Computer lab 3 block 1

Laura Julià Melis

12/17/2019

Assignment 1. Kernel Methods.

We have defined that the latitude and longitud of the point of interest is $(55.90000, 12.71660)$ and the date of interest is "2004-05-28". As said in the exercise, our times of interest, for which we will make predictions, are: "04:00:00", "06:00:00", ... , "22:00:00", "24:00:00".

Now, the first step is to compute the differences from the data points, dates and times with the point, date and times of interest (*See code in the Appendix session to see how these are calculated*).

Choose an appropriate smoothing coefficient or width for each of the three kernels above. Answer to the following questions:

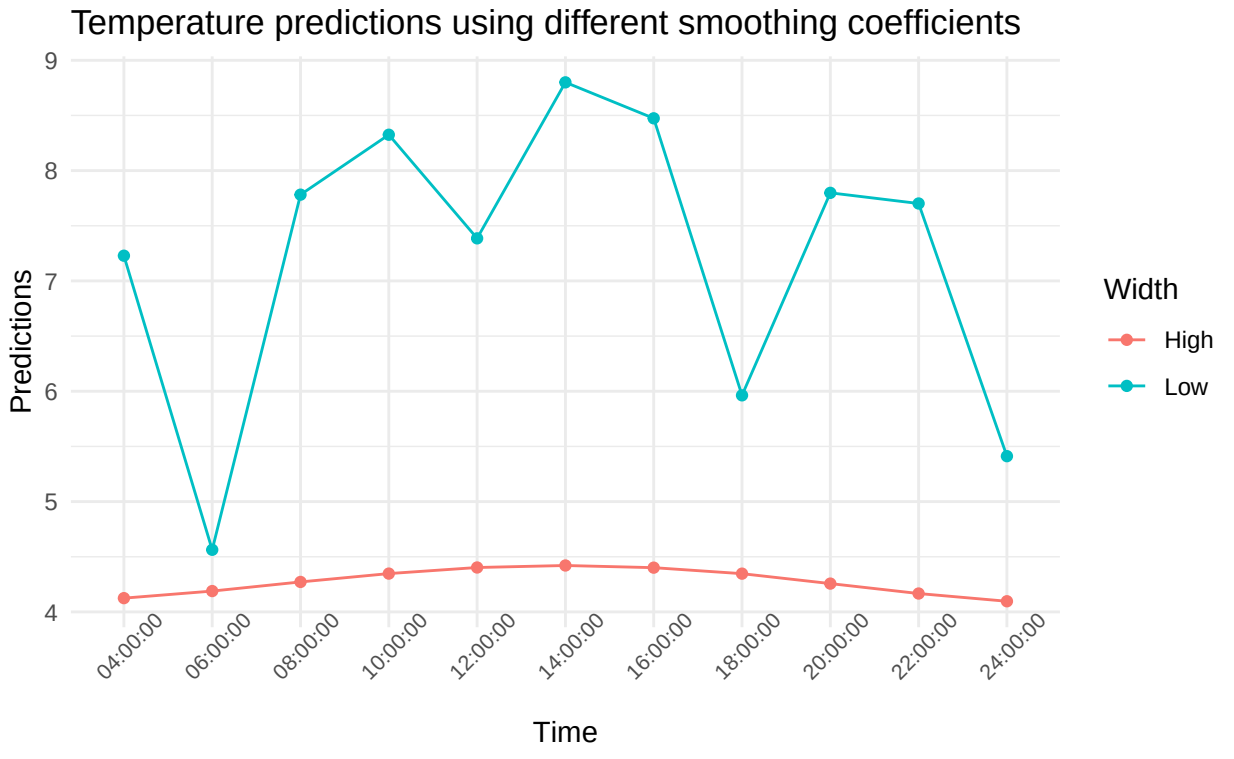
a) Show that your choice for the kernels' width is sensible, i.e. that it gives more weight to closer points. Discuss why your of definition of closeness is reasonable.

To show that the kernels' width is sensible, we will choose low and high widths for each of the kernels. The chosen low widths are: $h_1 = 30000$, $h_2 = 10$,$h_3 = 0.5$ and the high widths: $h_1 = 1000000$, $h_2 = 300$,$h_3 = 11$.

