Machine Learning Computer Lab 1 Block2 (Group A7)

Qinyuan Qi(qinqi464) Satya Sai Naga Jaya Koushik Pilla (satpi345)
Daniele Bozzoli(danbo826)

2023-12-17

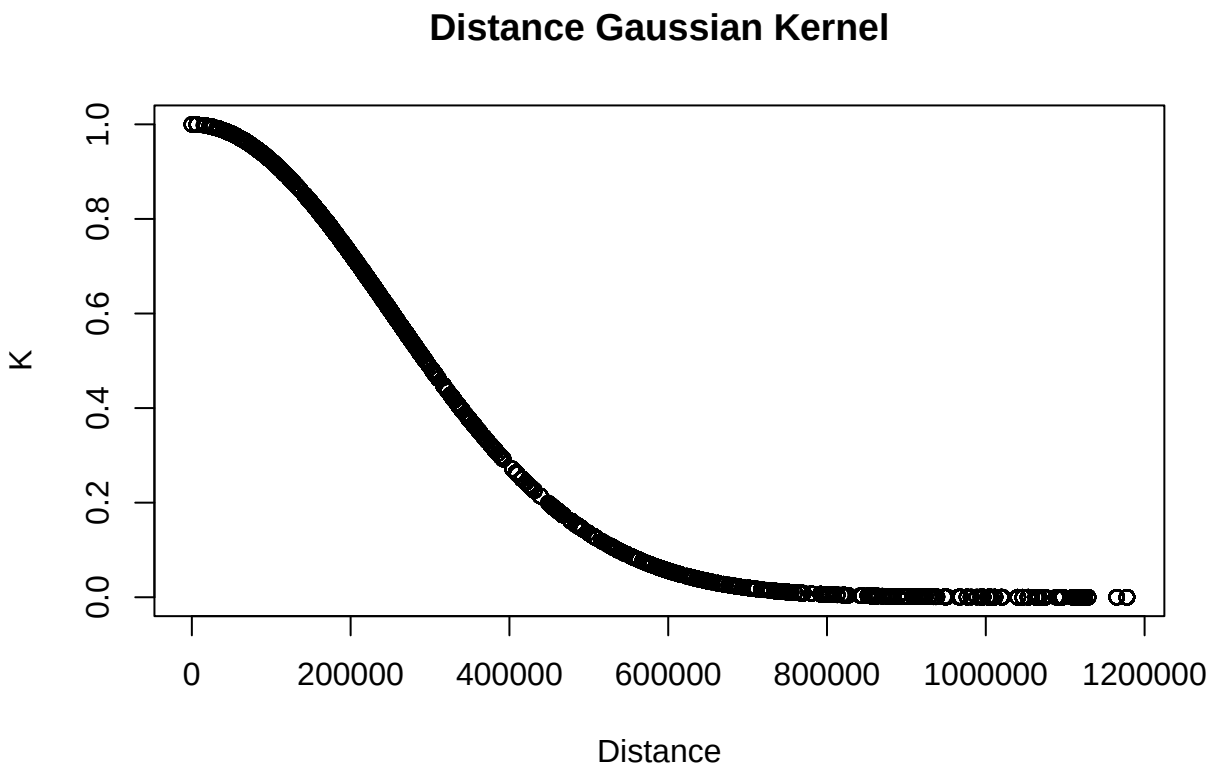
Assignment 1: KERNEL METHODS

Answer:

