Jacob Larsson (Assignment 1) jacla460
Erik Matti (Assignment 2) erima329
Oskar Magnusson (Adjuster) oskma287

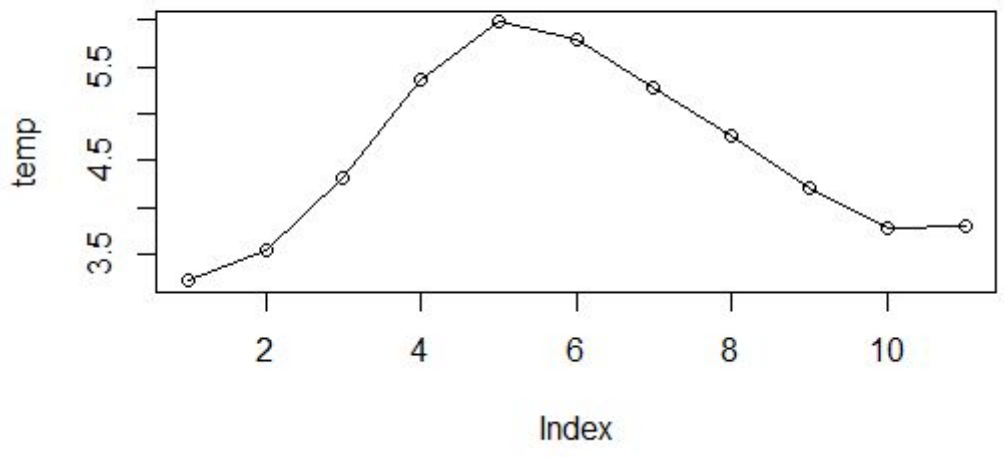
Group B11 Report - Lab 3

Assignment 1: Kernel Methods

First off, both of the csv files were loaded in and then merged into a single big data set. Then a date and a set of coordinates were chosen which our algorithm is going to estimate the temperature for. Since we are going to use the code template given to us, we'll be using the date and the coordinates provided in the template. To estimate the temperature curve we created a loop that goes through each row in the data set, that is each temperature measurement in the given data files, and calculates the distance in position, date and time from the target, which is then used together with each respective kernel width to calculate the value of a gaussian kernel for each distance. Using the equation:

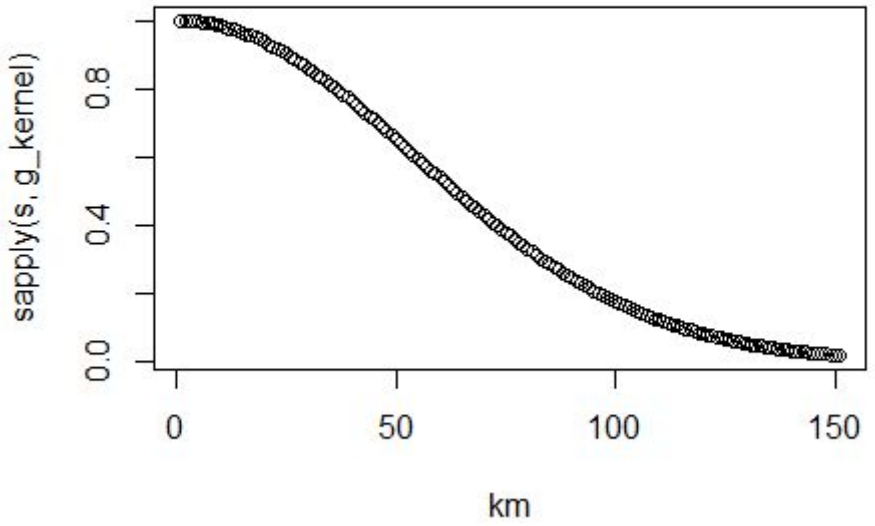
$$y_k(x) = \frac{\sum_n k(\frac{x-x_n}{h})t_n}{\sum_n k(\frac{x-x_n}{h})}$$

Where k is the sum of the gaussian kernels and t is the temperature of that row of data. The resulting temperature curve for the date 2013-11-04 at the coordinates 58.4274, 14.826 was:

