lab3

Statement of Contribution

Alexander Bois is responsible for assignment 1, Daniel Bissessar for assignment 2 and Filip Brunander for assignment 3. Moreover, several programming sessions was held where we discussed problems and helped each other. However, the help and discussions where limited where the one responsible for the assignment also wrote all the code and answered all questions related to that assignment. Sessions were held to explain the different assignments to each group member.

Lab 3 assignment 1

Alexander Bois

Assignment 1

First we set the parameters for the different h values, date, location and specifying the times to predict for.

```
h_distance <- 100000
h_date <- 15
h_time <- 4
a <- 58.4108 # The point to predict (up to the students) Linköping
b <- 15.6213
my_date <- "2013-01-04" # The date to predict (up to the students)
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", "24:00:00")
temp <- vector(length=length(times))
```

We then create functions for filtering the data to not make predictions based on data that is posterior to our date.

```
filter_data <- function(){
  filtered_data <- filter(st, date < my_date)
  return(filtered_data)
}
```

Then we create the three different kernels that will be used to construct our final kernel with three functions. The kernels are based on the distance from our point to all points and are Gaussian.

```
dist_kernel <- function(fd){
  res1 <- vector(length=dim(fd)[1])
```

```
  diff <- vector(length=dim(fd)[1])
  for (i in 1:dim(fd)[1]){
    diff[i] <- distHaversine(c(b, a), c(fd$longitude[i], fd$latitude[i]))
    res1[i] <- exp(-1*(diff[i]/h_distance)^2)
  }
  return(cbind(res1, diff))
}

day_kernel <- function(fd){
  res1 <- vector(length=dim(fd)[1])
  diff <- vector(length=dim(fd)[1])
  split <- strsplit(my_date, "\\-")
  day_x <- as.numeric(split[[1]][3])
  month_X <- as.numeric(split[[1]][2])
  days_x <- ((month_X-1)*30) + day_x
  for (i in 1:dim(fd)[1]){
    split <- strsplit(fd$date[i], "\\-")
    day_fd <- as.numeric(split[[1]][3])
    month_fd <- as.numeric(split[[1]][2])
    days_fd <- ((month_fd-1)*30) + day_fd
    diff[i] <- abs(days_x - days_fd)
    res1[i] <- exp(-1*(diff[i]/h_date)^2)
  }
  return(cbind(res1, diff))
}

hour_kernel <- function(fd, time){
  res1 <- vector(length=dim(fd)[1])
  diff <- vector(length=dim(fd)[1])
  split <- strsplit(time, "\\:")
  hour_x <- as.numeric(split[[1]][1])
  for (i in 1:dim(fd)[1]){
    time_i <- strsplit(fd$time[i], "\\:")
    hour <- as.numeric(time_i[[1]][1])
    diff[i] <- abs(hour_x - hour)
    res1[i] <- exp(-1*(diff[i]/h_time)^2)
  }
  return(cbind(res1, diff))
}
```

We then create two functions that combines the three kernels above according to the instructions with the first adding the kernels to make the predictions and the second multiplies the kernels to make predictions.

```
temperatures1 <- function(fd){
  dist_k <- dist_kernel(fd)
  day_k <- day_kernel(fd)
  for (i in 1:length(times)){
    hour_k <- hour_kernel(fd, times[i])
    kernels <- dist_k[,1] + day_k[,1] + hour_k[,1]
```

```
    temp[i] <- sum(kernels*fd$air_temperature)/sum(kernels)
  }
  return(temp)
}

temperatures2 <- function(fd){
  dist_k <- dist_kernel(fd)
  day_k <- day_kernel(fd)
  for (i in 1:length(times)){
    hour_k <- hour_kernel(fd, times[i])
    kernels <- dist_k[,1]*day_k[,1]*hour_k[,1]
    temp[i] <- sum(kernels*fd$air_temperature)/sum(kernels)
  }
  return(temp)
}
```

To determine which h values seems reasonable we can plot the kernel values vs distance to see how much influence (kernel value between 0 to 1) of the decision points have based on their distance. To choose the right h values is up to our ability to reason about what influence points at different distances should have. The graphs are limited on the x axis to make them more readable.

