Computer lab 3 block 1

Biswas Kumar, bisku859

12/31/2019

Assignment 1. Implement a kernel method to predict the hourly temperatures for a date and place in Sweden.

To do so, you are provided with the files stations.csv and temps50k.csv. These files contain information about weather stations and temperature measurements in the stations at different days and times. The data have been kindly provided by the Swedish Meteorological and Hydrological Institute (SMHI).You are asked to provide a temperature forecast for a date and place in Sweden. The forecast should consist of the predicted temperatures from 4 am to 24 pm in an interval of 2 hours. Use a kernel that is the sum of three Gaussian kernels:

1. 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.

Explained in step no#1- step#3 below.

step 1 . The first to account for the distance from a station to the point of interest.

smoothing coefficient (distance kernel) : The distance kernel has been set at 50,000 m (or 50 K.m).It has been considered at 50 Km because its the ideal distance from which the weather conditions will change noticebly. Also, as these data sets are for Sweden, where climatic condition differs north-south and also along the coast , 50 k.m seems to be ideal distance kernel under given conmditions.

step 2. The second to account for the distance between the day a temperature measurement was made and the day of interest.

smoothing coefficient (date kernel) :The month of each year changes significantly and moreover if its the first part or the last part of month.To capture effect wisely, I understand that 1 week (7 days)should be ideal timeframe to be considered as date kernel for our purpose.

step 3. The third to account for the distance between the hour of the day a temperature measurement was made and the hour of interest.

smoothing coefficient (time kernel) :The day is divided in 24 hours that see huge variation in day time and night time values. To capture variations appropriately, I prefer to have it divided into 6 parts. hence. the time kernel should be 4.

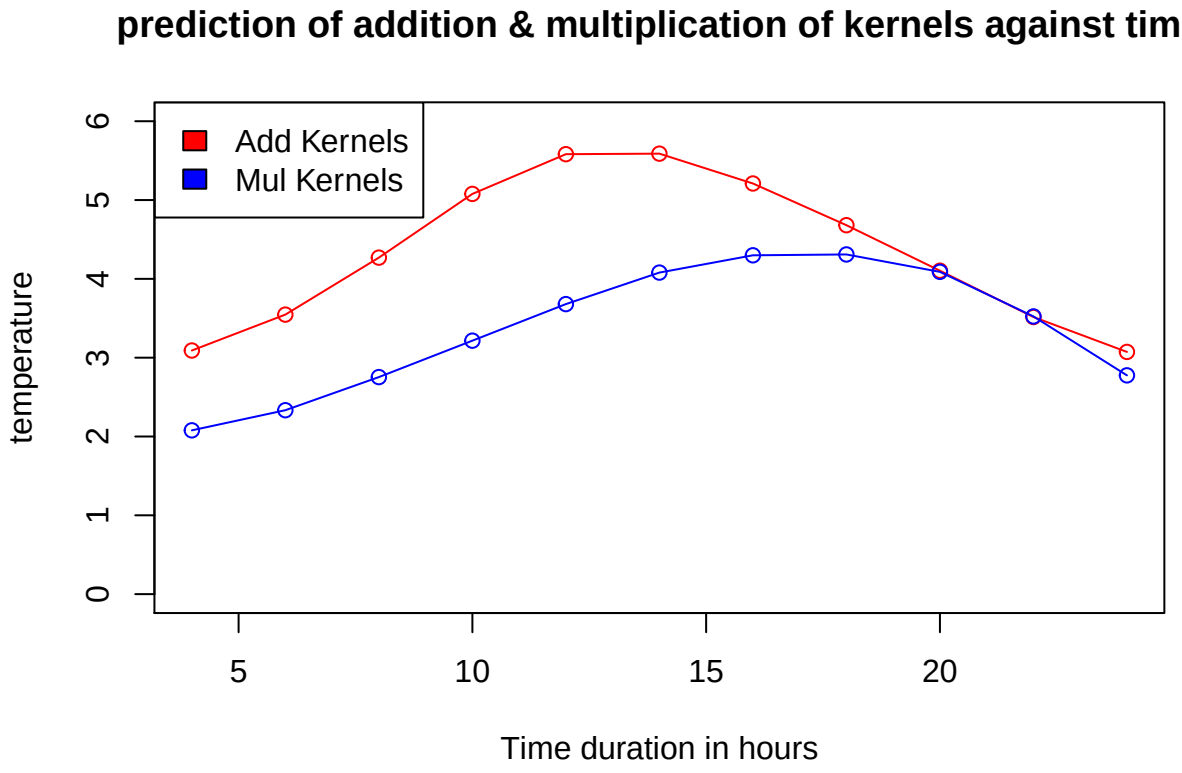
Choose an appropriate smoothing coefficient or width for each of the three kernels above.

