Lab 3 - Elias Alesand, elial148

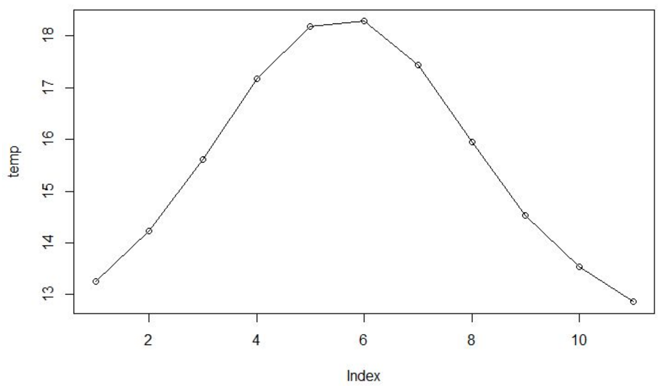
Assignment 1

I use meters, days and seconds as units for my kernels and i chose the following h-values:
h_distance = 180000
h_date = 20
h_time = 15000

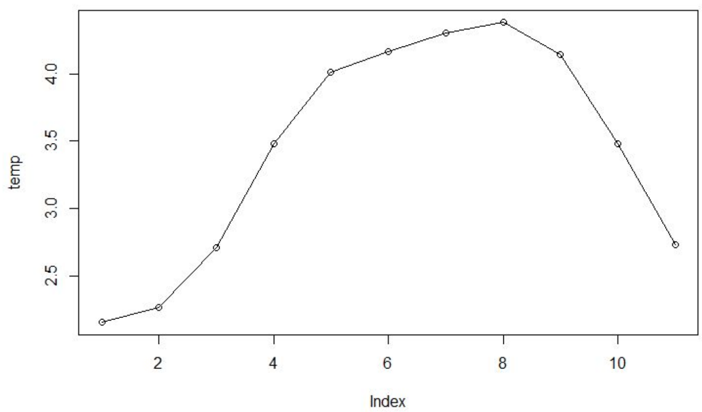
I chose these values in part by trial and error and by looking at how the results changed when i changed these parameters and also by doing some reasoning about how the different data looked like and how the h values should be chosen accordingly.

Using these h-values and a kernel obtained by multiplying the three original kernels i get the following graphs:

Linköping(58.4000, 15.5760) 2015-07-05



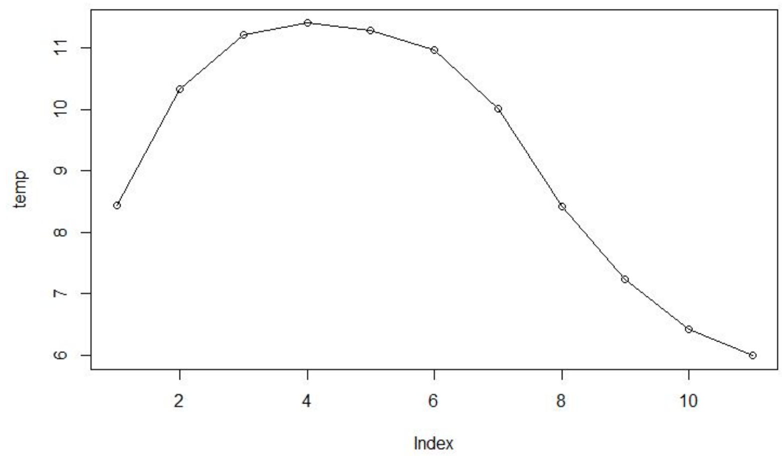
Linköping(58.4000, 15.5760) 2014-01-01



I see that changing the date to a winter day from a summer day really changes the predicted temperatures so the kernel definitely gives more weight to dates closer in time to the predicted date. The weights for the time of day also seems to make sense since it gradually gets warmer towards the middle of the day and then gets colder towards the night.

I will now see what happens if i change the location instead of the date.

