Machine Learning - Lab 3 Block 1 - Group A12

Mubarak Hussain, Ali Etminan, Abhinay Krishna

2019/12/17

Assignment 1 - Kernel Methods

Distance Kernel Width

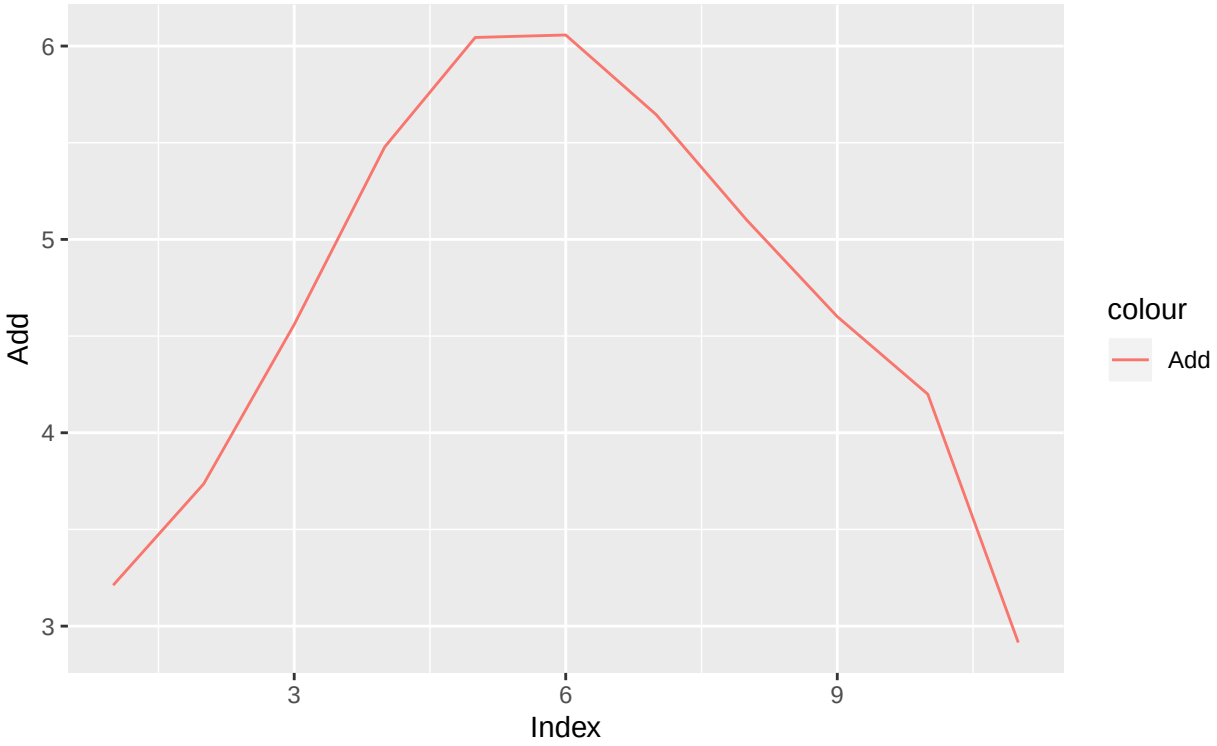
This prediction is been made for the 4th of November 2014, from 4 in the morning to 24 at night for that day. As the first step in the process, the distance between the point of interest and the stations were calculated and used to create the first kernel. The smoothing coefficient or the width of this kernel here was set to 30000. Distances are presented in meters, therefore a $30km$ range to give the closer stations a larger weight was considered.