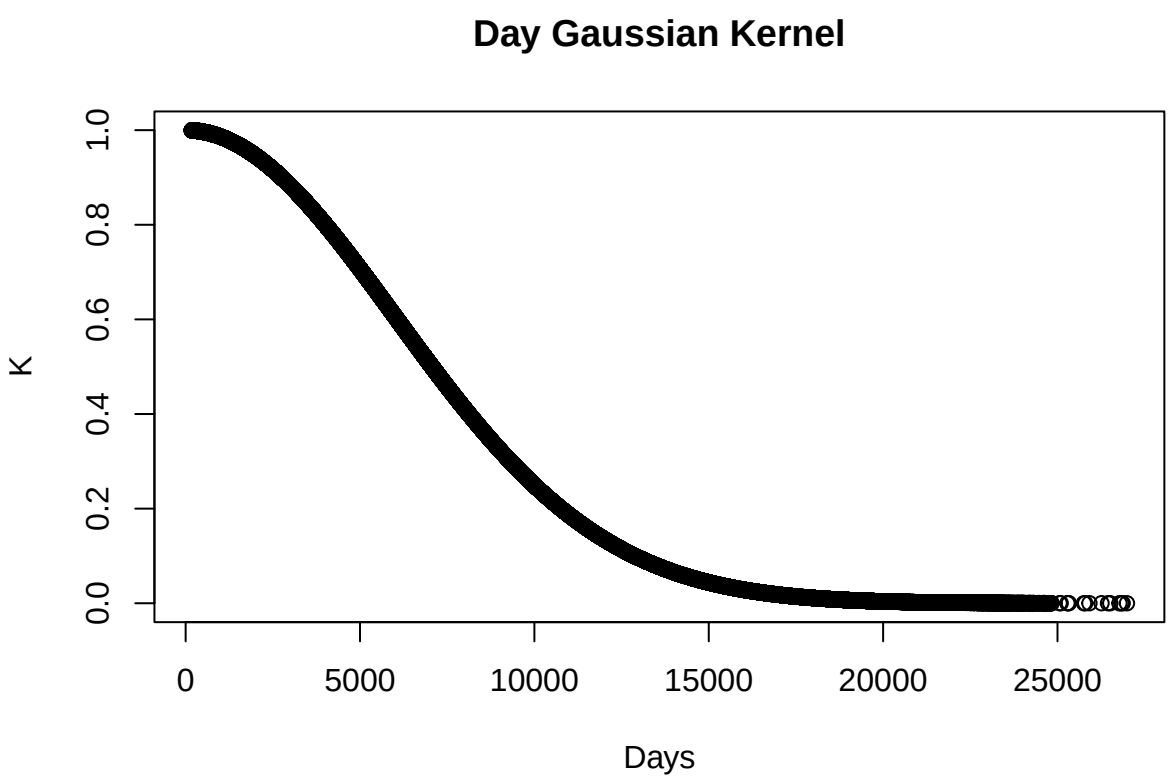
The kernel functions are implemented as follows.

```
########################## Kernel Code  ######################################
# Gaussian kernel of geo distance
geo_distance <- function(data, location_interest, h_dist) {

  location <- data.frame(longitude = data$longitude,
                         latitude  = data$latitude)

  distances <- distHaversine(location_interest,location)

  kernel_result <- exp(-(distances)^2 / (2 * h_dist^2))

  return(c(distances, kernel_result))
}

# Gaussian kernel of day distance
day_distance <- function(data, day_interest, h_day) {
  distances <- as.numeric(difftime(day_interest,data,
                           units = "days"))

  kernel_result <- exp(-(distances)^2 / (2 * h_day^2))
  return(c(distances, kernel_result))
}

# Gaussian kernel of hour distance
hour_distance <- function(data, hour_interest, h_hour) {
  distances <- as.numeric(difftime(strptime(hour_interest, "%H:%M:%S"),
                                strptime(data$time, "%H:%M:%S"),
                                units = "hours"))
  kernel_result <- exp(-(distances)^2 / (2 * h_day^2))
  return(c(distances, kernel_result))
}
```

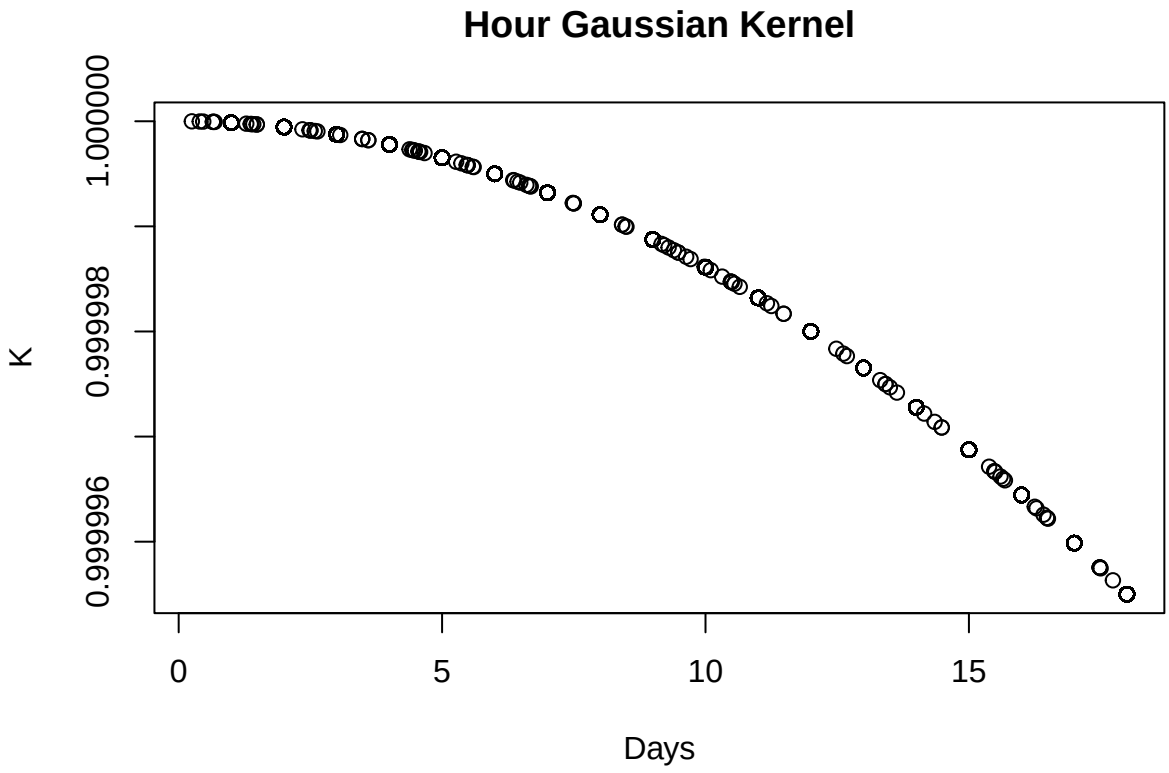
We set h_distance=250000,and set station to Linköping, then we plot the kernel function as follows