Kiruna(67.8500, 20.2400) 2015-07-05

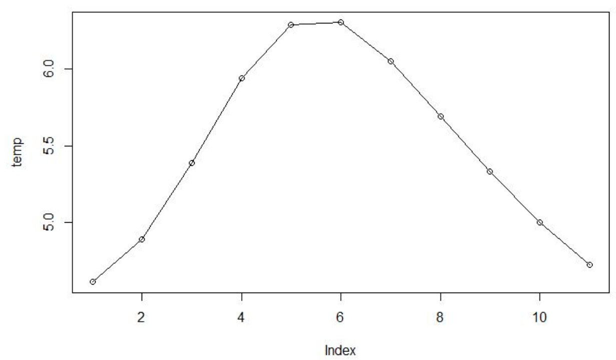


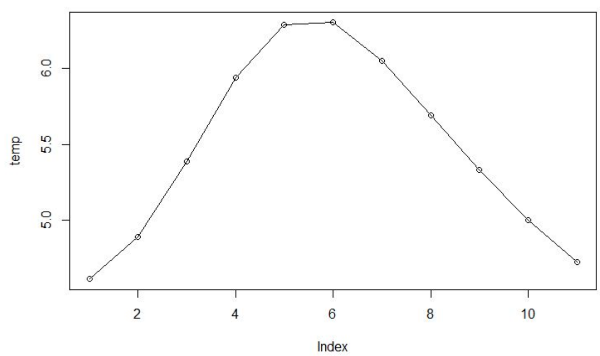
When using the same date but with with a location in the north of Sweden the temperature is lower for any point in time on the day and the warmest time of day is a bit earlier than the previous plot. These results are reasonable since the temperature should be lower on average.

When using the sum instead of product and the same parameters in the first two plots i get the following graphs:

Linköping(58.4000, 15.5760) 2015-07-05

Linköping(58.4000, 15.5760) 2014-01-01

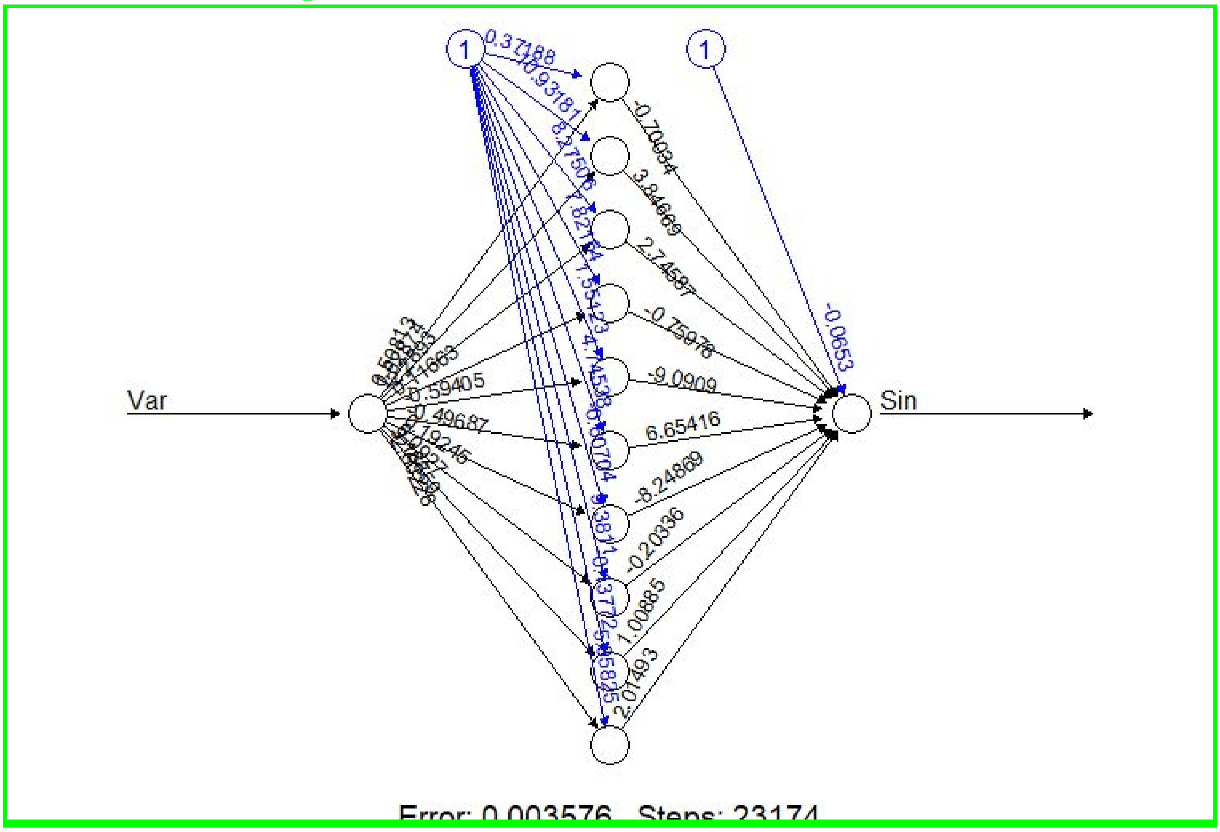




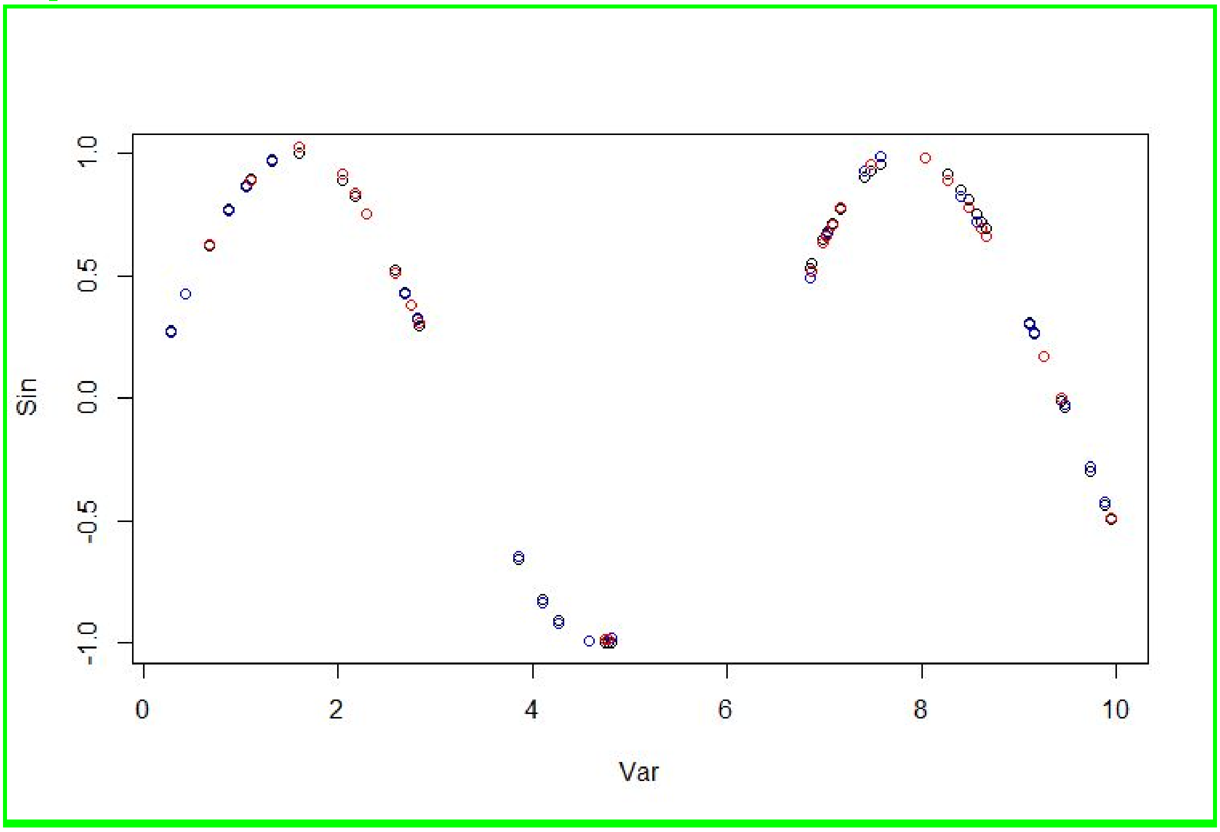
I see no real difference between the results and neither of them seem probable (maybe the right one if it is a warm winter) so either this is not a problem where sum should be used or the h values has to be altered for a summation to make sense. When using a product i think every kernel value has to be fairly high or else the value will be dragged down by a low scoring kernel. For a summation, two values can be high and the third low while still resulting in a high value. Therefore i don't think a summation is appropriate for this problem since only the points where all kernels score high should be used to predict the weather in my opinion.