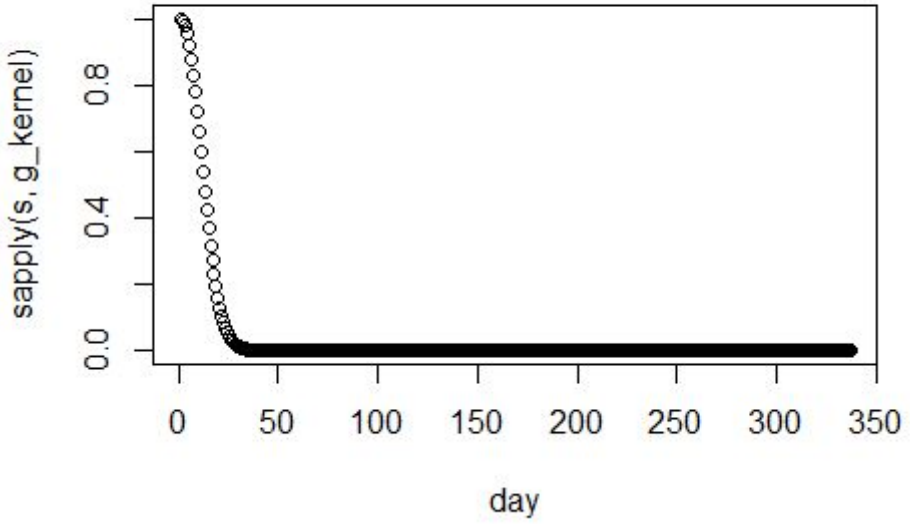


As we can see in the plot above the temperature span is between 3 and 6 degrees which is reasonable for early november and it peaks during midday which is also reasonable. So we can deduce that the algorithm works as intended.

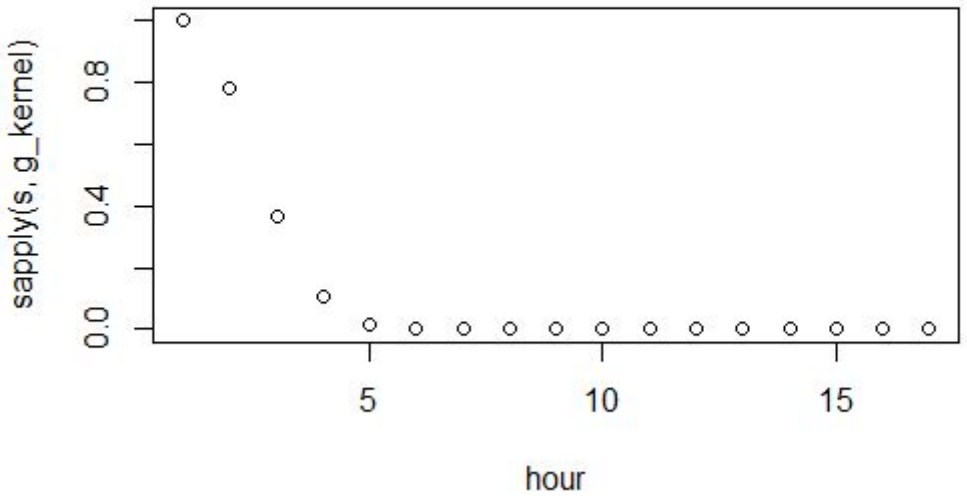
The kernel widths were mostly chosen based on what "feels" sensible. The distance width was chosen to be 75 000, which means that we only take into account the weather in places about 150 km away. This is a reasonable distance, for example Linköping and Jönköping is about that distance away from each other and within certain time frames it is usually similar weather in both cities, therefore 75 000 works well. A plot of the kernel can be seen below.



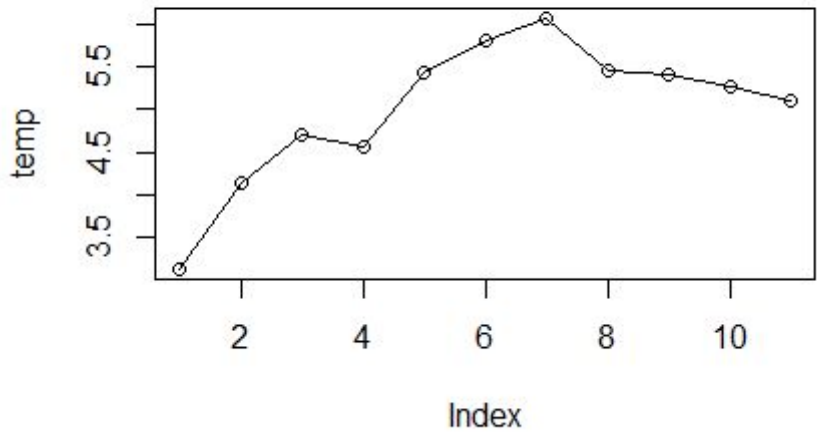
The width for the date kernel is 14, which means that we care about dates within about a month from the target date. This is also reasonable since the weather in neighbouring months are usually pretty similar. A modulo operator has been used when calculating the distance so that similar dates from other years are taken into account as well, since the weather during a month is usually similar to the previous year. A plot for its gaussian over a year can be seen below



