Lab 5 732A95

Anton Persson antpe404

12 december 2016

Lab 5

I implemented a kernel that is the sum och three gaussian kernels. They're supposed to account for the hourly difference between observations, the distance geographically from different weather station and the passage of time, i.e. the number of days passed between observations.

The code is presented below.

```
#Lab 5 732A95 Kernels

#Initializing data and smoothing and stuff.
stations<-read.csv("data/stations.csv", sep=",", header=T)
temps<-read.csv("data/temps50k.csv", sep=",", header=T)
library(geosphere)
set.seed(1234567890)
st <- merge(stations,temps,by="station_number")

h_distance <- 100000 # These three values are up to the students
  h_days <- 7#dessa tre är alltså smoothing factor för passage_of_tome
  h_time <- 2

a <- 58.4274 # The point to predict (up to the students)
b <- 14.826

pred_date <- as.Date("2010-12-24") # The date to predict (up to the students)
times <- c("02:00:00","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")
pred_time<-("12:00:00")
temp <- vector(length=length(times))
# Students' code here

#################################Kernel function###########################

gauss_kernel<-function(diff, h){
  svar<-exp(-((diff**2)/(h**2))) #h motsvarar 2sd^2
  return(svar)
}

#------------------------------The three different kernals------------------------------------
################Distance Kernel, to account for distances between stations############

longlat<-data.frame(cbind(longitude=st$longitude, latitude=st$latitude))
nkpg<-st[st$station_number==86340,]
nkpg_longlat<-c(nkpg$longitude[1], nkpg$latitude[1])

dist_func<-function(data=data.frame(cbind(longitude=st$longitude, latitude=st$latitude)), city_coord){
  distances<-distHaversine(p1 = data , p2 = city_coord)
```