With these smoothing coefficients we have calculated the gaussian kernels as follows:

$$k_i(x, x') = exp\left(- \left(\frac{||x - x'||}{h_i} \right)^2 \right)$$

Finally we have combined the three kernels by summing them up. We will plot now the predicted temperatures for the same point date and times obtained with each kernel:



From the plot above we can observe that when the kernel widths are low (30km, 10 days and half an hour, for the distance kernel, the date kernel and the time kernel, respectively) there are a lot of variations in the values (the result is very noisy).

On the other hand, when the widths are high (1000km, 300 days and 11 hours) the line is really smooth and the values are smaller.

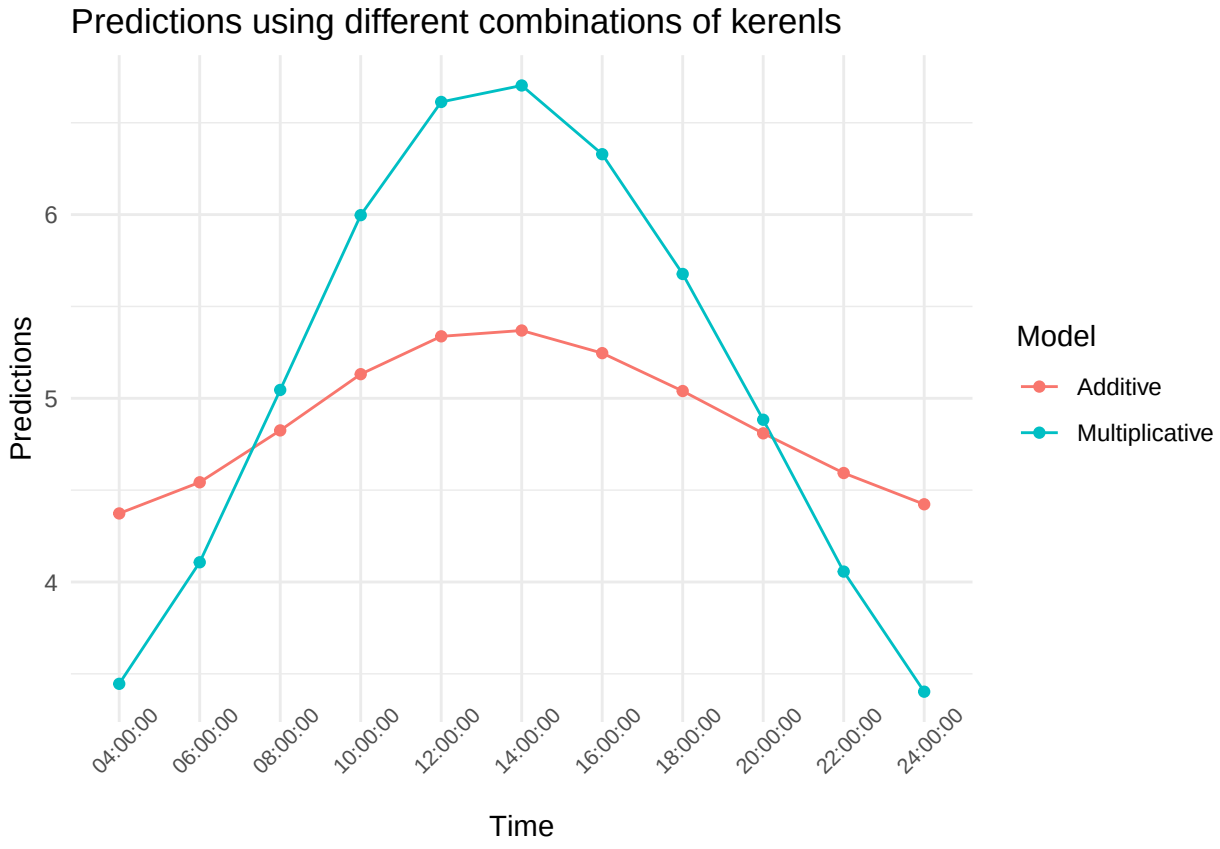
From the theory, "the best density model is obtained for some intermediate value of h".

b) Instead of combining the three kernels into one by summing them up, multiply them. Compare the results obtained in both cases and elaborate on why they may differ.

