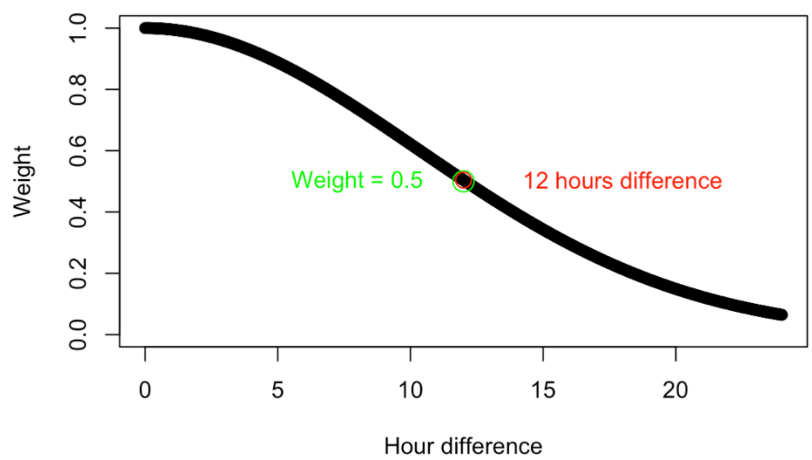
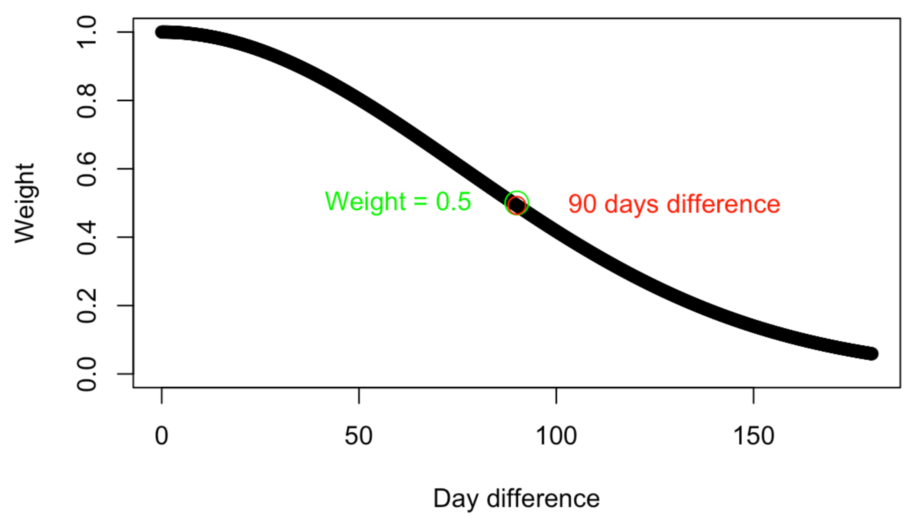
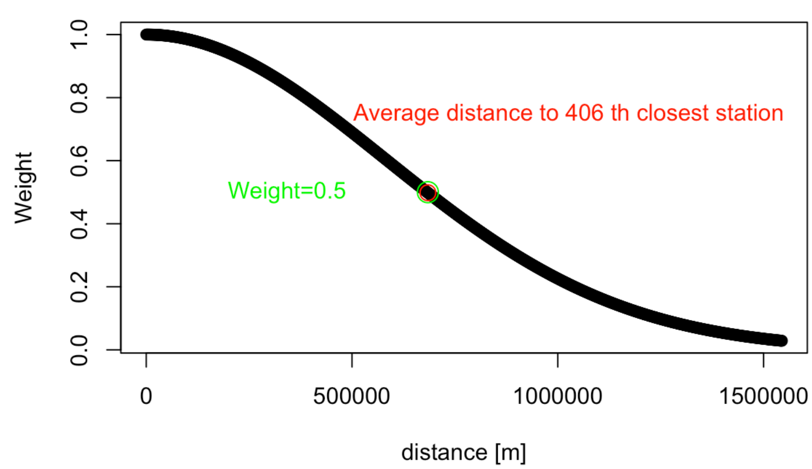
Assignment 1: Kernel Methods