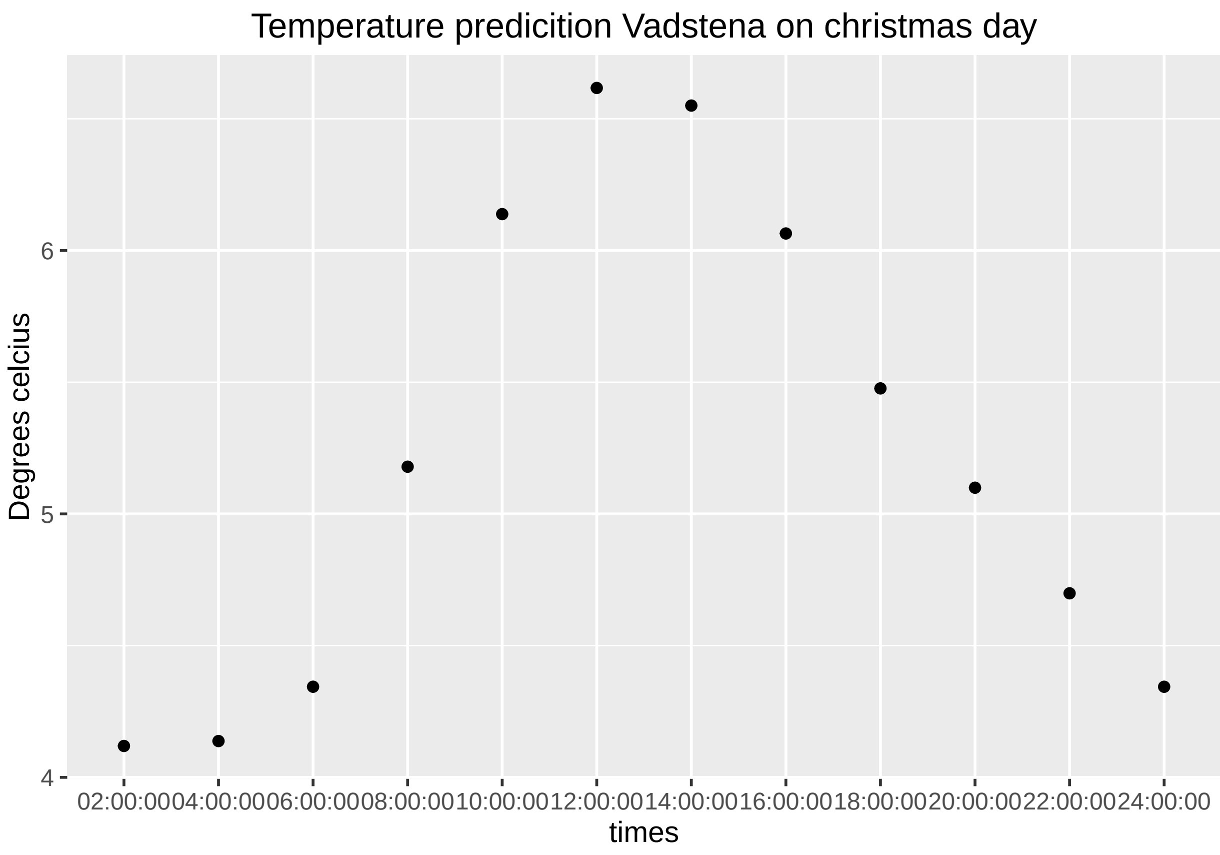
```r
  return(distances)
}

#dist_diff<-dist_func(city_coord = nkpg_longlat)

#######Days Kernel, to account for number of days between observations and day of interest########


time_func<-function(prediction_date, dates=as.Date(st$date)){
passage_of_time=as.numeric(difftime(time1 = prediction_date, time2=dates, units="days"))
#detta är antalet dagar fr alla mätningar till mitt datum
return(passage_of_time)
}

#time_diff<-time_func(prediction_date=pred_date)

#####Hour Kernel, to account for the number of hours between observations and hour of interest########

hour_func<-function(time_to_predict=pred_time, hours=substr((st$time), start=1, stop=8)){
hours_differential<-c(length(hours))
for (i in 1:length(hours)){
  klockslag<-as.difftime(c(hours[i], time_to_predict), format="%H:%M:%S", units="hours")
  if (abs(klockslag[1]-klockslag[2])>12){
    hours_differential[i]<-24-abs(klockslag[1]-klockslag[2])
  }
  else{
    hours_differential[i]<-abs(klockslag[1]-klockslag[2])
}
}
return(hours_differential)
}

#hours_diff<-hour_func()


#De ovanstående callsen nu, dvs hours_diff, time_diff och dist_diff
#är så att säga mina x-xnew i gau_kernel.
#Kallar dem diff i gauss_kernel nu.

###################################Övergripande funktion#####################################
#Denna använder alltså ovanstående funktioner för att dra ihop ett resultat.

totfunk<-function(data=st, h_time, h_days, h_distance, city_coord, pred_time, pred_date){
  #data<-st
  #city_coord<-nkpg_longlat
  hours<-substr((data$time), start=1, stop=8) #Defined to be able to cut the data
  dates<-as.Date(data$date) #Same here
  logic_date_vector<-paste(dates, hours)<paste(pred_date, pred_time)
  data<-data[logic_date_vector,]

  longlat<-data.frame(cbind(longitude=data$longitude, latitude=data$latitude))
  dates<-as.Date(data$date) #Redefining them here after the data subset
  hours<-substr((data$time), start=1, stop=8)
  dist_ker<-gauss_kernel(diff=dist_func(data=longlat, city_coord=city_coord),h=h_distance)
  time_ker<-gauss_kernel(diff=time_func(prediction_date=pred_date, dates=dates), h=h_days)
  hour_ker<-gauss_kernel(diff=hour_func(time_to_predict=pred_time, hours=hours), h=h_time)

  #Use a kernel that is the sum of three Gaussian kernels:
  predict<-(sum((dist_ker*data$air_temperature)+(time_ker*data$air_temperature)+
                (hour_ker*data$air_temperature)))/
    sum(hour_ker + time_ker + dist_ker)

  #Annars kan man också köra multiplikativt
  #predict<-sum((prod((dist_ker*data$air_temperature),
  #(time_ker*data$air_temperature),(hour_ker*data$air_temperature))))/
    #prod(sum(hour_ker),sum(time_ker),sum(dist_ker))
  return(predict)
}
```

To show that it works, I show the results of the kernel for predicting every second hour on christmas day this year, 2016. The coordinates used in this case are the one given in the template (a and b in the code above). They respond to Vadstena.

```r
library(ggplot2)
for (i in 1:length(times)){
  temp[i]<-totfunk(h_time=2, h_days=7, h_distance=100000, city_coord=c(b,a), pred_time=times[i],
               pred_date = as.Date("2016-12-24"))
}

hourly_predictions<-data.frame(cbind(times, temp=round(temp, 5)))

hourly_predictions2<-hourly_predictions
hourly_predictions2$times<-as.factor(hourly_predictions2$times)
hourly_predictions2$temp<-as.numeric(as.character(hourly_predictions2$temp))
hourly_predictions2
```

```
##       times    temp
## 1  02:00:00 4.11918
## 2  04:00:00 4.13785
## 3  06:00:00 4.34401
## 4  08:00:00 5.17929
## 5  10:00:00 6.13825
## 6  12:00:00 6.61707
## 7  14:00:00 6.55022
## 8  16:00:00 6.06468
## 9  18:00:00 5.47656
## 10 20:00:00 5.09953
## 11 22:00:00 4.69862
## 12 24:00:00 4.34390
```

```r
dec16<-ggplot(data=hourly_predictions2)+geom_point(aes(x=times, y=temp))+
  geom_line(aes(x=times, y = temp))+
  ggtitle("Temperature predicition Vadstena on christmas day")+
  ylab(label = "Degrees celcius")

dec16
```



The predictions show in the plot above seems to be a bit warm, even though it's not impossible. The variation within the day looks reasonable, with lower temperatures during the night.