Assignment 2

```
##Lab 3

# Lab 3 block 1 of 732A99/TDDE01 Machine Learning
# Author: jose.m.pena@liu.se
# Made for teaching purposes
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=whic
h.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=whic
h.min(err_va)*by)
mailtype <- predict(filter3,te[,-58])
t <- table(mailtype,te[,58])
err3 <- (t[1,2]+t[2,1])/sum(t)
err3
```

Questions

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

The filter we should return to the user is filter2 due to it training on the largest set of data while still having holdout data to make an estimation of the generatization error

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

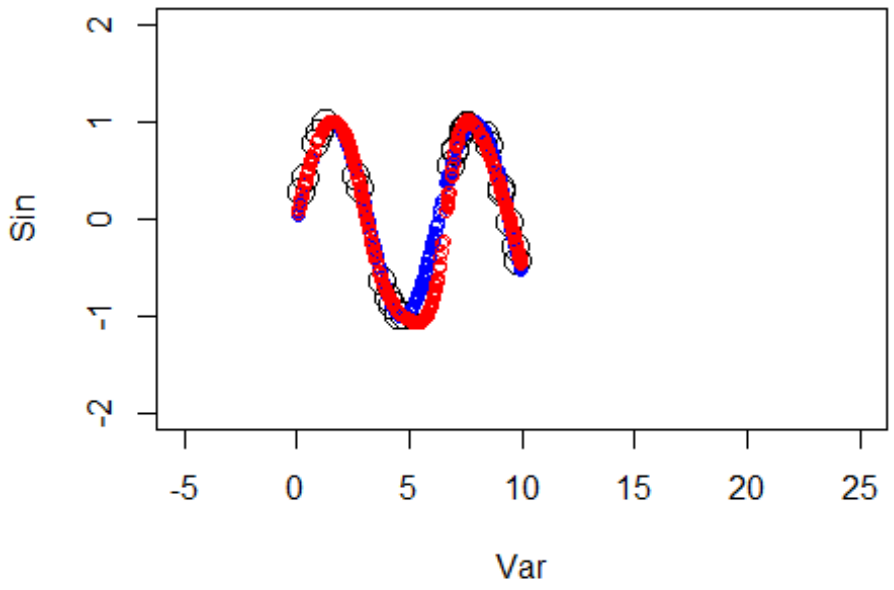
err2 due to it being a calculation of the missclassification rate for filter2 on data the model has previously not trained on.

Assignment 3

Task 1

```
set.seed(1234567890)
Var <- runif(500, 0, 10)
mydata <- data.frame(Var, Sin=sin(Var))
tr <- mydata[1:25,]
te <- mydata[26:500,]

# Random initialization of the weights in the interval [-1, 1]
set.seed(1234567890)
winit <- runif(7, -1, 1)
nn <- neuralnet(formula=Sin~Var, data=tr, hidden = 6, startweights = winit)
# Plot of the training data (black), test data (blue), and predictions (red)
plot(tr, cex=2, xlim = c(-5, 25), ylim = c(-2,2))
points(te, col = "blue", cex=1)
points(te[,1],predict(nn,te), col="red", cex=1)
```



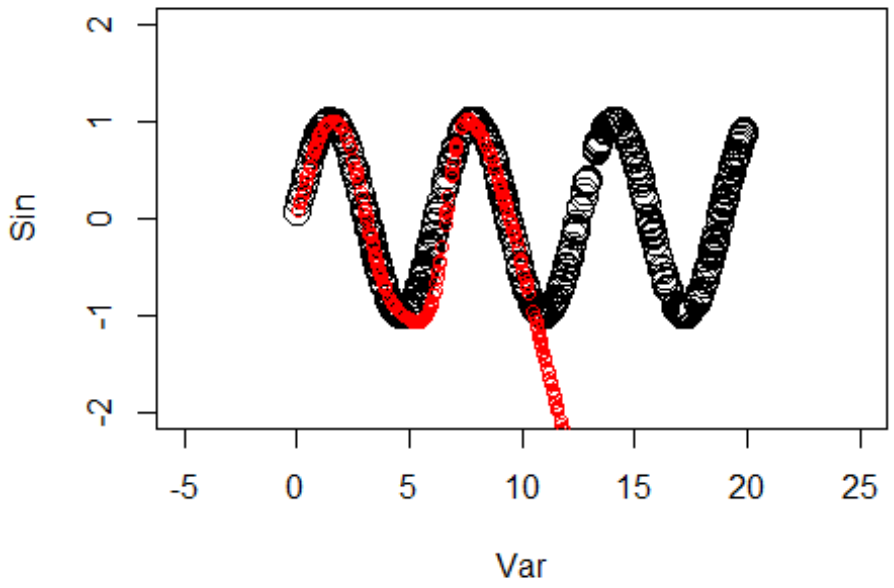

```
plot(nn)
```

The number of hidden neurons in the single hidden layer is decided to be 6 based on some trials with different number of hidden neurons. What we looked for was a number that was as low as possible but still provided a neural net that gave us good results. Some of the lower number also provided reasonable results, however, the error of the neural net with 6 hidden neurons was noticeable lower and was therefore our choice.

The results can be seen in the graph. The black larger circles are the training data, the blue the test data and the red are the predicted values. The predictions stray a bit from the actual test data between Var=5 and Var=7 where the prediction of Sin values are a bit too low. However, overall the prediction follows the actual test values good, and therefore is our neural network providing good results.