In this quetion we want to compare models where the combination of the kernels is done by summing or multiplying them. So, for both cases we will choose widths that we think that are reasonable for the nature of the variables and our problem:

For the kernel of the distance, we will set a width of 500km because we think it is a distance from which the temperature can be different. For the kernel of the date, we will choose a width of 100 days, and for the kernel of the time, we will set a smoothing parameter of 5 hours. Temperature could change in a period of 5 hours (day and night, for example) and in a period of 100 days (in approximately three months the season might change and thus, also the temperatures).

To compare the results for the additive and the multiplicative kernel combinations, we will plot the predictions obtained in each case:



In both cases the curve is quite smooth but the predicted values obtained by using the multiplicative combination of the kernels offers higher predictions than the additive model.

Assignment 2. Support Vector Machines.

Use the function ksvm from the R package kernlab to learn a SVM for classifying the spam dataset that is included with the package. Consider the radial basis function kernel (also known as Gaussian) with a width of 0.05. For the C parameter, consider values 0.5, 1 and 5. This implies that you have to consider three models.

a) Perform model selection, i.e. select the most promising of the three models (use any method of your choice except cross-validation or nested cross-validation).

The SVM model with "rbfdot" (Radial Basis kernel "Gaussian") has been performed with the function `ksvm()` from the package **kernlab**. We have set in the *kpar* argument that the width for the Radial Basis kernel that we want to use is 0.05: `kpar=list(sigma=0.05)`. We will fit three models with this characteristics, each one with a different cost of constraints violation $C = 0.5, 1, 5$.

To method used for the model selection will be the holdout method so the spam dataset will be separated into three sets: training validation and test, with 0.5, 0.25 and 0.25 of the original observations (chosen randomly), respectively. Then, the models will be fitted using the training data and the predictions will be performed for the validation dataset.

The best model will be chosen based on the validation set by calculating the misclassification errors.

The confusion matrices for the test data for every model are:

```
## Model with C=0.5
```

```
##          pred_svm1
## y_valid   nonspam spam
##   nonspam     690   13
##   spam        276  171
```

```
## Model with C=1
```

```
##          pred_svm2
## y_valid   nonspam spam
##   nonspam     634   69
##   spam        189  258
```

```
## Model with C=5
```

```
##          pred_svm3
## y_valid   nonspam spam
##   nonspam     593  110
##   spam         84  363
```

Finally, we will show a table with the test misclassification errors obtained in each model:

```
##            Model Misclassification.errors
## 1 SVM with C=0.5               0.2513043
## 2   SVM with C=1               0.2243478
## 3   SVM with C=5               0.1686957
```

The model using C = 5 is the one that offers the lowest error. This means that this model predicts better than the other two. For this reason, this is the model that will be selected as the optimal.

However, its worth mentioning that, although only 16.87% of the observations are classified wrong when using the third model, the number of nonspam mails classified as spam in this model is greater than in the others.

b) Estimate the generalization error of the SVM selected above (use any method of your choice except cross-validation or nested cross-validation). We have computed the predictions and misclassification errors on the test dataset, which is unseen data for the model. So now we can use "train" and "valid" datasets to train the model.

Let's see the results:

```
## The confussion matrix is:
```

```
##          pred_svm
##           nonspam spam
##   nonspam     614   60
##   spam        158  319
```

```
## The  generalization error is:
```

```
## [1] 0.1894005
```

So, the final model predicts wrongly around 19% of the observations. Also, from the confussion matrix we can observe that the number of nonspam mails classified as spam is considerably smaller compared to the number of spam mails classified as nonspam.

c) Produce the SVM that will be returned to the user, i.e. show the code.

The final model is performed by using the following code:

```
svm_fit <- ksvm(as.matrix(new_train[,-58]), new_train[,58], scale=FALSE,
                kernel="rbfdot", kpar=list(sigma=0.05), C=5)
```

d) What is the purpose of the parameter C ?

The parameter C is the cost of constraints violation, it tells the SVM optimization how much we want to avoid misclassifying. It is useful to controll the outliers: if we define a low C we will allow more outliers, and for a high C fewer outliers will be allowed.

Appendix.

Assignment 1.