We set h_day=6000, and set date to 2016-12-26, we also filter the data after 2016-12-26, then we plot the kernel function as follows.



We set h_hour = 7, and time to 18:00, also filter the time after the specified time, then we plot the kernel function as follows.



Now we setting up the addition kernel of sum of 3 kernels, and plot the kernel function as follows.

Assignment 2: SUPPORT VECTOR MACHINES

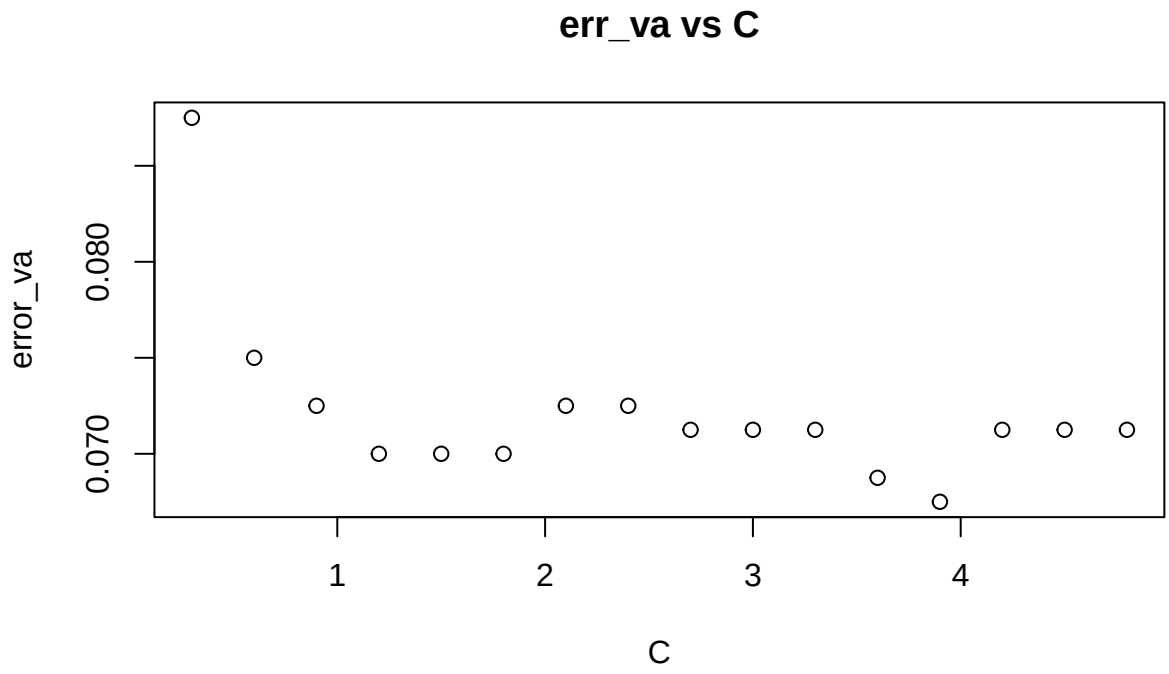
Answer:

According to the code, we know that the spam data is splitted into training,validation,training+validation, and test data respectively.

According to the code output, we know that the minimal error is 0.0675 when C = 0.3.

```
## minimal error = 0.0675  when C = 3.9
```

Also we draw the plot of error_va versus C as follows.



2.1. Which filter do we return to the user ? filter0, filter1, filter2 or filter3? Why?

Then we calc the error rate of 4 filters, accoring to this, we know the accuracy rates are as follows.

Filter	Accuracy	Accept / Reject
filter0	93.25 %	Accept

Filter0 follow the best practice and get the highest accuracy rate. It train on training data,it predict on validation data

Filter	Accuracy	Accept / Reject
filter1	91.51061 %	Reject

Filter1 follow the best practice but its accuracy rate is lower than filter 0. It train on training data,it predict on test data

Filter	Accuracy	Accept / Reject
filter2	91.7603 %	Reject

Filter2 does not follow the best practice but still practiciable, It train on training+validation data, predict on test data, also its accuracy rate is lower than filter 0.

Filter	Accuracy	Accept / Reject
filter3	97.87765 %	Reject

Filter3 training on all data,and using the seen data to predict which is not accepetable.

Based on the analysis above, we should return filter0 to the user.

