that are after chosen date


times <- c("04:00:00", "06:00:00", "08:00:00", "10:00:00", "12:00:00",
           "14:00:00", "16:00:00", "18:00:00", "20:00:00", "22:00:00", "00:00:00")

temp <- vector(length=length(times))
set.seed(1234567890)
# physical distance
k_phys <- c()
dp <- function(obs_lon, obs_lat, new_lon, new_lat, L){
  stopifnot(is.vector(obs_lon), is.vector(obs_lat)) # input check
  n <- length(obs_lon)
  d <- c() # distance vector

  # haversine distances & kernel
  for(i in 1:n){
    d[i] <- distHaversine(c(new_lon, new_lat), c(obs_lon[i], obs_lat[i[]]))
    k_phys[i] <<- exp(-d[i]^2 / (2*L^2))
    #cat(i, "\r")
  }

  # plotting
  plot(d, k_phys, type = "p")
}

dp(st$longitude, st$latitude, a, b, h_distance)
set.seed(1234567890)
# day distance
k_day <- c()
dd <- function(obs_date, new_date, L){
  stopifnot(is.vector(obs_date))
  n <- length(obs_date)
  d <- c()

  for(i in 1:n){
    d[i] <- abs(round(as.numeric(difftime(obs_date[i], new_date, units = "days"))))
    k_day[i] <<- exp(-d[i]^2 / (2*L^2))
    #cat(i, "\r")
  }

  plot(d, k_day, type = "p")

}
dd(st$date, date, 1e03)
```

```r
set.seed(1234567890)
# hours distance
k_hour <- matrix(nrow = length(st$time), ncol = length(times))
dh <- function(obs_hour, hours_vec, L){
  stopifnot(is.vector(obs_hour))
  n <- length(obs_hour)
  d <- matrix(NA, nrow = n, ncol = length(hours_vec))

  for(i in 1:n)
  for(j in 1:length(times)){
    t1 <- as.POSIXct(obs_hour[i], format = "%H:%M:%S")
    t2 <- as.POSIXct(hours_vec[j], format = "%H:%M:%S")

    d[i,j] <- abs(as.numeric(difftime(t1,t2, units = "hours")))
    k_hour[i,j] <<- exp(-d[i,j]^2 / (2*L^2))
    #cat(i, "\r")
  }

  plot(d, k_hour, type = "p")
}

dh(st$time, times, 0.5e01)
# summed kernel
kernel_sum <- function(hours_vec){
  hours_vec <- as.Date(hours_vec, format = "%H:%M:%S")
  for(i in 1:length(hours_vec)){
    temp[i] <<- sum((k_phys + k_day + k_hour[,i]) * st$air_temperature) / sum(k_phys + k_day + k_hour[,i
  }

  df <- data.frame("Time" = times,
                   "Temperature" = temp)

  plot(temp, type="o")
  return(df)
}
# product kernel
kernel_prod <- function(hours_vec){
  hours_vec <- as.Date(hours_vec, format = "%H:%M:%S")
  for(i in 1:length(hours_vec)){
    temp[i] <<- sum((k_phys * k_day * k_hour[,i]) * st$air_temperature) / sum(k_phys * k_day * k_hour[,i
  }

  df <- data.frame("Time" = times,
                   "Temperature" = temp)

  plot(temp, type="o")
  return(df)
}
kernel_sum(times)
```

```r
kernel_prod(times)
# Lab 3 block 1 of 732A99/TDDE01/732A68 Machine Learning
# Author: jose.m.pena@liu.se
# Made for teaching purposes

library(kernlab)
set.seed(1234567890)

data(spam)
foo <- sample(nrow(spam))
spam <- spam[foo,]
spam[,-58]<-scale(spam[,-58])
tr <- spam[1:3000, ]
va <- spam[3001:3800, ]
trva <- spam[1:3800, ]
te <- spam[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,scaled=FALSE)
  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,scaled=FALSE
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,scaled=FALSE
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,scaled=FAL
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,scaled=FAL
mailtype <- predict(filter3,te[,-58])
t <- table(mailtype,te[,58])
err3 <- (t[1,2]+t[2,1])/sum(t)
err3
```

```r
knitr::kable(err1, caption = "Generalization error of the returned filter(1)")

sv<-alphaindex(filter3)[[1]]
co<-coef(filter3)[[1]]
inte<- - b(filter3)
k<-NULL
rbf <- rbfdot(0.05) # the used kernel

for(i in 1:10){ # We produce predictions for just the first 10 points in the dataset.
  k2<-NULL
  for(j in 1:length(sv)){
    # go through every obs in the supportvector and doing the multiplication of the coefficient and kern
    k2 <- c(k2,co[j] * kernelMatrix(as.matrix(spam[sv[j],-58]),as.matrix(spam[i,-58]), kernel = rbf))
  }
  k <-c(k, sum(k2) +inte) # adding the intercept to the predicted k2
}
k
predict(filter3,spam[1:10,-58], type = "decision")


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

#Plotting train and test data
plot(train, cex=2)
points(test, col = "blue", cex=1)



model1<-neuralnet(Sin~Var,data = mydata,hidden = 10)


points(test[,1],predict(model1,test), col="red", cex=1)

plot(train, cex=2)
points(test, col = "blue", cex=1)
linear<-function(x){
  x
}


model2<-neuralnet(Sin~Var,data = mydata,hidden = 10,act.fct = linear)
points(test[,1],predict(model2,test), col="green", cex=1)
```

```
model3<-neuralnet(Sin~Var,data = mydata,hidden = 10,act.fct = relu)
points(test[,1],predict(model3,test), col="orange", cex=1)

# https://stackoverflow.com/questions/34532878/package-neuralnet-in-r-rectified-linear-unit-relu-activat

# We can see that directly using a function and feeding max(0,x) will not work as in the neuralnet() it
# activation function should be differentiable so go around this problem by using the "sigmoid" package.


softplus<-function(x){
  log(1+exp(x))
}
model4<-neuralnet(Sin~Var,data = mydata,hidden = 10,act.fct = softplus,threshold = 0.1,stepmax = 1e7)
points(test[,1],predict(model4,test), col="purple", cex=1)

# We had to increase the threshold and stepmax values since the neural network required more time to con

# https://stackoverflow.com/questions/16631065/r-neuralnet-does-not-converge-within-stepmax-for-time-ser

Var<- runif(500, 0, 50)
newdata <- data.frame(Var, Sin=sin(Var))

plot(newdata, col = "blue", cex=1, xlim= c(0,50), ylim= c(-5,5))
points(newdata[,1],predict(model1,newdata), col="red", cex=1)

Var<- runif(500, 0, 10)
train_2 <- data.frame(Var, Sin=sin(Var))

model5<-neuralnet(Var~Sin,data = train_2,hidden = 10,threshold = 0.1)

plot(train_2$Sin,train_2$Var, col = "blue", cex=1,ylim=c(-1,10))
points(train_2[,2],predict(model5,train_2), col="red", cex=1)
```