Assignment 2 Changes are marked in green. In the code i only removed a line which set the seed.

I use Mean-square error(MSE) to evaluate the quality of the neural network. Then i choose the threshold which results in the lowest error. A threshold of 4/1000 results in a MSE of 0.00034 which is the lowest of the ten different examined thresholds. The network that this results in looks like this:

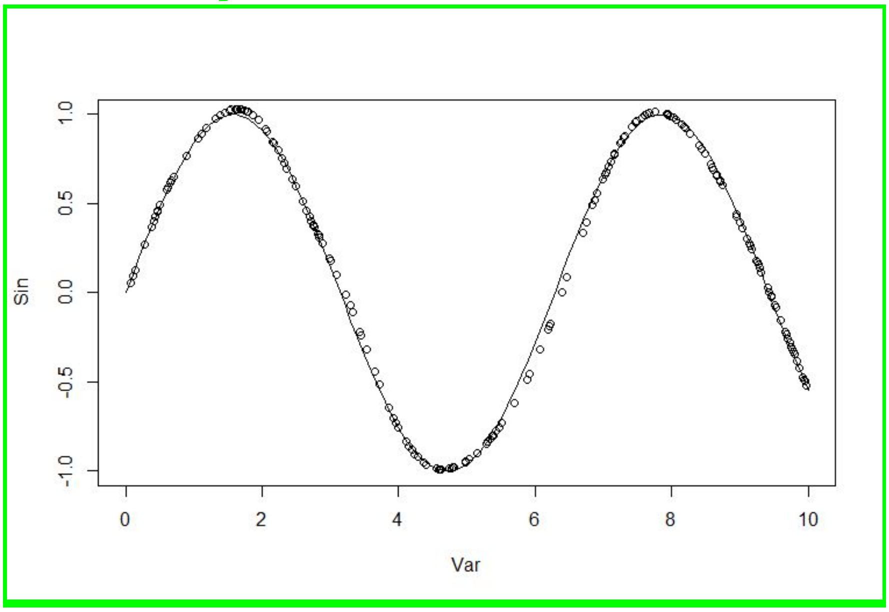


And the plot of the predicted data points for the train(blue) and validation(red) sets looks like this:



It is a bit hard to see but the red and blue dots both follow the original(black) data points very closely so the network seems to work well.

Finally, i tried randoming 200 values in the range [1:10] to see how the neural network performs over the whole interval and i plot the predicted Sin values for them together with an actual sine wave:



You can see that the predictions follow the sine wave pretty well over the whole interval which suggests that the neural network performs well with the chosen threshold.