Task 2

```
set.seed(1234567890)
Var2 <- runif(500, 0, 20)
mydata2 <- data.frame(Var=Var2, Sin=sin(Var2))
plot(mydata2, cex=2, xlim = c(-5, 25), ylim = c(-2,2))
#points(mydata2, col = "blue", cex=1)
points(mydata2[,1],predict(nn,mydata2), col="red", cex=1)
```

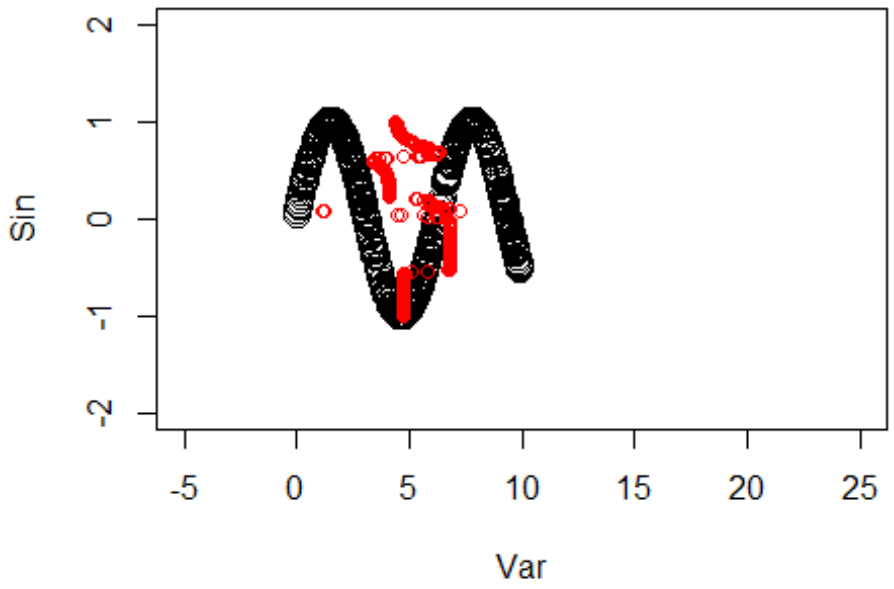


In this figure we have the black circles as the training data and the red circles the predicted values based on the same data. Here the predictions are the same as in task 1 until we reach Var=10 where the neural network no longer are providing reasonable results. The training data is based on random Var numbers between 0 and 10 while the test data in this case is based on random Var values between 0 and 20. This results in a neural network that has not trained on Var values larger than 10 and therefore explains the bad results when Var>10.

Task 3

```
set.seed(1234567890)
winit <- runif(7, -1, 1)
```

```
Var3 <- runif(500, 0, 10)
mydata3 <- data.frame(Var=Var3, Sin=sin(Var3))
nn3 <- neuralnet(formula=Var~Sin,data=mydata3,hidden = 6, startweights = wini
t)
plot(mydata3, cex=2, xlim = c(-5, 25), ylim = c(-2,2))
#points(mydata3, col = "blue", cex=1)
points(predict(nn3,mydata3), mydata3$Sin, col="red", cex=1)
```



Here we have the black circles which is the training data and the red circles that are the predictions based on the same training data. The predictions are nowhere close to the actual training values resulting in much larger error compared to the good predictions in task 1.

```
plot(nn)
plot(nn3)
```

Appendix: code for all labs:

Lab 1:

set.seed(1234567890)

library(geosphere)

