TDDE01: Lab 5
2016-12-12
Simon Hadenius, simha713
930717-1471

The task in this lab was to implement a kernel method to predict the hourly temperatures for a date and place in Sweden using the files *stations.csv* and *temps50k.csv.* These contain information about temperatures at different weather stations at certain dates.

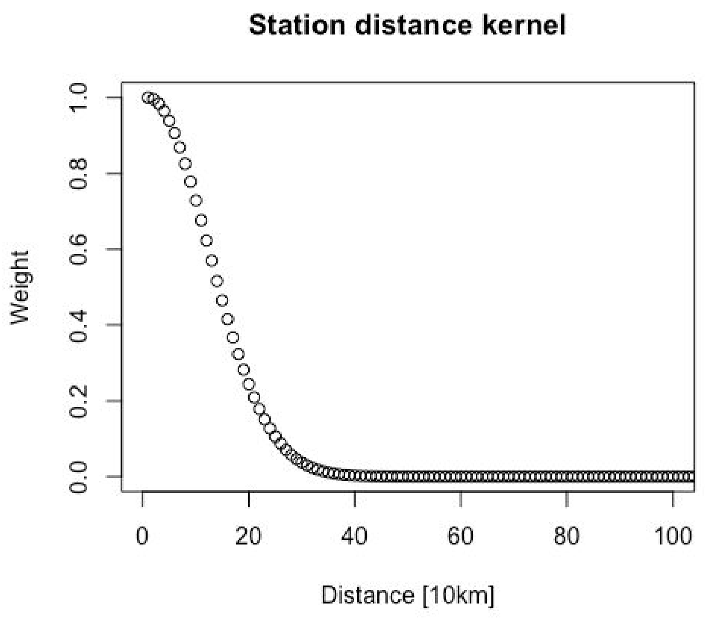
The kernel created was a sum of three Gaussian Kernels;

- The first accounting for the distance from a station to the point of interest.
- The second to account for the distance between a date and the day of interest.
- The third to account for the distance between an hour and the hour of interest.

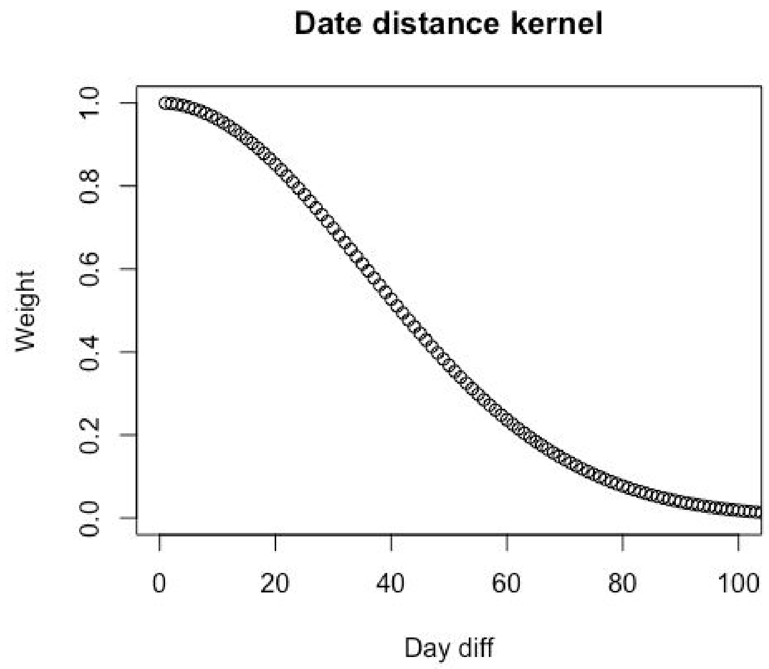
A smoothing coefficient was picked for each kernel. I picked the following:

Kernel	**h**
Station distance	160 000
Date distance	50
Hour distance	3 600

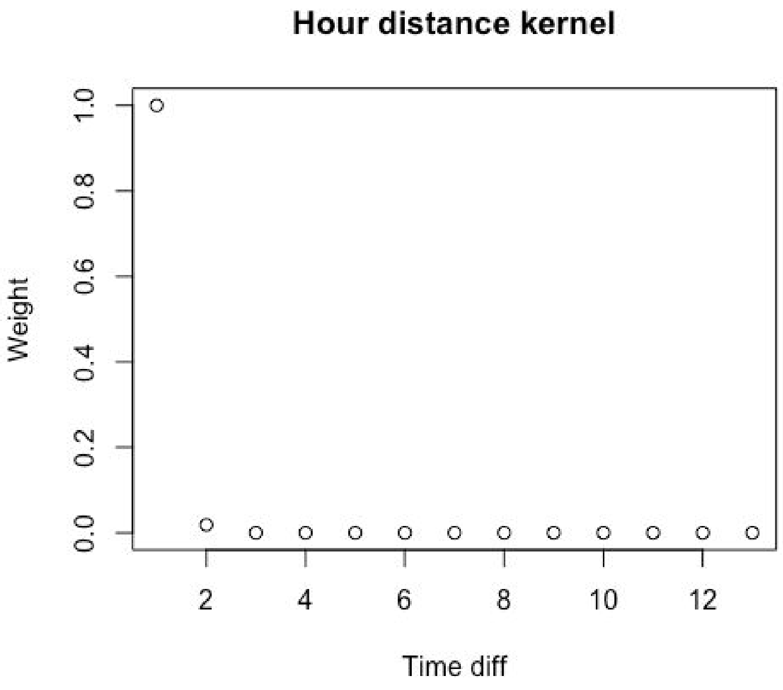
I plotted a graph for each h value to make it easier to see why I choose these values.



As you can see, the kernel accounts for distances up to 20 km. This sounds reasonable because weather stations with larger distances than 20 km can have big weather differences.



I realised that the date was the most important to predict the weather. Therefore I chose an h value that takes a lot of different dates into account. You can compare this to the different seasons, the weather is mostly the same during the summer, winter, etc.



Here, I only chose to account the for the exact hour. Due to the large data set (50 000 readings) I will have enough data for every hour to make this work.

I also picked the following constants to take into account how much each kernel affects the combined kernel

Kernel	**c**
Station distance	0.1
Date distance	1
Hour distance	0.1

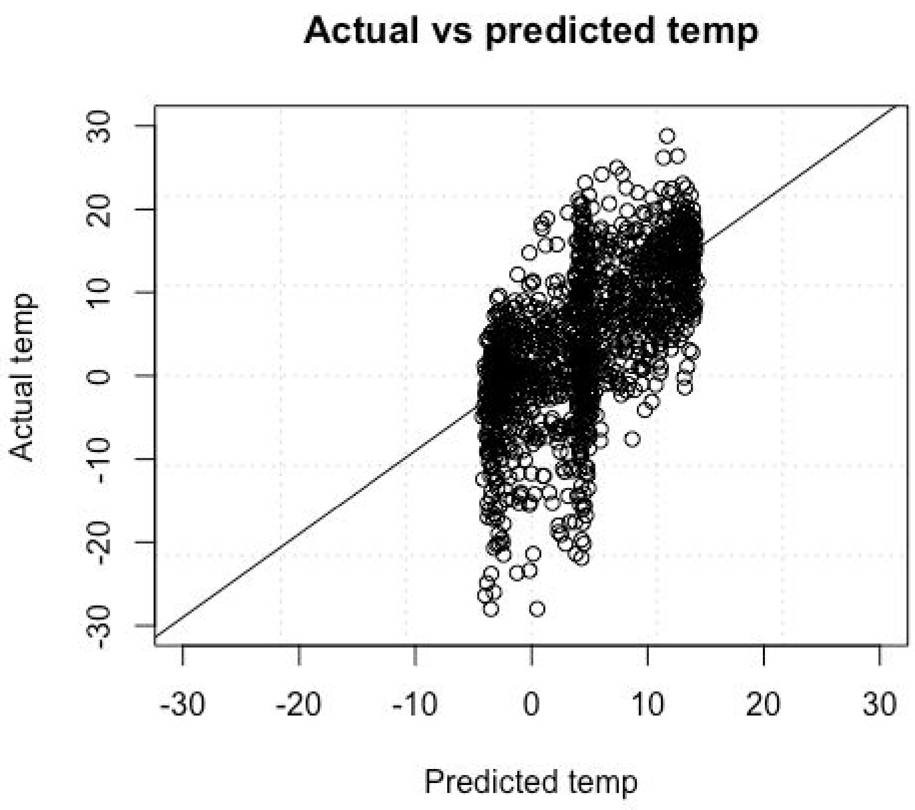
The following formula was used to calculate the kernel score:

$$K_{score} = c_{Station} * K_{Station} + c_{Date} * K_{Date} + c_{Hour} * K_{Hour}$$

Using this score a temperature was predicted using:

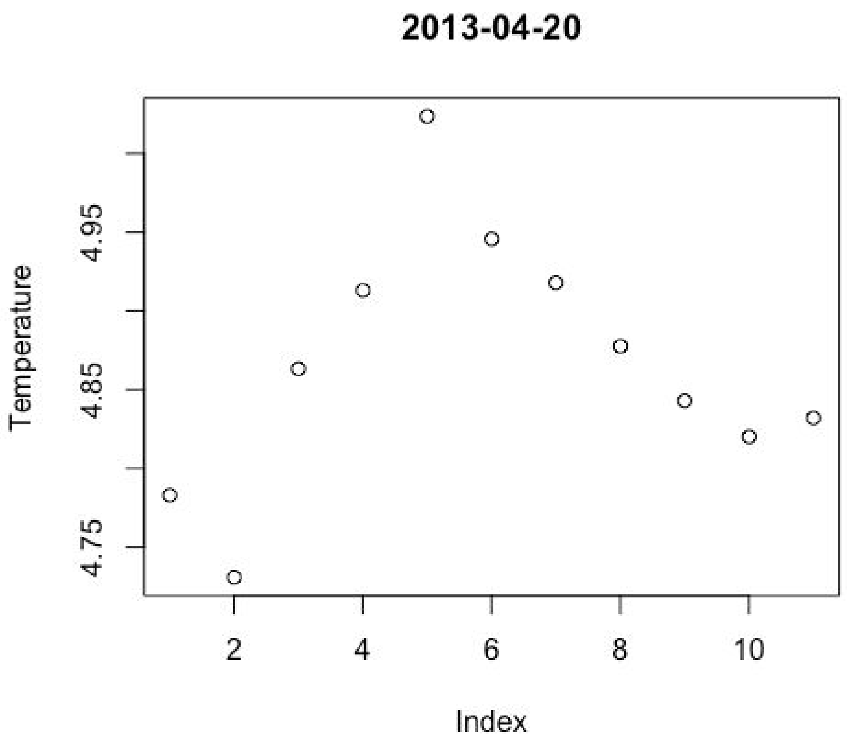
$$\widehat{t}(x) = \frac{\sum_{i=1}^{n} K(\frac{x_i - x}{h}) * y_i}{\sum_{i=1}^{n} K(\frac{x_i - x}{h})}$$

Using the formula above to predict the temperature for 5000 data points gave me the following plot:



Most predictions can be found between -5 and 15 degrees. The result seems reasonable considering the method I used is not viable to actually predict the weather.

I used the same model again to predict the weather for a specific date, 2013-04-20. It gave me the following result:



As seen in the graph, the temperature peaks around noon and falls of before and after. A problem with my kernels is that it predicts almost the same temperature every day of the year. The reason for that is that the average temperature is around 4 degrees.