Distances

```
# 1.1. Importing the data:
RNGversion('3.5.1')
set.seed(1234567890)
library(geosphere)

stations <- read.csv("stations.csv", sep=",", header=TRUE, stringsAsFactors=FALSE,
                     fileEncoding="latin1")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")

# 1.2. Computing Gaussian kernels:
a <- 55.90000 # latitude of interest
b <- 12.71660 # longitude of interest
date <- "2004-05-28" # The date to predict
times <- as.data.frame(as.factor(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")))
colnames(times) <- "time_of_interest"
times$time_ind <- row.names(times)
st <- merge.data.frame(st,times,all=TRUE)

# Filtering our date (removind observations posteriors to our date)
filter_date <- ifelse(as.character(st$date) < date, TRUE, FALSE)
st <- st[which(filter_date==TRUE),]

### A. Distance from a station to the point of interest.
point_interest <- c(a,b)
st$dif_distance <- distHaversine(p1=point_interest, p2=st[,c(4,5)])

### B. Distance between the day a temp measurement was made and the day of interest.
st$dif_date <- abs(as.integer(difftime(date, st$date, units = "days")))
st$dif_date <- st$dif_date %% 365

### C. Distance between the hour of the day a temp measurement was made and the hour of interest.
st$dif_time <- abs(as.numeric(difftime(strptime(st$time_of_interest,"%H:%M:%S"),
                                       strptime(st$time,"%H:%M:%S") ,units = "hours")))
st$dif_time <-ifelse(st$dif_time <= 12 , st$dif_time , (24-st$dif_time))
```

Question a

```
# New data frame for this question:
st_1 <- st

# Setting low and high smoothing factors:
low_h1 <- 30000
low_h2 <- 10
low_h3 <- 0.5

high_h1 <- 1000000
high_h2 <- 300
high_h3 <- 11
```

```r
# Calculating guassian kernels
st_1$h_distance_low <- exp(-(st_1$dif_distance/low_h1)^2)
st_1$h_date_low <- exp(-(st_1$dif_date/low_h2)^2)
st_1$h_time_low <- exp(-(st_1$dif_time/low_h3)^2)

st_1$h_distance_high <- exp(-(st_1$dif_distance/high_h1)^2)
st_1$h_date_high <- exp(-(st_1$dif_date/high_h2)^2)
st_1$h_time_high <- exp(-(st_1$dif_time/high_h3)^2)

# Kernels combinations: summing and multiplying them up.
st_1$kernel_sum_low <- (st_1$h_distance_low + st_1$h_date_low + st_1$h_time_low)
st_1$kernel_sum_high <- (st_1$h_distance_high + st_1$h_date_high + st_1$h_time_high)

# Predicted temperatures:
result<- NULL
for(i in 1:11){
  temp <- st_1[st_1$time_ind==i,]
  pred_sum_low <- sum(temp$kernel_sum_low*temp$air_temperature)/sum(temp$kernel_sum_low)
  pred_sum_high <- sum(temp$kernel_sum_high*temp$air_temperature)/sum(temp$kernel_sum_high)

  temp <- cbind(i, pred_sum_low, pred_sum_high)
  result <- rbind(result,temp)
}
result <- as.data.frame(result)
result <- merge(x =result, y = times, by.x = "i", by.y = "time_ind", all.x = TRUE)
result$pred_sum <- as.numeric(as.character(result$pred_sum_low))
result$pred_prod <- as.numeric(as.character(result$pred_sum_high))

# Plot of predicted temperatures for each kernel:
library(ggplot2)

df <- data.frame("Time"= result$time_of_interest ,
                 "Predictions"= c(result$pred_sum_low, result$pred_sum_high),
                 "Width"= rep(c("Low", "High"), each=11))

ggplot(data=df, aes(x=Time, y=Predictions, group=Width)) + geom_line(aes(color=Width)) +
  geom_point(aes(color=Width)) +
  ggtitle("Temperature predictions using different smoothing coefficients") +
  theme_minimal() + theme(axis.text.x =element_text(size = rel(0.9), angle = 45))
```

