Machine Learning Lab3 Block1

Combined by Fengjuan Chen

Fengjuan Chen, Jooyoung Lee, Weng Hang Wong

Good report, not much to complain about :)

2019 12 17

Assignment 1 - *Fengjuan Chen and Jooyoung Lee*

Table 1: different distances with different weights

locationdist	h_10km
0e+00	1.0000000
1e+02	0.9950125
1e+03	0.9512294
1e+04	0.6065307
5e+04	0.0820850
1e+05	0.0067379
2e+05	0.0000454
5e+05	0.0000000
1e+06	0.0000000

Nice overview.

Table 2: different distances with different weights

datedist	h180
0	1.0000000
1	0.9672161
3	0.9048374
7	0.7918896
30	0.3678794
60	0.1353353
180	0.0024788
300	0.0000454
500	0.0000001

Table 3: different distances with different weights

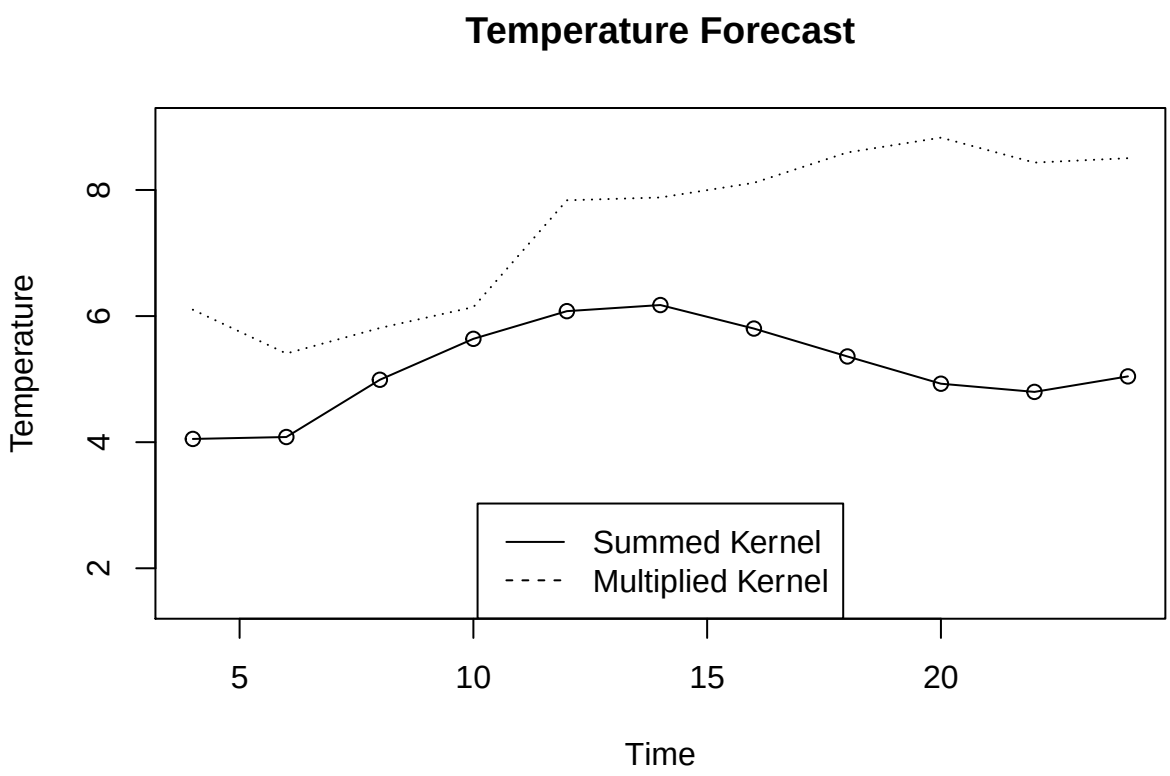
timedist	h120
0	1.0000000
40	0.7165313
60	0.6065307
90	0.4723666
120	0.3678794
150	0.2865048
200	0.1888756
250	0.1245145
300	0.0820850

timedist	h120
500	0.0155039

choice for kernels' width

Since we use three Gaussian kernels, the kernels' width h can adjust the weight of each observation according to the distance between them. When h increases, the weight of those observations near the object to be predicted will reduce. When h decrease, the weights of closer observations will increase.