The width for the time kernel is set to 2, which means 2 hours, so that we only care about the temperature during similar hours. This is because the temperature rarely change so drastically that it is much different in that time interval, but we know that there is always a difference in temperature during the night compared to the day so we try to avoid using day temperatures when we are interested in night temperatures. One could argue that we should use the entire day since temperatures rarely change more than a couple of degrees, which isn't that much compared to something like using a wider interval of dates. But, since we have access to so much data we should not have a problem with getting data from a smaller interval. Below we can see the plot for the time kernel over 24 hours.



After this we were supposed to combine the three kernels by multiplication instead of summation. The resulting plot can be seen below. The most notable things about this plot is that it is not as smooth as the previous one and that the start and end nodes do not have similar temperatures. The reason for this is probably because the multiplication causes the algorithm to be more extreme in how much it cares about each data row, so all distances has to be very close for it to affect the calculated plot and those far away are basically disregarded completely. Before if only one distance was close we would take that data row into account, at least a bit. This means that using multiplication is probably a better solution, but we would have to increase the width of some of the kernels since it looks like it's too narrow in plot below.



Jacob Larsson (Assignment 1) jacla460
Erik Matti (Assignment 2) erima329
Oskar Magnusson (Adjuster) oskma287

Assignment 3: Neural Networks

By following the instructions from assignment 3, a neural network could be calculated as described. By creating ten different neural networks with ten different values for the threshold, from 0.001 to 0.01, the minimum square error could be calculated for each of them and compared with. The results can be seen in figure 3. As can be observed, the best threshold which gives the least amount of MSE is with index 4, which gives the threshold 4/1000 = 0.004. Because it gives the least amount of MSE, it is therefore the most appropriate neural network of the ten. Figure 5 shows the final neural network and figure 4 shows the difference between the actual data with red points together with the predicted data with black points. By observing figure 4, one can observe that the predicted data is very similar to the real data and therefore the neural network was good.