responded above under each segment

Solution

Step 1 : Importing excel document influenza.xlsx

```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```



2. 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.

From the plot it can be visualized that the additive kernel curve follows a smooth curve pattern as against multiplication kernels. Additive models gives consistent forecast becuse of irrelevant kernels gets added in the prediction.On the other hand,the multiplication kernel has has non smooth curve because it filter out all irrelevant kernels /least influenced ones. It is therefore, though multiplication kernels are irregular, it can give better prediction.

Assignment 2. Support vector machine

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. 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).

1. Estimate the generalization error of the SVM selected above (use any method of your choice except cross-validation or nested cross-validation).

Mentioned below

```
## [1] 4601   58
```

```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```

```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```

```
##   For C = 0.5
##  Misclasification rate  0.08608696
## ker_predict nonspam spam
##     nonspam     681   77
##     spam         22  370
##   For C = 1
##  Misclasification rate  0.06956522
## ker_predict nonspam spam
##     nonspam     680   57
##     spam         23  390
##   For C = 5
##  Misclasification rate  0.07478261
## ker_predict nonspam spam
##     nonspam     676   59
##     spam         27  388
```

Using the output above, we can see that we get the minimum misclassification rate when C=1.

Hence, model with C=1 is ideal choice using SVM method.

(Plesae note that my last submission highlighted that C=5 was the best choice with least misclassification. Now,as per seminar discussions, I have divided the data sets into 3 parts ,the result is now different and the minimum misclassification is when C=1)

```
##  Misclasification rate  0.08253692
```

```
##
## ker_predict nonspam spam
##     nonspam     641   62
##     spam         33  415
```

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

As mentioned ahead