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

Generalization error is the expectation of the the predicted values of an independent test set on our model.

Based on the test data, we got the following result.

The reason why filter0 and filter1 are same is because these 2 filters share the same parameters and are same.

```
## generalization error of filter0 is 8.489388 %
```

```
## generalization error of filter1 is 8.489388 %
```

```
## generalization error of filter2 is 8.2397 %
```

```
## generalization error of filter3 is 2.122347 %
```

2.3 Implementation of SVM predictions.

The missing code implemented as follows.

```
sv <- alphaindex(filter3)[[1]]
co<-coef(filter3)[[1]]
inte<- - b(filter3)
# RBF kernel with sigma = 0.05
rbf <- rbfdot(sigma = 0.05)
k<-NULL
# We produce predictions for just the first 10 points in the dataset.
for(i in 1:10){
  k2<-NULL
  for(j in 1:length(sv)){
    k2 <- k2 + co[j] * rbf(as.numeric(spam[sv[j], -58]),
                           as.numeric(spam[i, -58]))
  }
  k<-c(k,  k2 + inte)
}
k
```

```
## numeric(0)
```

```
pred_filter3 <- predict(filter3,spam[1:10,-58], type = "decision")
pred_filter3
```

```
##            [,1]
##  [1,] -1.998999
##  [2,]  1.560584
##  [3,]  1.000278
##  [4,] -1.756815
##  [5,] -2.669577
##  [6,]  1.291312
##  [7,] -1.068444
```

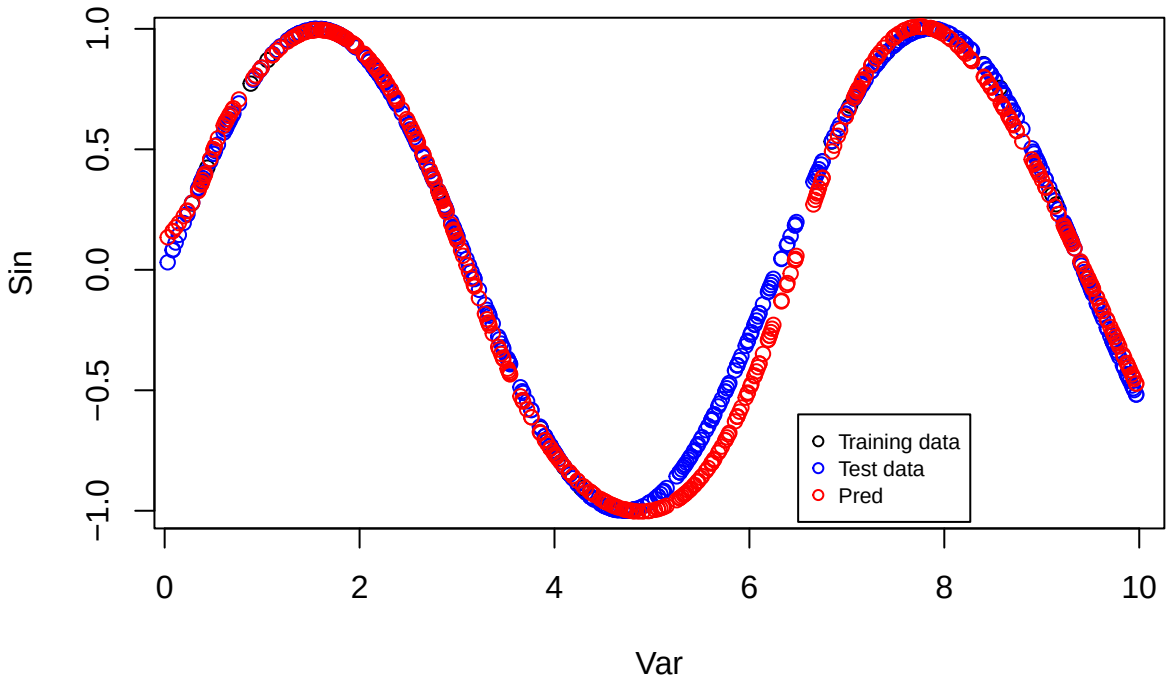
```
##  [8,] -1.312493
##  [9,]  1.000184
## [10,] -2.208639
```