We choose h_distance=200000 (20km), h_date=30 (1 month) and h_time=2 (120 minutes). We use some concrete distances of these three types to show that closer points have more weights.



compare the results of two kernel functions

The result of the sum of three kernels is near 4 to 6 degrees, while the result of the multiplication of three kernels is near 6 to 9 degrees.

By search with the original data, we find there are only four observations on the date "2013-11-03", which is the nearest points to the predicted date "2013-11-04". And the temperatures are from -3.9 to 5.9 degrees. Similarly, we search for the location "58.4274, 14.826", and get 10 observations at that position. Their temperatures are from -5.8 to 18 degrees.

When using sum of the three kernels, each kernel who has higher weight will play an heavy role for the final decision. However, when we adopt the multiplication of three kernels, big single kernel value will be eliminated by other very small kernel values. At the same time, if one kernel value is extremely small, for

Good!

example near zero, the final kernel value will be very small, such as near zero. That means the three Gaussian kernels of one point must all have high values to make this point important in the decision when the final kernel function is the multiplication of three kernel functions.

Assignment 2 *Weng Hang Wong*

1 All fine.

First we split the data into 3 parts, which are training(50%), test(25%) and valid(25%) for the holdout method. Now we have the most promising model for the valid error which is the model 2 with C=1, although the error rate is the lowest of the training set model is model 3 wich C=5, but we should choose the the model with the lowest error rate of valid set.

```
##     C train_error
## 1 0.5  0.04304348
## 2 1.0  0.03565217
## 3 5.0  0.02173913
```

```
##     C train_error valid_error
## 1 0.5  0.04304348  0.09478261
## 2 1.0  0.03565217  0.08782609
## 3 5.0  0.02173913  0.08956522
```

2

We estimate generalization error by using test data, with the hyperparameter C=1 which we have calculated from above. We can see the test error rate as below.

```
## The test error rate:
```

```
## [1] 0.08861859
```

3

```
# Produce SVM
svm_model <- ksvm(type~., data=spam, C=1,kernel="rbfdot" ,
                  type="C-svc", kpar=list(sigma=h), cross=0)
```

4.

What's the purpose of the parameter C?

The purpose of the parameter C is to reduce the missclassification rate of SVM model,it can also defines as the vector which classifies the binary data when controlling the margin between them.

A small parameter C controls a large minimum margin between both side of data, which means the hyperplane of which have the wide margin between the two classes, however, a small C will sometimes causes more missclassfication rate to the result since it would probably underfit the model.

A large value of parameter C means a narrow margin between classes, sometimes it will be better off than the small C classifier, however, it will probably lead to a overfitting model. Therefore, we can not conclude how large the value of parameter C is better, because it always depends on different models and datas, but we can try to reduce the missclassfication rate by choosing the most promising C with holdout methods or cross-validations method etc.