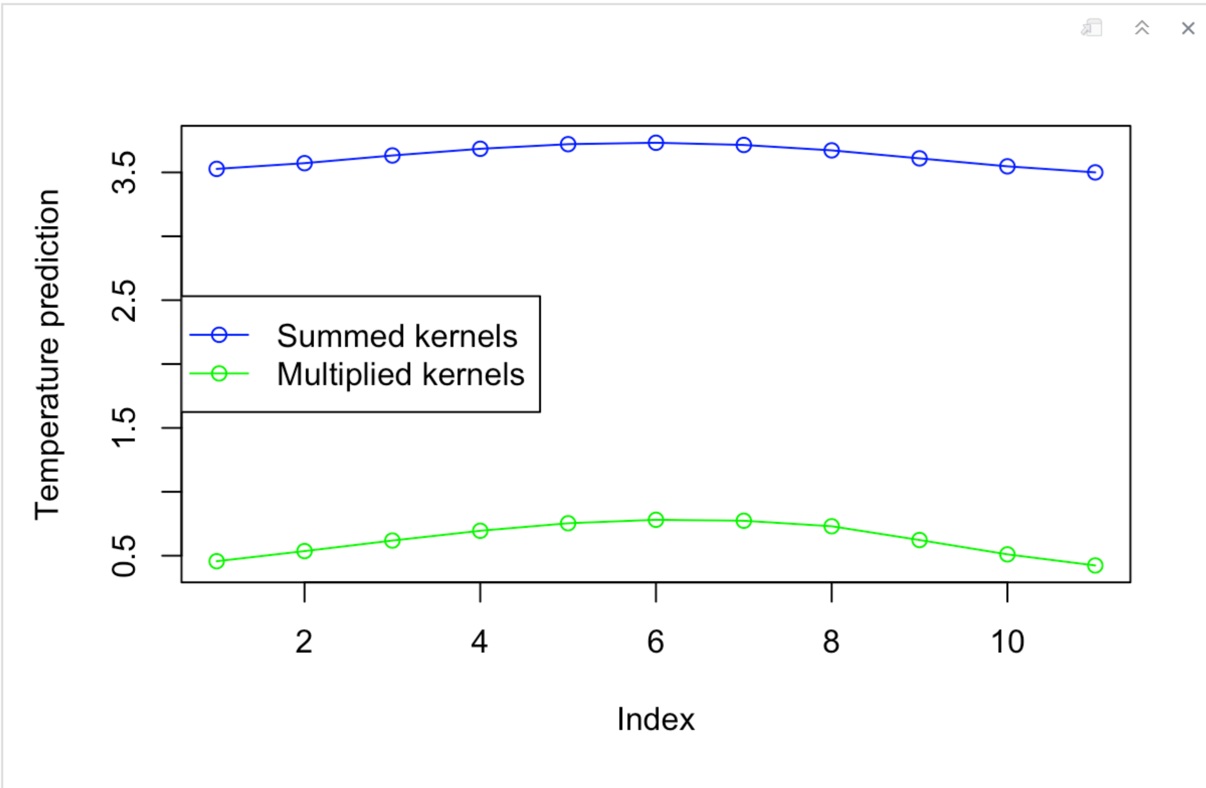




The width of kernels were chosen so that the weight had decreased to 0.5 at a certain distance I figured was reasonable.

```
Most weighted predictions at  08:00:00 with summed kernels:
          distance               date               time
"64209.3475626027"       "2012-12-28"         "08:00:00"
          distance               date               time
"67647.8518002752"       "1998-12-26"         "09:00:00"
          distance               date               time
"64428.0670248181"       "1994-12-20"         "09:00:00"


Most weighted predictions at  08:00:00 with multiplied kernels:
          distance               date               time
"64209.3475626027"       "2012-12-28"         "08:00:00"
          distance               date               time
"67647.8518002752"       "1998-12-26"         "09:00:00"
          distance               date               time
"64428.0670248181"       "1994-12-20"         "09:00:00"


Most weighted predictions at  14:00:00 with summed kernels:
          distance               date               time
"55443.5735022029"       "2011-12-31"         "14:00:00"
          distance               date               time
"55443.5735022029"       "2001-12-24"         "15:00:00"
          distance               date               time
"64209.3475626027"       "2010-12-22"         "15:00:00"


Most weighted predictions at  14:00:00 with multiplied kernels:
          distance               date               time
"55443.5735022029"       "2011-12-31"         "14:00:00"
          distance               date               time
"55443.5735022029"       "2001-12-24"         "15:00:00"
          distance               date               time
"64209.3475626027"       "2010-12-22"         "15:00:00"


Most weighted predictions at  20:00:00 with summed kernels:
          distance               date               time
"67647.8518002752"       "2007-12-25"         "21:00:00"
          distance               date               time
"67647.8518002752"       "1973-12-16"         "21:00:00"
          distance               date               time
"55443.5735022029"       "2009-12-23"         "18:00:00"


Most weighted predictions at  20:00:00 with multiplied kernels:
          distance               date               time
```

```
"67647.8518002752"       "2007-12-25"        "21:00:00"
          distance               date              time
"67647.8518002752"       "1973-12-16"        "21:00:00"
          distance               date              time
"55443.5735022029"       "2009-12-23"        "18:00:00"
```



The contribtution from each temperature measurement with the summed kernel gets a higher average weight, compared to the multiplied. This is because if any of the kernels generate weight near zero or equal to zero, the entire weight for that measurement becomes zero with the multiplied kernel

Assignment 3: Neural Networks

Question Assignment 3: Train neural networks to learn the trigonometric sine function. Perform early stop of the gradient descent using the threshold argument. Consider thresholds i/10000, where i = 1...10. Initialize the weights of the neural networks to random values in the intervall [-1,1]. Use a hidden layer of 10 units. Choose the most appropriate value for the threshold. Motivate your choice.

Answer Assignment 3: To answer the question, the library neuralnet was used (as suggested in assignment). To evalute the different neuralnets, the least square of model predictions was calculated for each model, and the model that is most appropriate is the model with the least square, for the validation set. The least square for each model is plotted in the following plot:

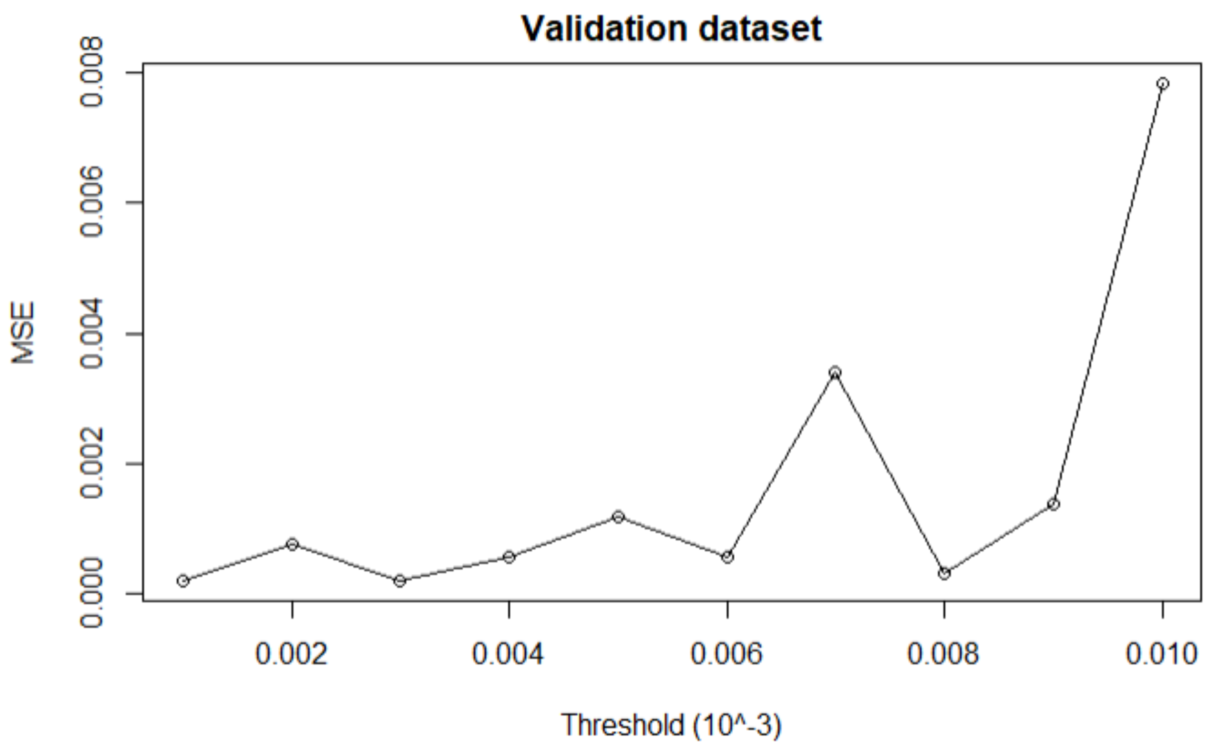


Figure 1, the squared error of model predictions vs. true values for validation set, for different thresholds

As we can see, the model performance differs for different thresholds. The smallest squared error is identified by:

which.min(SSE)

Where SSE is the vector containing the squared errors, indexed after i used in threshold. The result is that index 1 has the lowest SSE, with SSE[1] = 0.00521 and threshold = 1/1000. The resulting neural network has the following structure:

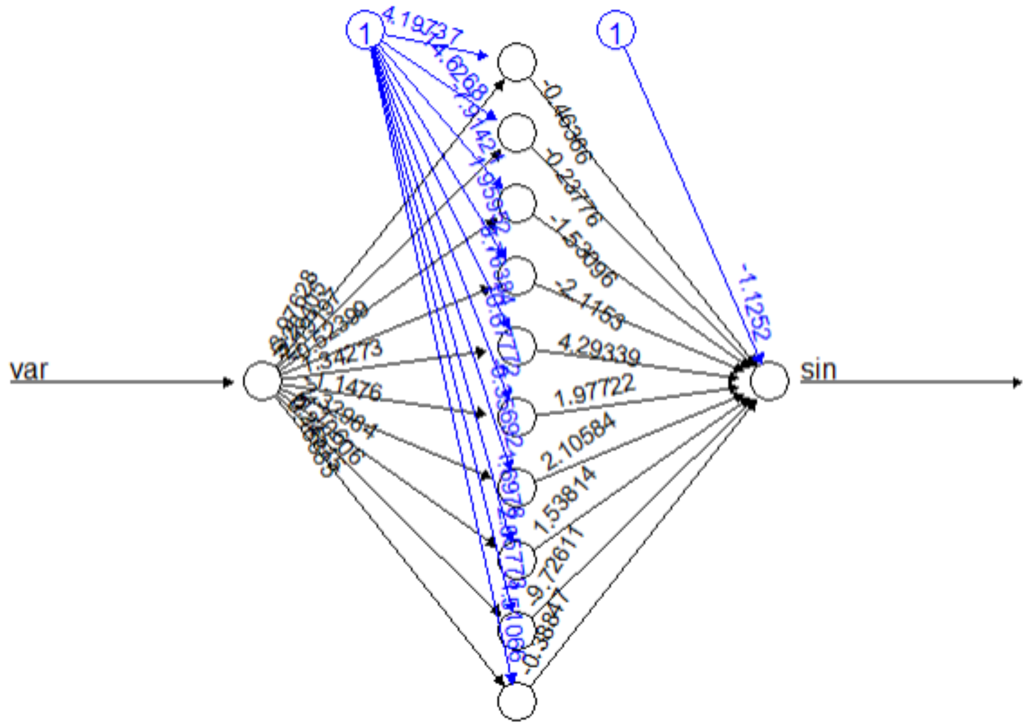
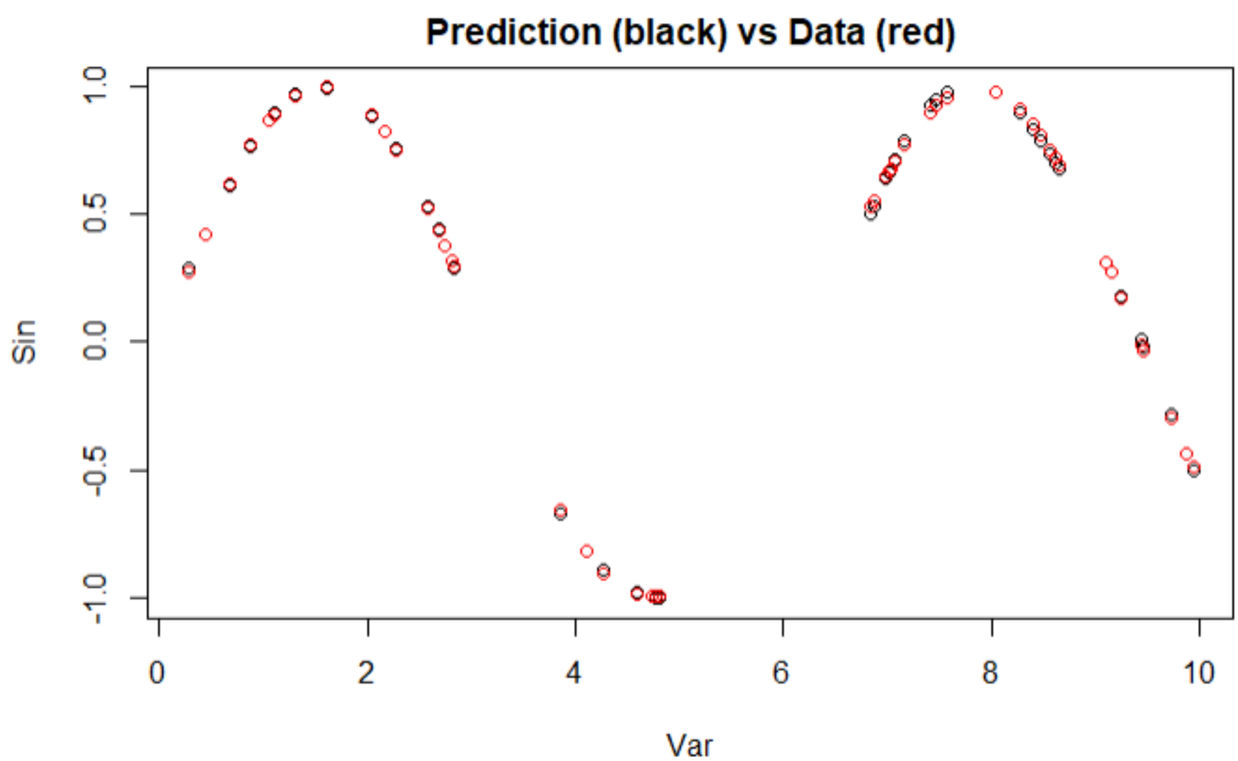


Figure 2, the structure of the best neural network

The resulting function has the following plot:



Appendix 1: code for assignment 3

```
###############
##    1.1    ##
###############
```

<u>DETERMING PARAMETERS</u>

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

gaussian_kernel <- function(distance, h){
  return (exp(-(distance/h)^2))
}

#https://en.wikipedia.org/wiki/Geographical_center_of_Sweden
lat <- 62.173276
long <- 14.942265

stations <- read.csv(file="stations.csv", fileEncoding="latin1", stringsAsFactors=FALSE)
stations <- cbind(stations,distHaversine(p1=stations[,c("latitude","longitude")],p2= c(lat,long)))
colnames(stations)[ncol(stations)] <- "distance"

temps <- read.csv(file="temps50k.csv", stringsAsFactors=FALSE)