Assignment 3: NEURAL NETWORKS

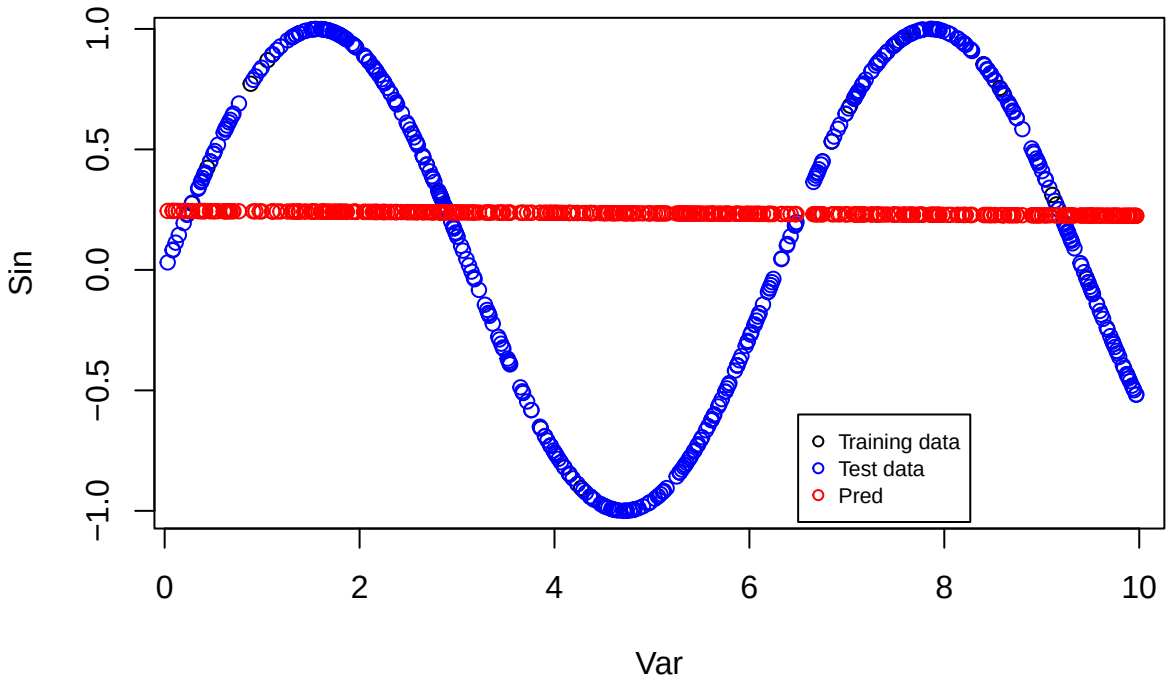
Answer:

3.1

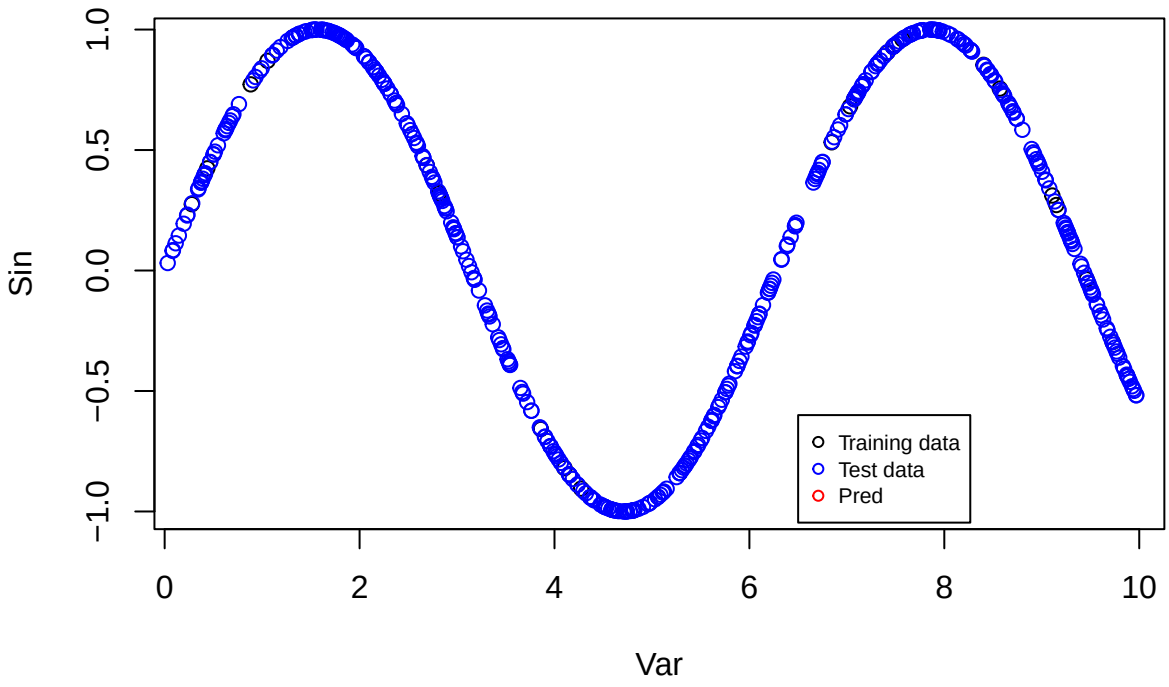
We observe in the plot that the predictions from the neural net seem to be generally good predictions for the sine function.