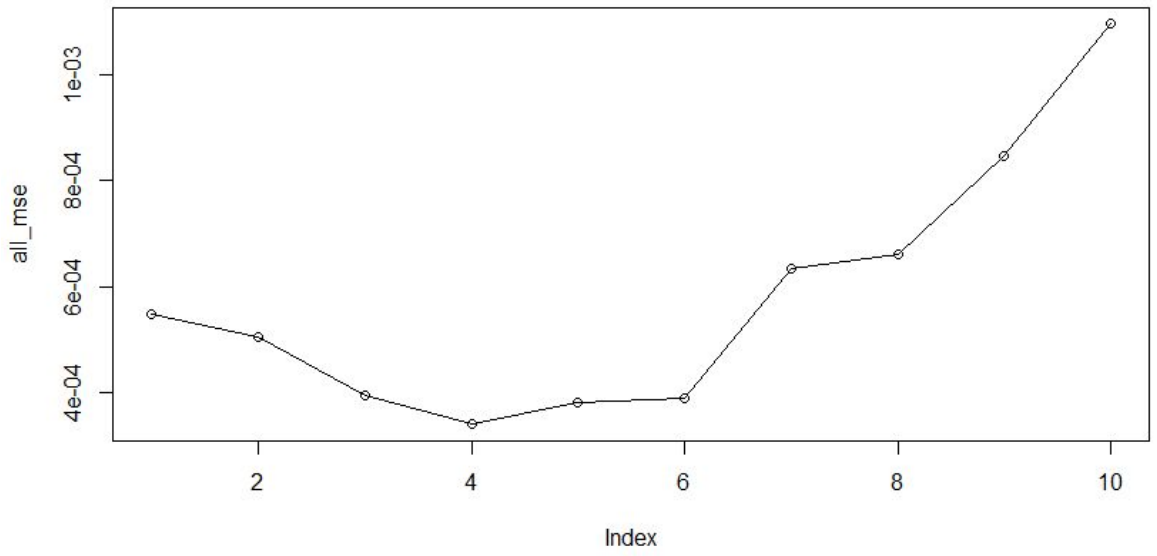


Figure 3 - MSE-values for ten different thresholds

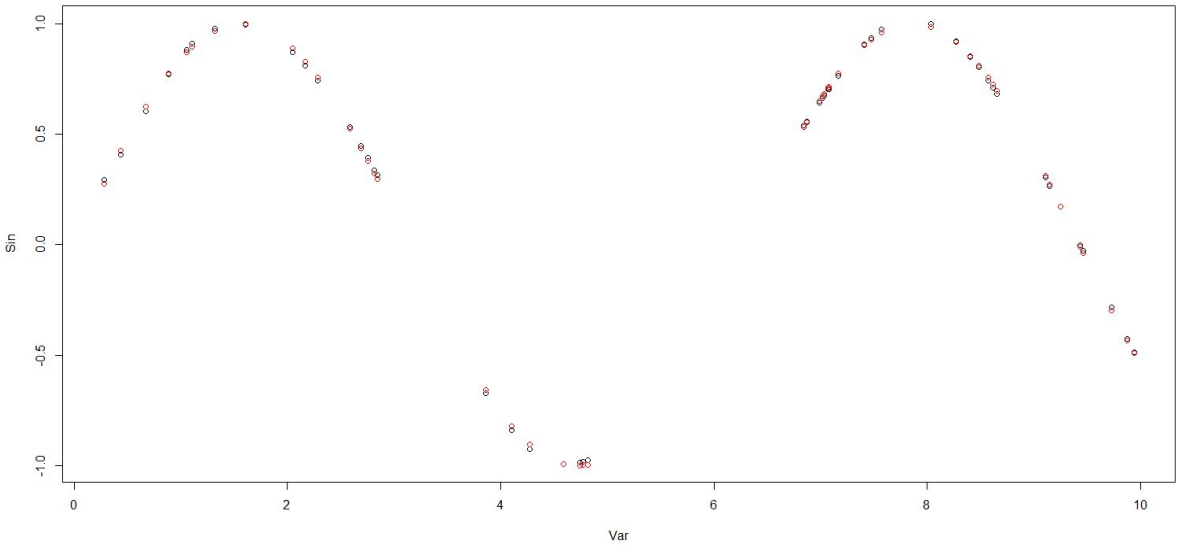


Figure 4 - Comparing the data (red) with the prediction (black)