In this case, I predict the temperature on 2016-12-24, i.e. a day far in to the future. That's many days from even the latest observation in my data, and very many days from most observations. I did look into the code to see what that does with my kernels. I found out that the fact that the day of interest is so far ahead in time does make my time kernel irrelevant. I show the sum of all three kernels individually when I run the prediction for Vadstena, 2016-12-24 at 12:00:00. The smoothing factors are the same as in the first code chunk, distance=100000, days=7 and time=2.

```
pred_date <- as.Date("2016-12-24") # The date to predict (up to the students)
pred_time<-("12:00:00")
   data<-st
  city_coord<-c(b,a)
  hours<-substr((data$time), start=1, stop=8) #Defined to be able to cut the data
  dates<-as.Date(data$date) #Same here
  logic_date_vector<-paste(dates, hours)<paste(pred_date, pred_time)
  data<-data[logic_date_vector,]

  longlat<-data.frame(cbind(longitude=data$longitude, latitude=data$latitude))
  dates<-as.Date(data$date) #Redefining them here after the data subset
  hours<-substr((data$time), start=1, stop=8)

  dist_ker<-gauss_kernel(diff=dist_func(data=longlat, city_coord=city_coord),h=h_distance)
  time_ker<-gauss_kernel(diff=time_func(prediction_date=pred_date, dates=dates), h=h_days)
  hour_ker<-gauss_kernel(diff=hour_func(time_to_predict=pred_time, hours=hours), h=h_time)
```

```r
  sums<-list(sum_of_distance_kernel=sum(dist_ker), sum_of_time_kernel=sum(time_ker),
             sum_of_hour_kernel=sum(hour_ker))
  sums
```

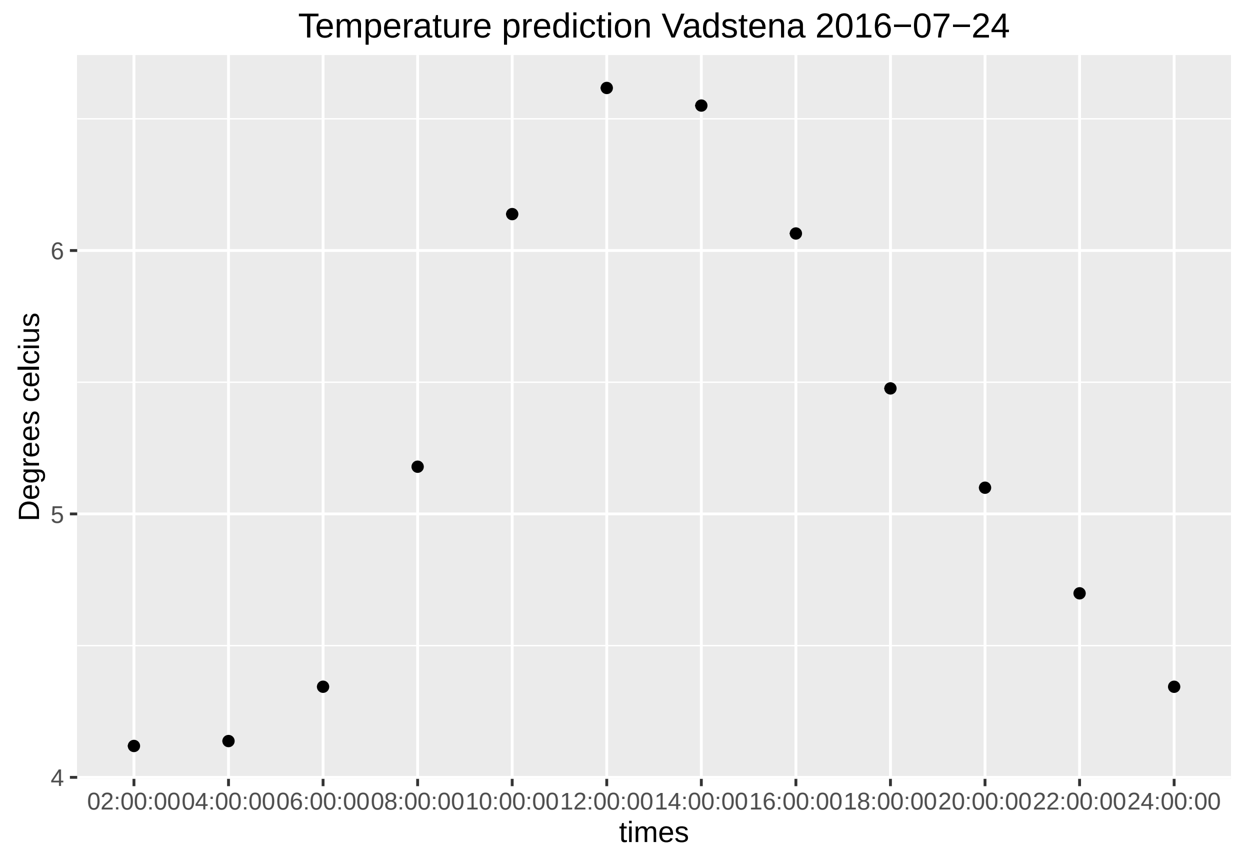
```
## $sum_of_distance_kernel
## [1] 3877.737
##
## $sum_of_time_kernel
## [1] 8.561185e-275
##
## $sum_of_hour_kernel
## [1] 8917.173
```