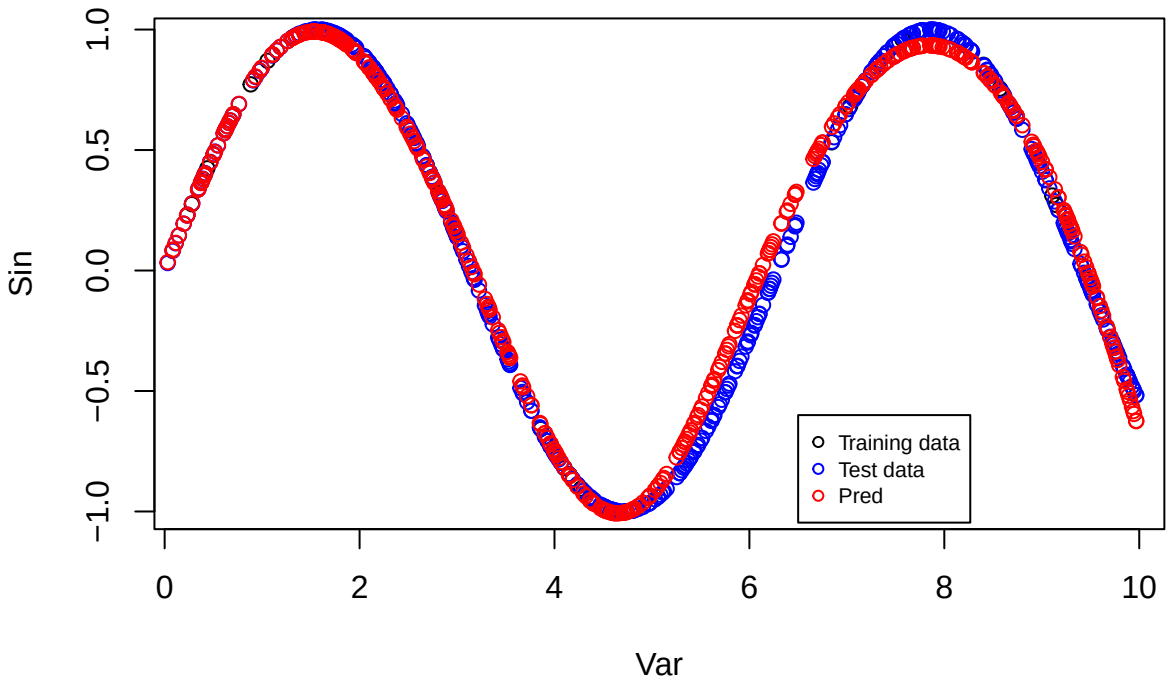


3.2

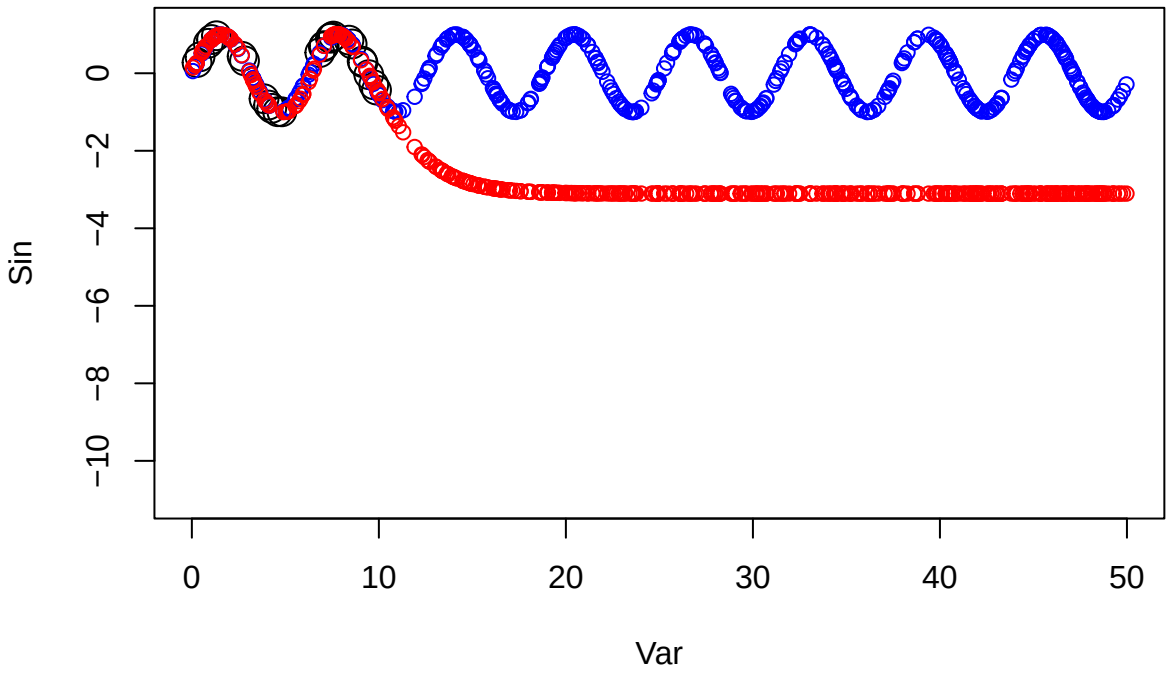



```
## Error in Deriv_(a, x[ix], env, use.D, dsym, scache, drule. = drule.): Could not retrieve body of 'ma
## Error in h(simpleError(msg, call)): error in evaluating the argument 'object' in selecting a method
```