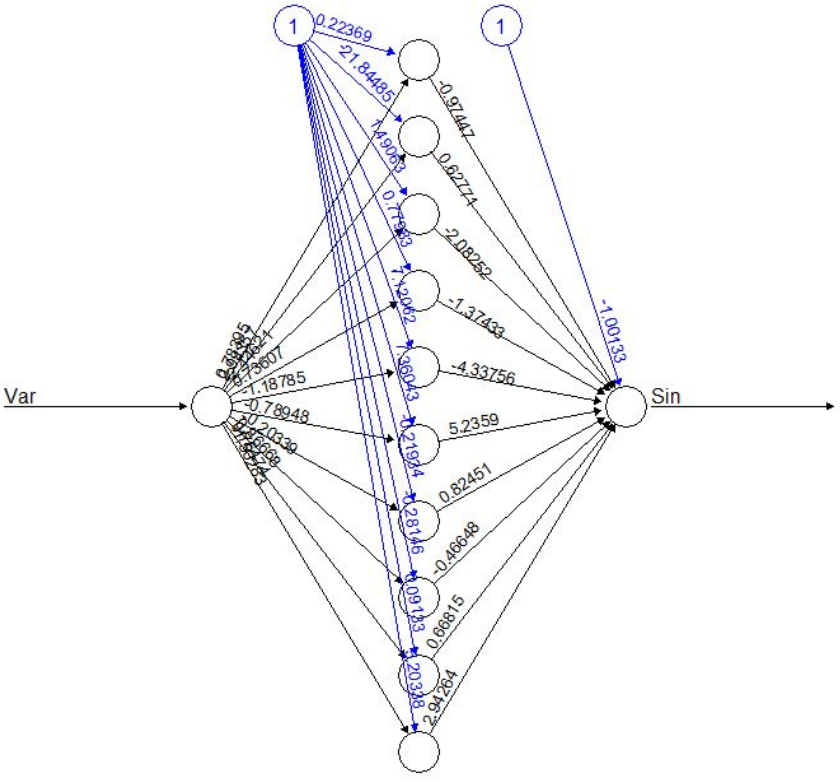


Figure 5 - The optimal neural network

Appendix

Assignment 1

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

temps <- read.csv(file="D:/Jacob/Documents/R Filer/lab3/temps50k.csv", header=TRUE,
stringsAsFactors=FALSE, fileEncoding="latin1")
stations <- read.csv(file="D:/Jacob/Documents/R Filer/lab3/stations.csv", header=TRUE,
stringsAsFactors=FALSE, fileEncoding="latin1")
st <- merge(stations, temps, by="station_number")
#plot(st$longitude, st$latitude, col = st$air_temperature - min(st$air_temperature))

# H values chosen by student
h_distance <- 75000
h_date <- 30
h_time <- 2
a <- 58.4274
b <- 14.826
sdate <- "2013-11-04"
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")

# Temperature curve
temp <- vector(length=length(times))
# Filter away the future
st = subset(st, as.vector(difftime(sdate, date, units='days')) > 0, select = c(latitude, longitude, date,
time, air_temperature))

# Gaussian kernel: k(u) = exp(-||u||^2) where ||.|| is the euclidean norm
g_kernel <- function(u){
  exp(-(sqrt(sum(u^2)))^2)
}
val_vec = vector(length=length(times))
n_val_vec = vector(length=length(times))
cat("Target date: ");print(sdate);cat("\n")

for (i in 1:nrow(st)){
  dist_norm = distHaversine(c(st[i,1], st[i,2]), c(a, b)) / h_distance
  date_diff_norm = as.numeric(difftime(sdate, st[i,3], units='days'))%% 365 / h_date

  for (j in 1:length(temp)){
    time_diff_norm = as.numeric(difftime(strptime(times[j], format="%H:%M"), strptime(st[i,4],
format="%H:%M"), units='hours')) / h_time
```

```
    val_vec[j] = val_vec[j] + prod(g_kernel(dist_norm), g_kernel(date_diff_norm),
g_kernel(time_diff_norm))*st[i,5]
    n_val_vec[j] = n_val_vec[j] + prod(g_kernel(dist_norm), g_kernel(date_diff_norm),
g_kernel(time_diff_norm))
  }
}
temp = (val_vec/n_val_vec)
plot(temp, type="o")

# Show that the kernels' are sensible

# In the plot below we can see the gaussian curve
s = seq(0,2,1000/h_distance)
plot(sapply(s,g_kernel), xlab = "km")

# A plot for its gaussian over a year can be seen below
s = seq(0,2*12,1/h_date)
plot(sapply(s,g_kernel), xlab = "day")

# Below we can see the plot for the time kernel over 24 hours.
s = seq(0,2*4,1/h_time)
plot(sapply(s,g_kernel), xlab="hour")
```

Assignment 3

```
RNGversion("3.5.1")
library(neuralnet)
set.seed(1234567890)
Var <- runif(50, 0, 10)
trva <- data.frame(Var, Sin=sin(Var))
tr <- trva[1:25,] # Training
va <- trva[26:50,] # Validation

winit <- runif(31, -1, 1)

calculate_MSE <- function(obs, pred) {
  mean((obs - pred)^2)
}

all_mse <- c()
min_mse <- 10000000
best_threshold <- 0

for(i in 1:10) {
```

```
  nn <- neuralnet(Sin ~ Var, tr, threshold = i/1000, startweights=winit, hidden = 10)
  prediction <- predict(nn, va)
  temp_mse <- calculate_MSE(va$Sin, prediction)
  all_mse[i] <- temp_mse
  if (temp_mse < min_mse) {
    min_mse <- temp_mse
    best_threshold <- i/1000
  }
}

plot(all_mse, type="o")

best_nn <- neuralnet(Sin ~ Var, trva, threshold = best_threshold, startweights=winit, hidden
= 10)

plot(best_nn)

# Plot of the predictions (black dots) and the data (red dots)
plot(prediction(best_nn)$rep1)
points(trva, col = "red")
```