As you can see above, the time kernel that accounts for the number of days between the observations and 2016-12-24 in this case, sums to almost 0. That means that that variable is neglected practically. When I realised this, I decided to try to predict the 24th of July 2016 as well, to compare. That's showed below.

```r
for (i in 1:length(times)){
  temp[i]<-totfunk(h_time=2, h_days=7, h_distance=100000, city_coord=c(b,a), pred_time=times[i],
                   pred_date = as.Date("2016-07-24"))
}

hourly_predictions_summer<-data.frame(cbind(times, temp=round(temp, 5)))
hourly_predictions_summer2<-hourly_predictions_summer
hourly_predictions_summer2$times<-as.factor(hourly_predictions_summer$times)
hourly_predictions_summer2$temp<-as.numeric(as.character(hourly_predictions_summer$temp))
hourly_predictions_summer2
```

```
##       times    temp
## 1  02:00:00 4.11918
## 2  04:00:00 4.13785
## 3  06:00:00 4.34401
## 4  08:00:00 5.17929
## 5  10:00:00 6.13825
## 6  12:00:00 6.61707
## 7  14:00:00 6.55022
## 8  16:00:00 6.06468
## 9  18:00:00 5.47656
## 10 20:00:00 5.09953
## 11 22:00:00 4.69862
## 12 24:00:00 4.34390
```

```r
juli16<-ggplot(data=hourly_predictions_summer2)+geom_point(aes(x=times, y=temp))+
  geom_line(aes(x=times, y = temp))+
  ggtitle("Temperature prediction Vadstena 2016-07-24")+
  ylab(label = "Degrees celcius")

juli16
```



As you can see, it looks like exactly the same as the predicition for 2016-12-24. That is because the time kernel gets totally neglected. Thus, the only thing that matters for the prediction is the geographical distance and time within the days, i.e. the hours. Then it's natural that the predictions get's the same for 2016-12-24 and 2016-07-24, since these two variables are the same for those two dates. This is because I try to predict so far ahead in time in this case. This could probably be solved by making that kernel more like the hour kernel. In the hour kernel, I compute the number of hours as a difference my hour of interest, which makes the maximum difference 12. If I'd do the same with time, i.e. make the variable a difference between weeks for example, it would be a variable that takes the season in account. Thus, instead of saying all observations gets 0 weight in this case since it's so many days passed compared to all observed temperatures, the observations made in week 50 gets most weight when I predict 2016-12-24. The observations in week 24 would thus get least weight. That would probably be more appropriate, since it makes the assumption that a day in week 50 will be more similar to week 50 another year. The only downside would be that I lose the general trend of temperature, but I don't even know if there is one.

The other two variables definitely seems reasonable, and the variable discussed above is also somewhat reasonable if I predict a day with other observations measured in fewer days. I don't know how to show that my kernels gives more weight to closer points in a good way. I could print like twenty observations from the dist/time/hour_ker, which is the vectors with the weight, but that would say so much. Instead I will just assure that I've checked them manually and they work, the hours is the diffenrential from my predicted hour clockwise, that is at most 12 hours differential. The distance seems to work, I compared Norrkoping to some different cities like Linkoping and Kiruna and the distances makes sense.

Of course, I could've changed the smoothing factor even more (it's already set at 100000) to make the passage of time (number of days) kernel more relevant, but this is just to make an example.

Note that all the code I've used for this lab is presented in the chunks above, why I don't have them as redundant code in an appendix.