Sin
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10
Var
Training data
Test data
Pred


Sin
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10
Var
Training data
Test data
Pred

3.3



3.4

```
##       Weights        Bias
## 1   2.0477256  -6.3100974
## 2   0.9339598  -1.6486819
## 3  15.9608771  -7.2656860
## 4  -1.4741810   1.8712084
## 5  -1.7007479  11.2056822
## 6  -0.8475234   0.7600561
## 7  -0.5578519   5.0994514
## 8  -1.5924443   1.3050353
## 9   1.9340528 -12.7759163
## 10 -0.8704172  -1.1242483
```

3.5

Appendix: All code for this report

```
############################  Init code For Assignment 1 ########################
rm(list = ls())
knitr::opts_chunk$set(echo = TRUE)
library(geosphere)
set.seed(1234567890)
############################ init data ######################################
stations <- read.csv("stations.csv", fileEncoding = "latin1")
temps <- read.csv("temps50k.csv")
st <- merge(stations, temps, by = "station_number")

# split the data to train and test (70/30)
n <- dim(st)[1]
id <- sample(1:n, floor(n * 0.7))
train <- st[id, ]
test <- st[-id, ]

######################### Kernel Code  ####################################
# Gaussian kernel of geo distance
geo_distance <- function(data, location_interest, h_dist) {

  location <- data.frame(longitude = data$longitude,
                         latitude  = data$latitude)

  distances <- distHaversine(location_interest,location)

  kernel_result <- exp(-(distances)^2 / (2 * h_dist^2))

  return(c(distances, kernel_result))
}

# Gaussian kernel of day distance
day_distance <- function(data, day_interest, h_day) {
  distances <- as.numeric(difftime(day_interest,data,
                           units = "days"))

  kernel_result <- exp(-(distances)^2 / (2 * h_day^2))
  return(c(distances, kernel_result))
}

# Gaussian kernel of hour distance
hour_distance <- function(data, hour_interest, h_hour) {
  distances <- as.numeric(difftime(strptime(hour_interest, "%H:%M:%S"),
                                   strptime(data$time, "%H:%M:%S"),
                                   units = "hours"))
  kernel_result <- exp(-(distances)^2 / (2 * h_day^2))
  return(c(distances, kernel_result))
}
################################# 1.1 Distance Kernel ############################
# Station Name to Linköping
station_row <- which(stations$station_name == "Linköping")
station <- stations$station_name[station_row]
```

```r
# These three values are up to the students
h_distance <- 250000

# set coordinates to predict
a <- stations$longitude[station_row]
b <- stations$latitude[station_row]

N <- nrow(train)
i <- 1:N
k_distance <- sapply(i, function(i){geo_distance(train[i, ], c(a, b), h_distance)})
plot(k_distance[1,], k_distance[2,], main = "Distance Gaussian Kernel", xlab = "Distance", ylab = "K")
################################## 1.2 Day Kernel ################################
h_day <- 6000
date_interest <- as.POSIXlt("2016-12-26")
train_filtered <- train[train$date < date_interest, ]
k_day <- sapply(i, function(i) day_distance(as.POSIXlt(train_filtered$date[i]), date_interest, h_day))
plot(k_day[1,], k_day[2,], main = "Day Gaussian Kernel", xlab = "Days", ylab = "K")

################################## 1.3 Hour Kernel ################################
times_interest <- "18:00:00"

h_hour <- 7
train_filtered <- train[train$time < times_interest, ]
k_hour <- sapply(i, function(i) hour_distance(train_filtered[i,], times_interest, h_hour))
plot(k_hour[1,], k_hour[2,], main = "Hour Gaussian Kernel", xlab = "Days", ylab = "K")
################################## 1.4 Sum Kernel ################################
# we filter 2 times
train_filtered <- train[train$time < times_interest, ]
train_filtered <- train_filtered[train_filtered$date < date_interest, ]

################################## 1.5 Multi Kernel ################################


############################  Init code For Assignment 2 ########################
rm(list = ls())
knitr::opts_chunk$set(echo = TRUE)
library(kernlab)
set.seed(1234567890)
############################  Load Data and split data sets ####################
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, ]
############################  Get error va and corresponding C ##################
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))
}