Code Appendix

Assignment 1:

```
set.seed(1234567890)
library(geosphere)
stations <- read.csv("stations.csv")

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

#Linköping
a= 58.4000
b= 15.5760
#Kiruna
#a=67.8500
#b=20.2400

h_distance <- 180000# These three values are up to the students
h_date <- 20
h_time <- 15000

date <- "2014-01-01" # The date to predict (up to the students)
date_split = strsplit(date,"-")
date_split=sapply(date_split,as.numeric)
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))

# Students' code here
st <- merge(stations,temps,by="station_number")
#Converting to dates to filter out future predictions with a simple "<" comparison
st$date=as.Date(st$date,format="%Y-%m-%d")
st=st[st$date<date,]
st$date=as.character(st$date)
st$time=as.character(st$time)

#Returns a value according to a gaussian distribution
gaussian_kernel = function(u){
  return(exp(-(u^2)))
}

#___DISTANCE____
distances = vector(length=length(st$longitude))
for (i in 1:length(st$longitude)){
  distances[i] = distHaversine(c(b,a),c(st$longitude[i],st$latitude[i]))/h_distance
}
kernel_distance=sapply(distances,gaussian_kernel)
#___DAYS_____
n_of_days = vector(length=length(st$date))
for (i in 1:length(st$date)){
  st_split=strsplit(st$date[i],"-")
```

```
  st_split=sapply(st_split,as.numeric)
  n_of_days[i] =
((date_split[1]-st_split[1])*365+(date_split[2]-st_split[2])*30+(date_split[3]-st_split[
3]))/h_date

  }
kernel_days=sapply(n_of_days,gaussian_kernel)
#___SECONDS____
for (t in 1:length(times)) {
  n_of_seconds = vector(length=length(st$time))
  time_split=strsplit(times[t],":")
  time_split=sapply(time_split,as.numeric)[1]
  for (i in 1:length(st$time)){
    st_split=strsplit(st$time[i],":")
    st_split=sapply(st_split,as.numeric)[1]
    #Have to use min to take care of cases like times 22:00 and 02:00 where the
difference
    #is 4 hours and cannot be calculated by a simple subtraction between the two times
    n_of_seconds[i] = (min(c(abs(24-time_split+st_split),
                        abs(24-st_split+time_split),
                        abs(time_split-st_split))))*60*60/h_time
  }
  kernel_seconds=sapply(n_of_seconds,gaussian_kernel)
  #___SUMMATION OF KERNELS
  kernel_sum=kernel_distance+kernel_days+kernel_seconds
  #___PRODUCT OF KERNELS
  #kernel_sum=kernel_distance*kernel_days*kernel_seconds
  temp[t] = sum(st$air_temperature*kernel_sum)/sum(kernel_sum)
}
plot(temp, type="o")
```

Assignment 2:

```r
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
# Random initialization of the weights in the interval [-1, 1]
winit = runif(31,-1,1)
# MSE used to store the mean square error of models
MSE=vector(length=10)
for(i in 1:10) {
  #Training using the training dataset
  nn <- neuralnet(Sin~Var,data=tr,hidden=10,threshold=i/1000,startweights = winit)
  #Predicting values for the validation dataset
  p = compute(nn,va$Var)
  #Calculating the error
  MSE[i]=mean((p$net.result-va$Sin)^2)
}
best_thresh=which.min(MSE)/1000
nn <- neuralnet(Sin~Var,data=tr,hidden=10,threshold=best_thresh,startweights = winit)
# Plot of the best neural network
plot(nn)
# Plot of the predictions (black dots) and the data (red dots)
plot(trva)
points(tr$Var,compute(nn,tr$Var)$net.result,col="blue")
points(va$Var,compute(nn,va$Var)$net.result,col="red")
# Predicting and plotting 200 random values in the same range as before
t=seq(0,10,0.1)
y=sin(t)
plot(t,y,ylab="Sin",xlab="Var",type="l")
set.seed(1234567890)
test <- runif(200, 0, 10)
points(test,compute(nn,test)$net.result)
```