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

edge_stations <- vector(length=4)
edge_stations[1] <- stations[which.max(stations$latitude),"station_number"]
edge_stations[2] <- stations[which.min(stations$latitude),"station_number"]
```

```
edge_stations[3] <- stations[which.max(stations$longitude),"station_number"]

edge_stations[4] <- stations[which.min(stations$longitude),"station_number"]


threshold_station <- round(nrow(stations)/2)

max_distance <- 0


threshold_distances <- vector(length = length(edge_stations))


for (i in 1:length(edge_stations)){

  distances <- vector(length = nrow(stations))

  for (j in 1:nrow(stations)){

  distances[j] <- 
distHaversine(stations[which(stations$station_number==edge_stations[i]),c("longitude","latitude")], 
stations[j,c("longitude","latitude")])

  }

  distances <- sort(distances,decreasing = FALSE)

  threshold_distances[i] <- distances[threshold_station]

  max_distance <- round(ifelse(max_distance<max(distances),max(distances), max_distance))

}

threshold_distance <- mean(threshold_distances)


h_distance <- 820000


plot(x = seq(0,max_distance,length.out = 1000), y = gaussian_kernel(distance = 
seq(0,max_distance,length.out = 1000), h = h_distance), ylab = "Weight", xlab = "distance [m]")

points(x=threshold_distance, y = gaussian_kernel(distance = threshold_distance,h_distance), col="red", 
type = "p", cex=1.5)

text(x=threshold_distance*1.5, y = gaussian_kernel(distance = threshold_distance,h_distance)*1.5, 
col="red",labels= paste("Average distance to", threshold_station,"th closest station"))

points(x=threshold_distance, y = 0.5, col="green", type = "p", cex=2)
```

text(x=threshold_distance/2, y = 0.5, col="green", labels = "Weight=0.5")

h_date <- 107

plot(x=seq(0,180,length.out = 1000), y = gaussian_kernel(seq(0,180,length.out = 1000), h_date), ylab = "Weight", xlab = "Day difference", xlim=c(0,180), ylim=c(0,1))

points(x=90, y = 0.5, col="green", type = "p", cex=2)

text(x=60, y = 0.5, col="green", labels = "Weight = 0.5")

points(x=90, y = gaussian_kernel(90, h_date), col="red", type = "p", cex=1.5)

text(x=130, y = gaussian_kernel(90, h_date), col="red", labels = "90 days difference")

h_time <- 14.5

plot(x=seq(0,24,length.out = 1000), y = gaussian_kernel(seq(0,24,length.out = 1000), h_time), ylab = "Weight", xlab = "Hour difference", xlim=c(0,24), ylim=c(0,1))

points(x=12, y = 0.5, col="green", type = "p", cex=2)

text(x=8, y = 0.5, col="green", labels = "Weight = 0.5")

points(x=12, y = gaussian_kernel(12, h_time), col="red", type = "p", cex=1.5)

text(x=18, y = gaussian_kernel(12, h_time), col="red", labels = "12 hours difference")

Predicting temperatures

date <- "2015-12-24" # 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")

temp1 <- vector(length=length(times))

temp2 <- vector(length=length(times))