cat("minimal error =", min(err_va), " when C =", which.min(err_va) * by,"\n")
##########################  Plot err_va vs C ####################
c_plot <- seq(by, 5, by)
plot(x = c_plot, y = err_va, type = "p",xlab = "C", ylab = "error_va", main = "err_va vs C")

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)

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)

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


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

#cat("accuracy of filter0 is", (1-err0)*100,"%\n")
#cat("accuracy of filter1 is", (1-err1)*100,"%\n")
#cat("accuracy of filter2 is", (1-err2)*100,"%\n")
#cat("accuracy of filter3 is", (1-err3)*100,"%\n")
filter0_pred <- predict(filter0, te, type = "response")
filter0_gerr <- mean(filter0_pred != te[, 58])

filter1_pred <- predict(filter1, te, type = "response")
filter1_gerr <- mean(filter1_pred != te[, 58])

filter2_pred <- predict(filter2, te, type = "response")
filter2_gerr <- mean(filter2_pred != te[, 58])

filter3_pred <- predict(filter3, te, type = "response")
filter3_gerr <- mean(filter3_pred != te[, 58])

cat("generalization error of filter0 is", filter0_gerr*100,"%\n")
cat("generalization error of filter1 is", filter1_gerr*100,"%\n")
cat("generalization error of filter2 is", filter2_gerr*100,"%\n")
cat("generalization error of filter3 is", filter3_gerr*100,"%\n")
sv <- alphaindex(filter3)[[1]]
co<-coef(filter3)[[1]]
inte<- - b(filter3)
```

```
# RBF kernel with sigma = 0.05
rbf <- rbfdot(sigma = 0.05)
k<-NULL
# We produce predictions for just the first 10 points in the dataset.
for(i in 1:10){
  k2<-NULL
  for(j in 1:length(sv)){
    k2 <- k2 + co[j] * rbf(as.numeric(spam[sv[j], -58]),
                           as.numeric(spam[i, -58]))
  }
  k<-c(k,  k2 + inte)
}
k
pred_filter3 <- predict(filter3,spam[1:10,-58], type = "decision")
pred_filter3
############################  Init code For Assignment 3 ########################
rm(list = ls())
knitr::opts_chunk$set(echo = TRUE)
library(neuralnet)
set.seed(1234567890)
Var <- runif(500, 0, 10)
mydata <- data.frame(Var, Sin=sin(Var))
tr <- mydata[1:25,] # Training
te <- mydata[26:500,] # Test

# Random initialization of the weights in the interval [-1, 1]
winit <- runif(31, -1, 1)
nn <- neuralnet(data = tr, formula = Sin ~ Var, hidden = c(10))

# Plot of the training data (black), test data (blue), and predictions (red)
plot(tr,  col = "black", cex=1)
points(te, col = "blue", cex=1)
points(te[,1], predict(nn,te), col="red", cex=1)
legend(x = 6.5, y = -0.6, legend = c("Training data", "Test data",
"Pred"), col = c("black", "blue", "red"), cex = 0.7,pch = 1)
#Linear
h1 <- function(x) {
  x
}

nn1 <- neuralnet(data = tr, formula = Sin ~ Var, hidden = c(10), act.fct = h1)

# Plot of the training data (black), test data (blue), and predictions (red)
plot(tr,  col = "black", cex=1)
points(te, col = "blue", cex=1)
points(te[,1], predict(nn1,te), col="red", cex=1)
legend(x = 6.5, y = -0.6, legend = c("Training data", "Test data",
"Pred"), col = c("black", "blue", "red"), cex = 0.7,pch = 1)


#ReLU
h2 <- function(x) {
  max(0,x)
```

```
}


nn2 <- neuralnet(formula = Sin ~ Var, data = tr, hidden = 10,
startweights = winit, act.fct = h2)

# Plot of the training data (black), test data (blue), and predictions (red)
plot(tr,  col = "black", cex=1)
points(te, col = "blue", cex=1)
points(te[,1], predict(nn2,te), col="red", cex=1)
legend(x = 6.5, y = -0.6, legend = c("Training data", "Test data",
"Pred"), col = c("black", "blue", "red"), cex = 0.7,pch = 1)


#Softplus
h3 <- function(x) {
  log(1+exp(x))
}

nn3 <- neuralnet(formula = Sin ~ Var, data = tr, hidden = 10,
startweights = winit, act.fct = h3)

# Plot of the training data (black), test data (blue), and predictions (red)
plot(tr,  col = "black", cex=1)
points(te, col = "blue", cex=1)
points(te[,1], predict(nn3,te), col="red", cex=1)
legend(x = 6.5, y = -0.6, legend = c("Training data", "Test data",
"Pred"), col = c("black", "blue", "red"), cex = 0.7,pch = 1)

# Sample 500 points
Var <- runif(500, min = 0, max = 50)
df <- data.frame(Var, Sin = sin(Var))
plot(tr, cex = 2, xlim = c(0, 50), ylim = c(-11, 1.2))
points(df, col = "blue", cex = 1)
points(df[, 1], predict(nn, df), col = "red", cex = 1)
data.frame(Weights = nn$weights[[1]][[1]][2, ], Bias = nn$weights[[1]][[1]][1,])
```