stations <- read.csv("stations.csv")

```
temps <- read.csv("temps50k.csv")

st <- merge(stations,temps,by="station_number")

h_distance <- 100000

h_date <- 15

h_time <- 4

a <- 58.4108 # The point to predict (up to the students) Linköping: Breddgraden: 58.41080700000001 Längdgrad: 15.621372699999938

b <- 15.6213

my_date <- "2013-11-04" # The date to predict (up to the students)

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", "24:00:00")

temp <- vector(length=length(times))

### Students' code here ###

library(dplyr)


filter_data <- function(){
  filtered_data <- filter(st, date < my_date)
  return(filtered_data)
}


dist_kernel <- function(fd){
  res1 <- vector(length=dim(fd)[1])
  diff <- vector(length=dim(fd)[1])
  for (i in 1:dim(fd)[1]){
    diff[i] <- distHaversine(c(b, a), c(fd$longitude[i], fd$latitude[i]))
    res1[i] <- exp(-1*(diff[i]/h_distance)^2)
  }
  return(cbind(res1, diff))
}
```

```
day_kernel <- function(fd){
  res1 <- vector(length=dim(fd)[1])
  diff <- vector(length=dim(fd)[1])
  split <- strsplit(my_date, "\\-")
  day_x <- as.numeric(split[[1]][3])
  month_X <- as.numeric(split[[1]][2])
  days_x <- ((month_X-1)*30) + day_x
  for (i in 1:dim(fd)[1]){
    split <- strsplit(fd$date[i], "\\-")
    day_fd <- as.numeric(split[[1]][3])
    month_fd <- as.numeric(split[[1]][2])
    days_fd <- ((month_fd-1)*30) + day_fd
    diff[i] <- abs(days_x - days_fd)
    res1[i] <- exp(-1*(diff[i]/h_date)^2)
  }
  return(cbind(res1, diff))
}

hour_kernel <- function(fd, time){
  res1 <- vector(length=dim(fd)[1])
  diff <- vector(length=dim(fd)[1])
  split <- strsplit(time, "\\:")
  hour_x <- as.numeric(split[[1]][1])
  for (i in 1:dim(fd)[1]){
    time_i <- strsplit(fd$time[i], "\\:")
    hour <- as.numeric(time_i[[1]][1])
    diff[i] <- abs(hour_x - hour)
```

```
   res1[i] <- exp(-1*(diff[i]/h_time)^2)
  }
  return(cbind(res1, diff))
}

temperatures1 <- function(fd){
  dist_k <- dist_kernel(fd)
  day_k <- day_kernel(fd)
  for (i in 1:length(times)){
    hour_k <- hour_kernel(fd, times[i])
    kernels <- dist_k[,1] + day_k[,1] + hour_k[,1]
    temp[i] <- sum(kernels*fd$air_temperature)/sum(kernels)
  }
  return(temp)
}

temperatures2 <- function(fd){
  dist_k <- dist_kernel(fd)
  day_k <- day_kernel(fd)
  for (i in 1:length(times)){
    hour_k <- hour_kernel(fd, times[i])
    kernels <- dist_k[,1]*day_k[,1]*hour_k[,1]
    temp[i] <- sum(kernels*fd$air_temperature)/sum(kernels)
  }
  return(temp)
}
```

```
test_data <- filter_data()
temp_list_test <- temperatures1(test_data)
temp_list_test2 <- temperatures2(test_data)

plot(temp_list_test, type="o", xlab = "Time", ylab = "Temperature", xaxt="n")
plot(temp_list_test2, type="o", xlab = "Time", ylab = "Temperature", xaxt="n")
axis(1, at=1:length(times), labels = times)

#plots for kernel value vs distance.
dis_gau_mat <- dist_kernel(test_data)
plot(dis_gau_mat[,2], dis_gau_mat[,1], xlim = c(0,500000))

day_gau_mat <- day_kernel(test_data)
plot(day_gau_mat[,2], day_gau_mat[,1], xlim=c(0, 100))

hou_gau_mat <- hour_kernel(test_data, times[1])
plot(hou_gau_mat[,2], hou_gau_mat[,1])
```

Lab 2:

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

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


# Questions


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

# The filter we should return to the user i filter2 due to it training on the largest set of data while still having holdout data to make an estimation of the generatization error

# 2. What is the estimate of the generalization error of the filter returned ? err0, err1, err2 or err3 ? Why ?

# err2 due to it being a calculation of the missclassification rate for filter2 on data the model has previously not trained on.
```

Lab 3:

```
library(neuralnet)


#Task 1
```

```
set.seed(1234567890)
Var <- runif(500, 0, 10)
mydata <- data.frame(Var, Sin=sin(Var))
tr <- mydata[1:25,]
te <- mydata[26:500,]

# Random initialization of the weights in the interval [-1, 1]
set.seed(1234567890)
winit <- runif(2, -1, 1)
nn <- neuralnet(formula=Sin~Var,data=tr,hidden = 6, startweights = winit)
# Plot of the training data (black), test data (blue), and predictions (red)
plot(tr, cex=2, xlim = c(-5, 25), ylim = c(-2,2))
points(te, col = "blue", cex=1)
points(te[,1],predict(nn,te), col="red", cex=1)

#Task 2
set.seed(1234567890)
Var2 <- runif(500, 0, 20)
mydata2 <- data.frame(Var=Var2, Sin=sin(Var2))
plot(mydata2, cex=2, xlim = c(-5, 25), ylim = c(-2,2))
#points(mydata2, col = "blue", cex=1)
points(mydata2[,1],predict(nn,mydata2), col="red", cex=1)

#Task 3
set.seed(1234567890)
winit <- runif(7, -1, 1)
Var3 <- runif(500, 0, 10)
mydata3 <- data.frame(Var=Var3, Sin=sin(Var3))
```

```
nn3 <- neuralnet(formula=Var~Sin,data=mydata3,hidden = 6, startweights = winit)
plot(mydata3, cex=2, xlim = c(-5, 25), ylim = c(-2,2))
#points(mydata3, col = "blue", cex=1)
points(predict(nn3,mydata3), mydata3$Sin, col="red", cex=1)


plot(nn3)
```