```
# Students’ code here

st <- st[which(st$date<=date),]

for (t in 1:length(times))
{
  exclude <- which((st$date==date)&(st$time>=times[t]))
  if (length(exclude)>0) {
    data <- st[-exclude,]
  } else {
    data <- st
  }

  rownames(data) <- 1:nrow(data)

  my_matrix <- matrix(NA,nrow=nrow(data), ncol=4)
  colnames(my_matrix) <- c("temperature", "k_sum", "k_prod", "index")

  for (s in 1:nrow(data)){
    k1 <- gaussian_kernel(data[s,"distance"], h_distance)

    days_diff <- as.numeric(as.Date(date)-as.Date(data[s,"date"]))%%365.25
    days_diff <- min(365-days_diff, days_diff)
    k2 <- gaussian_kernel(days_diff, h_date)

    hour_diff <- as.numeric(strptime(times[t],"%H:%M:%S")-
strptime(data[s,"time"],"%H:%M:%S"))%%24
    hour_diff <- min(24-hour_diff, hour_diff)
```

```
    k3 <- gaussian_kernel(hour_diff,h_time)
    my_matrix[s,] <- c(data[s,"air_temperature"], k1+k2+k3, k1*k2*k3, s)
  }

  my_matrix[,"k_sum"] <- my_matrix[,"k_sum"]/sum(my_matrix[,"k_sum"])
  my_matrix[,"k_prod"] <- my_matrix[,"k_prod"]/sum(my_matrix[,"k_prod"])

  prediciton <- my_matrix[,"temperature"]*my_matrix[,"k_sum"]
  temp1[t] <- sum(prediciton)
  prediciton <- my_matrix[,"temperature"]*my_matrix[,"k_prod"]
  temp2[t] <- sum(prediciton)

  my_matrix <- my_matrix[order(my_matrix[,"k_sum"], decreasing = TRUE),]

  if (t%%3 == 0) {

    cat("\n\nMost weighted predictions at ", times[t],"with summed kernels:\n")

    for (i in 1:3)
    {
      print(unlist(data[my_matrix[i,"index"],c("distance","date", "time")]))
    }

    my_matrix <- my_matrix[order(my_matrix[,"k_prod"], decreasing = TRUE),]

    cat("\n\nMost weighted predictions at ", times[t],"with multiplied kernels:\n")

    for (i in 1:3)
    {
```

```
        print(unlist(data[my_matrix[i,"index"],c("distance","date", "time")]))
      }
   }
}
```

Plot

```
ylim = c(min(min(temp1),min(temp2)), max(max(temp1),max(temp2)))
plot(temp1, type="o", ylim=ylim, col = "blue", ylab="Temperature prediction")
legend("left", legend=c("Summed kernels", "Multiplied kernels"),col=c("blue", "green"), lty = c(1,1), pch
= c(1,1))
lines(temp2, type="o", col = "green")
```

```{r}

```
#### Assignment 3 ####

library("neuralnet")

set.seed(1234567890)
Var = runif(50, 0, 10)
data = data.frame(Var, Sin=sin(Var))
train = data[1:25,]
validation = data[26:50,]

# Random initialization of the weights in the interval [-1, 1]
winit = runif(10,-1,1)

mse = function(prediction, observation)
{
```

```
  return (mean((observation - prediction)^2))
}

n = 10
mse_val = numeric()
mse_train = numeric()
threshold = numeric()

for(i in 1:n)
{
  nn = neuralnet(Sin ~ Var, data = train, startweights = winit, hidden = 10, threshold = i/1000)
  pred_train = compute(nn, as.matrix(train$Var))$net.result
  pred_val = compute(nn, as.matrix(validation$Var))$net.result
  threshold[i] = i/1000
  print(i)
  mse_val[i] = mse(pred_val, validation$Sin)
  mse_train[i] = mse(pred_train, train$Sin)
}

plot(threshold, mse_val, type="o", ylab="MSE", xlab="Threshold (10^-3)", main = "Validation dataset")
plot(threshold, mse_train, type="o", ylab="MSE", xlab="Threshold (10^-3)", main = "Training dataset")

# Train network with full dataset
nn = neuralnet(Sin ~ Var, data = data, startweights = winit, hidden = c(10), threshold = 4/1000)
plot(nn)

# Predictions
```

~~~
plot(prediction(nn)$rep1, col="black", main = "Prediction (black) vs Data (red)")

points(data, col = "red")


````
~~~