A problem with my model is that the kernels are independent of each other.
A way to solve this problem is to multiply each kernel instead of summarizing them. By multiplying the kernels I make them dependent on each other and the predictions will probably be a bit better. Using these kernels will, however, never produce a really good result due to the information available and how it is used.

Appendix

Lab5-1.r

```
setwd("/Users/Simon/Documents/TDDE01/tdde01/Simon/lab5_assignment1")
set.seed(1234567890)
library(geosphere)
stations <- read.csv("stations.csv", fileEncoding = "latin1")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")

#ind <- sample(1:50000, 5000)
#st <- st[ind,]

gaussian_kernel = function(euclidian_dist) {
  return(exp(-((euclidian_dist))^2))
}

#Calculate the time between two dates
date_diff = function(t1, t2){
  return(as.numeric(difftime(strptime(t1, "%Y-%m-%d "),
                    strptime(t2, "%Y-%m-%d ")))%%365)
}

#Calculate the time between two hours
hour_diff = function(t1,t2){
  return(as.numeric(difftime(strptime(t1, "%H:%M:%S"),
                         strptime(t2, "%H:%M:%S"))))
}

#Distance from a station to the point of interest
k1 = function(p, stations, h){
  return(gaussian_kernel(distHaversine(p, stations)/h))
}

#Distance from a date to the date of interest
k2 = function(t, times, h){
  return(gaussian_kernel(date_diff(t, times)/h))
}

#Distance from an hour to the hour of interest
k3 = function(t, times, h){
  return(gaussian_kernel(hour_diff(t, times)/h))
}

k = function(Xs,holder, h) {
  h_distance = 160000
  h_date = 50
  h_time=1*3600
  c_distance = 0.1
  c_date = 1
  c_time = 0.1

  k1_score = k1(as.numeric(holder[4:5]), Xs[, 4:5], h_distance)
  k2_score = k2(holder[9], Xs[,9], h_date)
  k3_score = k3(holder[10], Xs[,10], h_time)

  k_score = c_distance*k1_score+c_date*k2_score+c_time*k3_score

  guessed_temperature = sum(k_score*Xs[,11])/sum(k_score)
  actual_temperature = as.numeric(holder[11])
  #return(guessed_temperature - actual_temperature)
  return (guessed_temperature)
}

apply_kernels = function(train_data, validation_data, h) {
  holder = validation_data[1,]
  Ys = apply(validation_data, 1, function(Y, Xs_, h_) k(Xs_, Y, h_), Xs_ = train_data, h_ = h)
  return(Ys)
}

get_weight_plot = function() {

  plot(gaussian_kernel(matrix(seq(1,2000000,10000))/160000), xlim=range(0,100),
ylab="Weight", xlab="Distance [10km]", main="Station distance kernel")

  plot(gaussian_kernel(matrix(seq(1,365,1))/50), xlim=range(0,100), ylab="Weight",
xlab="Day diff", main="Date distance kernel")

  plot(gaussian_kernel(matrix(seq(0,24*3600,2*3600))/(1*3600)), ylab="Weight", xlab="Time
diff", main="Hour distance kernel")
}


n=dim(st)[1]
id=sample(1:n, floor(n*0.75))
train=st[id,]
test=st[-id,]
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",
          "23:00:00")
date <- "2013-04-20"
a <- 58.413497
b <- 15.582597
pred_data = st[1:11,]
for(i in 1:nrow(pred_data)){
  pred_data[i,4] = a
  pred_data[i,5] = b
  pred_data[i,9] = date
  pred_data[i,10] = times[i]
}

start.time <- Sys.time()

#res = apply_kernels(train,test,1)
#plot(res,test$air_temperature, xlim=range(-30,30),
#ylim=range(-30,30), xlab="Predicted temp", ylab="Actual temp", main="Actual vs predicted
temp", grid(6,6))
#abline(1,1)
res=apply_kernels(st,pred_data,0.1)
plot(res, main = date, ylab="Temperature")

end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
```