#Appendix

```
# Assignment 1

library(knitr)
com <- data.frame(locationdist=c(0,100,1000,10000,
                                 50000,100000,
                                 200000,500000,1000000),
                  h_10km=rep(0,9))
     com[,2] <- exp(-com[,1]/20000)

kable(caption = "different distances with different weights",
      com)
# second h_date
  com2 <- data.frame(datedist=c(0,1,3,7,30,60,180,300,500),
                     h180=rep(0,9))

  com2[,2] <- exp(-com2[,1]/30)
  kable(caption = "different distances with different weights",
      com2)

# third h_time
# take the minute as the unit of the distance of time
  com3 <- data.frame(timedist=c(0,40,60,90,120,
                                150,200,250,300,500),
                     h120=rep(0,10))
  com3[,2] <- exp(-com3[,1]/120)

kable(caption = "different distances with different weights",
      com3)

setwd("C:/Users/Jooyoung/Documents/ML/Lab3-1")

##WANT: TEMPERATURE FORECAST for a date and a place
####### TIME 4-24 WTIH 2 HOUR INTERVAL

set.seed(1234567890)
library(geosphere)
library(readr)
stations <- read_csv("stations.csv")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps, by="station_number")

# These three values are up to the students
## SET FOR LOOP FOR CONSTRUCTING THE VECTOR

h_distance <- 20000
h_date <- 30
h_time <- 2
```

```
### PLACE DATE AND TIME ARE SET
a <- 58.4274
b <- 14.826
date <- "2013-11-04"
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", "24:00:00")
temp1 <- vector(length=length(times))
temp2 <- vector(length=length(times))

# Students' code

#filter data
filtered_st <- st[which(as.Date(date)>as.Date(st$date)),]
############ REMOVE THIS AFTER DONE
#filtered_st <- filtered_st[1:100,]
#produce kernel values
## kernel 1 and kernel 2 : fixed place and date
place_date <- data.frame(K1=as.numeric(), K2=as.numeric(),
                         stringsAsFactors = FALSE)
for (i in 1:nrow(filtered_st)) {
  k1 <- distHaversine(c(b,a), c(filtered_st$longitude[i],
                                filtered_st$latitude[i]))
  k1 <- exp(-(abs(k1)/h_distance))
  k2 <- as.numeric(difftime(date, filtered_st$date[i], units="days"))
  #December of 1950 is more likely to have higher correlation than
  # May of 2012 -- remove the year part
  k2 <- k2-((floor(k2/365))*365)
  k2 <- exp(-(abs(k2)/h_date))
  new <- data.frame(K1=k1, K2=k2)
  place_date <- rbind(place_date, new)
}

## produce k3 and use it for prediction
for (j in 1:length(times)) {
  target <- times[j]
  num1 <- 0
  denom1 <- 0
  num2 <- 0
  denom2 <- 0
  for (i in 1:nrow(place_date)) {
    k3 <- as.numeric(difftime(as.POSIXct(target, format="%H:%M:%S"),
                  as.POSIXct(filtered_st$time[i], format="%H:%M:%S"),
                  units="hours"))
    k3 <- exp(-(abs(k3)/h_time))
    num1 <- num1+
      (place_date$K1[i] + place_date$K2[i] + k3)*filtered_st$air_temperature[i]
    num2 <- num2+
      (place_date$K1[i] * place_date$K2[i] * k3)*filtered_st$air_temperature[i]
    denom1 <- denom1+(place_date$K1[i] + place_date$K2[i] + k3)
    denom2 <- denom2+(place_date$K1[i] * place_date$K2[i] * k3)
  }
  temp1[j] <- num1/denom1
```

```
  temp2[j] <- num2/denom2
}

plot(main="Temperature Forecast", x=seq(from=4, to=24, by=2),
     y=temp1, type="o", ylab="Temperature", xlab="Time",
     ylim=c(1.5,9))
lines(x=seq(from=4, to=24, by=2), y=temp2, lty="dotted")
legend("bottom", legend=c("Summed Kernel", "Multiplied Kernel"),
       lty=c(1,2))

# Assignment 2

library(kernlab)
data(spam)

set.seed(12345)
n = dim(spam)[1]

id = sample(1:n, floor(n*0.5) )
train <- spam[id,]
id1<- setdiff(1:n, id)
id2<- sample(id1, floor(n*0.25))
valid=spam[id2,]
id3=setdiff(id1,id2)
test=spam[id3,]

C <- c(0.5, 1,5)
h <- 0.05

#1.

error <- data.frame(C=C, train_error=c(0,0,0))
for ( i in 1:3){
  svm_model <- ksvm(type~., data=train, C=C[i],kernel="rbfdot" ,
                    type="C-svc", kpar=list(sigma=h))
  res <- predict(svm_model, newdata=train)
  res_conf <- table(predicted=res, actual=train$type)
  res_error <- 1-sum(diag(res_conf))/sum(res_conf)
  error$train_error[i] <- res_error

}
error

valid_error <- data.frame(C=C, valid_error=c(0,0,0))
for ( i in 1:3){
  svm_model <- ksvm(type~., data=train, C=C[i],kernel="rbfdot" ,
                    type="C-svc", kpar=list(sigma=h))
  res <- predict(svm_model, newdata=valid)
  res_conf <- table(predicted=res, actual=valid$type)
  res_error <- 1-sum(diag(res_conf))/sum(res_conf)
  valid_error$valid_error[i] <- res_error

}
```

```r
cbind(error, valid_error=valid_error$valid_error)
```