Question b

```r
# The parameter h is called smoothing factor or width.
h1 <- 500000 # h1: width for kernel 1
h2 <- 100 # h2: width for kernel 2
h3 <- 5 # h3: width for kernel 3

# Calculating guassian kernels
st$h_distance <- exp(-(st$dif_distance/h1)^2)
st$h_date <- exp(-(st$dif_date/h2)^2)
st$h_time <- exp(-(st$dif_time/h3)^2)

# Kernels combinations: summing and multiplying them up.
st$kernel_sum <- (st$h_distance + st$h_date + st$h_time)
```

```r
st$kernel_prod <- (st$h_distance * st$h_date *st$h_time)

# Predicted temperatures:
result<- NULL
for(i in 1:11){
  temp <- st[st$time_ind==i,]
  pred_sum <- sum(temp$kernel_sum*temp$air_temperature)/sum(temp$kernel_sum)
  pred_prod <- sum(temp$kernel_prod*temp$air_temperature)/sum(temp$kernel_prod)

  temp <- cbind(i, pred_sum, pred_prod)
  result <- rbind(result,temp)
}
result <- as.data.frame(result)
result <- merge(x =result, y = times, by.x = "i", by.y = "time_ind", all.x = TRUE)
result$pred_sum <- as.numeric(as.character(result$pred_sum))
result$pred_prod <- as.numeric(as.character(result$pred_prod))

# Plot of predicted temperatures for each kernel:
library(ggplot2)

df <- data.frame("Time"= result$time_of_interest ,
                 "Predictions"= c(result$pred_sum, result$pred_prod),
                 "Model"= rep(c("Additive", "Multiplicative"), each=11))

ggplot(data=df, aes(x=Time, y=Predictions, group=Model)) + geom_line(aes(color=Model))
+ geom_point(aes(color=Model)) + ggtitle("Predictions using different combinations of kerenls")
+ theme_minimal() + theme(axis.text.x =element_text(size = rel(0.9), angle = 45))
```

Assignment 2.

Question a

```r
# 2.0. Dataset:
library(kernlab)
data(spam) # spam[,58] or spam$type is the target variable

# Divide data into training, validation and test sets:
n=dim(spam)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=spam[id,]

id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=spam[id2,]

id3=setdiff(id1,id2)
test=spam[id3,]


x_train <- as.matrix(train[,-58])
y_train <- train[,58]
```

```r
x_valid <- as.matrix(valid[,-58])
y_valid <- valid[,58]

svm_fit1 <- ksvm(x_train, y_train, scaled=FALSE, kernel="rbfdot", kpar=list(sigma=0.05), C=0.5)
svm_fit2 <- ksvm(x_train, y_train, scaled=FALSE, kernel="rbfdot", kpar=list(sigma=0.05), C=1)
svm_fit3 <- ksvm(x_train, y_train, scaled=FALSE, kernel="rbfdot", kpar=list(sigma=0.05), C=5)

pred_svm1 <- predict(svm_fit1, x_valid)
pred_svm2 <- predict(svm_fit2, x_valid)
pred_svm3 <- predict(svm_fit3, x_valid)

# Confussion matrices:
cat("Model with C=0.5\n")
table(y_valid, pred_svm1)
cat("Model with C=1\n")
table(y_valid, pred_svm2)
cat("Model with C=5\n")
table(y_valid, pred_svm3)

# Table of results:
data.frame("Model"=c("SVM with C=0.5","SVM with C=1","SVM with C=5"),
           "Misclassification errors"=c(mean(pred_svm1 != y_valid),
                                        mean(pred_svm2 != y_valid),mean(pred_svm3 != y_valid)))
```

The confusion matrices for the test data for every model are:

```
## Model with C=0.5
```

```
##          pred_svm1
## y_valid   nonspam spam
##   nonspam     690   13
##   spam        276  171
```

```
## Model with C=1
```

```
##          pred_svm2
## y_valid   nonspam spam
##   nonspam     634   69
##   spam        189  258
```

```
## Model with C=5
```

```
##          pred_svm3
## y_valid   nonspam spam
##   nonspam     593  110
##   spam         84  363
```

Finally, we will show a table with the test misclassification errors obtained in each model:

```
##            Model Misclassification.errors
## 1 SVM with C=0.5                0.2513043
## 2   SVM with C=1                0.2243478
## 3   SVM with C=5                0.1686957
```

Question b

```r
new_train <- rbind(train, valid)
svm_fit <- ksvm(as.matrix(new_train[,-58]), new_train[,58], scale=FALSE,
                kernel="rbfdot", kpar=list(sigma=0.05), C=5)
```

```
pred_svm <- predict(svm_fit,  test[,-58])

cat("The confussion matrix is:\n")
table(test[,58], pred_svm)
cat("The  generalization error is:\n")
mean(pred_svm != test[,58])
```

Question c

```
svm_fit <- ksvm(as.matrix(new_train[,-58]), new_train[,58], scale=FALSE,
                kernel="rbfdot", kpar=list(sigma=0.05), C=5)
```