```
#using KSVM to check for models & predictions
        ker<-ksvm(x=as.matrix(spam[,-58]),y=spam[,58],scaled = TRUE,C=1,kernel="rbfdot",kpar = list(sig
print(ker)
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.05
##
## Number of Support Vectors : 1655
##
## Objective Function Value : -802.313
## Training error : 0.039774
```

The SVM model and probabilistic parameters are highlighted above which shall be given to the user.

3. What is the purpose of the parameter C ?

It is important to state that C is a regularization parameter that controls the bias-variance trade off between the achieving a low training error and a low testing error.So larger the value of C will help model to be more flexible with fewer misclassification, which means more variance and less bias.

Code Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(geosphere)
library(kernlab)
RNGversion('3.5.1')
set.seed(1234567890)
stations <- read.csv("stations.csv")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")
#check the stations on plot below
# plot(stations[,4:5])

a <- 58.4274 # The point to predict is considered as given in exercise
b <- 14.826
h_date <- 7 #my date
h_distance<- 50000 # my distance in meter
h_time <- 4# my time
date <- "2013-06-14" # The date to predict (up to the students)

# the file temps50k.csv may contain temperature measurements that are posterior to the day and hour of
p<-which(as.Date(st$date) < as.Date(date))
st<-st[p,] # selecting the filtered rows and saving back to st for further process
times <- c("04:00:00", "06:00:00","8:00:00","10:00:00","12:00:00","14:00:00","16:00:00","18:00:00","20:
```

```
temp <- vector(length=length(times))

#First kernel - to account for the distance from a station to the point of interest
ll<-vector(length=length(st$station_number))
for (i in 1:length(st$station_number))
      ll[i]<-distHaversine(c(b,a),st[i,c("longitude","latitude")])
dist_kernel<- exp(-(ll/h_distance)^2) # Gausian Kernel


# Second kernel- to account for the distance between the day a temperature measurement was made and the

dd<-vector(length=length(st$station_number))
for (k in 1:length(st$station_number)){
date_strings = c("2013-11-04",as.character(st[k,c("date")]))
datetimes = strptime(date_strings, format = "%Y-%m-%d")
dd[k] = abs(difftime(datetimes[2], datetimes[1], units = "days")) # days
}
d= dd %% 365

nearest_d=vector(length=length(st$station_number))

for (k in 1:length(st$station_number)){
nearest_d[k]<-abs(min(d[k],365-d[k]))
}
date_kernel<- exp(-(nearest_d/h_date)^2) # Gausian Kernel

# The third - to account for the distance between the hour of the day a temperature measurement was mad

tt<-vector(length=length(st$station_number))
y<- matrix(nrow=length(st$station_number),ncol=11) # total limit hours
for (i in 1:length(times)){
for (k in 1:length(st$station_number)){
hours_strings = c(as.character(times[i]),as.character(st[k,c("time")]))
hourtimes = strptime(hours_strings, format = "%H:%M:%S")
tt[k] = abs(difftime(hourtimes[2], hourtimes[1], units = "hours")) # hours
}

for (k in 1:length(st$station_number)){
if(tt[k] > 12){
     tt[k]<- 24-tt[k]
}
}
 y[,i]<- tt
}
i=1
predict_add<-vector(length = 11)# 11 nos. of time based prediction interval of 2 hrs each starting 4 am
predict_mul<-vector(length = 11)
while(i < 12){
 time_kernel<- exp(-(y[,i]/h_time)^2) # Gausian Kernel
 # addition of all kernel
 add_kernel<- dist_kernel+date_kernel+time_kernel

 mul_kernel<- dist_kernel * date_kernel * time_kernel
```

```
 # Now the above two kernels i.e add and mul kernel need to be multiplied with air temperature for pred
 add_temp <-sum(add_kernel*st$air_temperature)
 mul_temp <-sum(mul_kernel*st$air_temperature)
 predict_add[i] <- add_temp/sum(add_kernel)  # to predict addition of kernel against temperature
 predict_mul[i] <- mul_temp/sum(mul_kernel)  # to predict multiplication of kernel agaist temperature
  i=i+1
}

time_vector= seq(4,24,2) # creting a vector for time plotting on x axis

#plotting the kernels predictions against time
plot(x=time_vector,y=predict_add, type="o",xlab ="Time duration in hours",ylab="temperature",main="predi
lines(x=time_vector,predict_mul, type="o",ylab="multiplication of kernels",col = "blue")# for multiplic
legend('topleft',legend=c("Add Kernels","Mul Kernels"),fill=c("red","blue")) # plotting legends on char


# Kernel method on train data
#training error as below
library(kernlab)
data(spam)
# Dividing data into 3 categories (Train- 50% , Validation- 25%  ,Test- Balance 25% ) as a part of corr
# The split of data is performed as highlighted in lecture 1 block 1 (slide)
n=dim(spam)[1] # dimension of spam and no.of rows
dim(spam)
# Train sample classification
RNGversion('3.5.1')
set.seed(12345)
ind=sample(1:n, floor(n*0.5)) # using sample and floor function to extract 50% of spam (data)
train=spam[ind,]
ind1=setdiff(1:n, ind)
# Validation sample classification
RNGversion('3.5.1')
set.seed(12345)
ind2=sample(ind1, floor(n*0.25)) # using sample and floor function to extract 25% of spam (data)
valid=spam[ind2,]
# Test sample classification
ind3=setdiff(ind1,ind2)
test=spam[ind3,]

# 1. Training the model

#for saving C value & corresponding Misclassification rate we build null vectors and then substitute th
cvalue<- c(0,0,0)
Misclassification_error <- c(0,0,0)
#initializing k=1, for saving output from loop for Cvalues and Misclassification rate
k=1
a<-c(0.5,1,5)
for(i in a){
        #using KSVM to check for models & predictions
        ker<-ksvm(x=as.matrix(train[,-58]),y=train[,58],scaled = TRUE,C=i,kernel="rbfdot",kpar = list(si
cat( "\t For C =", i,"\n")
# predicting inside loop to get confusion matrix & misclassification error
ker_predict <- predict(ker,newdata = valid[,-58])
```

```
#ker_predict
plan=table(ker_predict,valid$type);plan
#True Positive, True Negative, False Positive & False Negative calculation
TrueP=plan[2,2];TrueN=plan[1,1];FalseN=plan[2,1];FalseP=plan[1,2]
cat("\tMisclasification rate ",(FalseP + FalseN)/ sum(plan))
Misclassification_error[k] <- (FalseP + FalseN)/ sum(plan)
print(plan)
k=k+1
}

# Trainig the data sets ( Train & Valid) combiningly and then predicting over test data set to check re

train_valid<- rbind(train,valid) # using r bind to adding row-wise
# ksvm model to train the data
ker<-ksvm(x=as.matrix(train_valid[,-58]),y=train_valid[,58],scaled = TRUE,C=i,kernel="rbfdot",kpar = lis

#prediction of outcome of Ksvm model on test data
ker_predict <- predict(ker,newdata = test[,-58])
# confusion matrix
plan=table(ker_predict,test$type)
#True Positive, True Negative, False Positive & False Negative calculation
TrueP=plan[2,2];TrueN=plan[1,1];FalseN=plan[2,1];FalseP=plan[1,2]
cat("\tMisclasification rate ",(FalseP + FalseN)/ sum(plan))
print(plan)

#using KSVM to check for models & predictions
        ker<-ksvm(x=as.matrix(spam[,-58]),y=spam[,58],scaled = TRUE,C=1,kernel="rbfdot",kpar = list(sig
print(ker)
```