Date Kernel Width

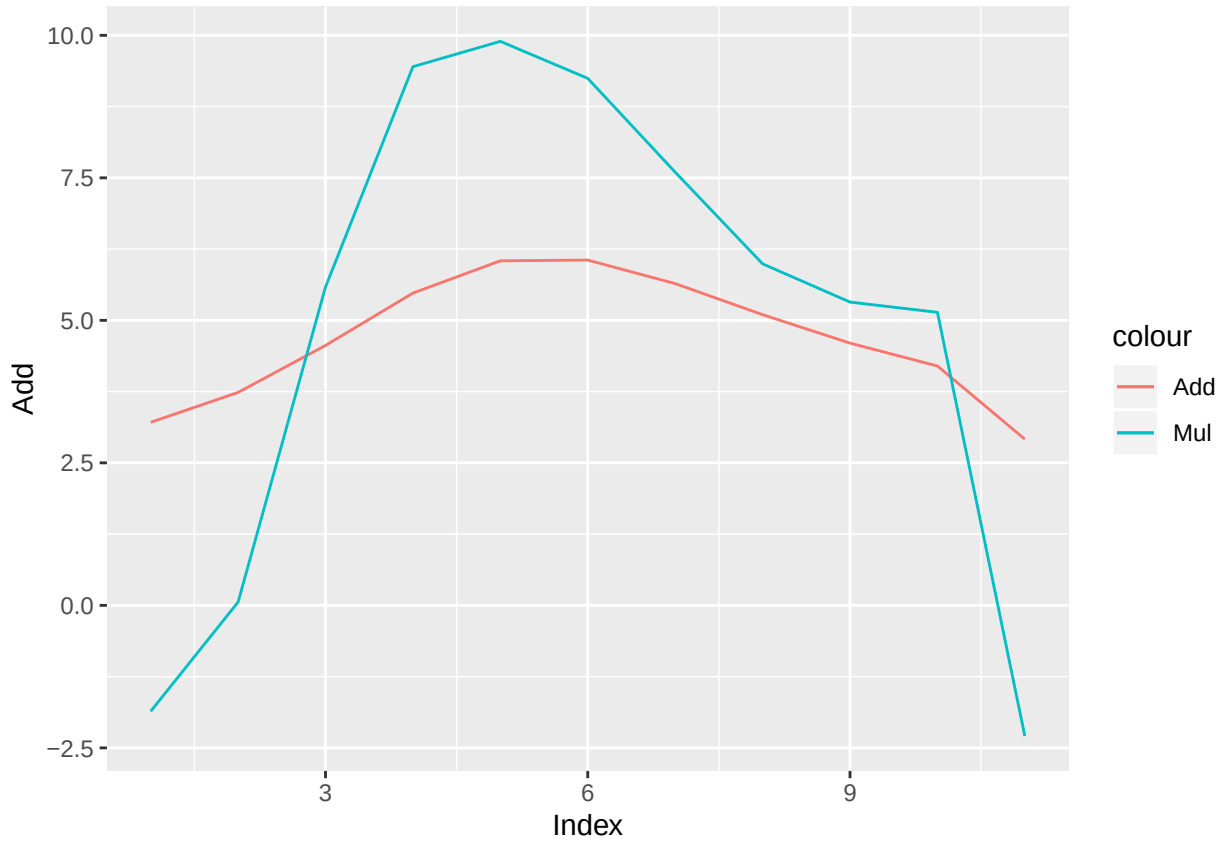
The data was reordered according to the *date* column to omit dates posterior to the *prediction date*. The date difference were then computed in terms of *days* and used to create the date kernel. The kernel width here was set to 12, giving more weight to 10 days around the prediction date. The tempreatre usually does not change much in this interval.

Time Kernels' Width

Since we are to predict the tempreture of different times throughout a day, a kernel for each time point was created. A smoothing coefficient of 4 was selected for all the time kernels. This means that the measurements taken in a 3 hour range from the desired time will be considered.


Add
6
5
4
3
3
6
9
Index
colour
Add


Mul
10.0
7.5
5.0
2.5
0.0
−2.5
3
6
9
Index
colour
Mul



Analysis for Kernels

The comparitive plot indicates that adding the three kernels results in a different range of predicted temperatures compared to multiplied kernels. Since the additive model takes an average of the three kernels, the model results in a smoother graph of predictions. We can conclude that the additive model could be the better options since it avoids sudden changes in the predictions.

Assignment 2 - Support Vector Machines

The spam data is first divide (70/30) into training and test sets.

```
data(spam)
data <- spam
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.70))
train=data[id,]
test=data[-id,]
```

We are now going to perform the model selection for different values of **C**

```
##
## Model misclassification rate for C =  0.5 is:  0.07530775
```

```
##          pred.model
##           nonspam spam
##   nonspam     832   27
##   spam         77  445

##
## Model misclassification rate for C =  1 is:  0.06806662

##          pred.model
##           nonspam spam
##   nonspam     832   27
##   spam         67  455

##
## Model misclassification rate for C =  5 is:  0.07023896

##          pred.model
##           nonspam spam
##   nonspam     828   31
##   spam         66  456
```

According to the results from the 3 models predicted on the test data, the model with $C = 1$ yields the smallest classification error of 0.07458364. Thus, we are going to choose this model for the next step. This is worth noting that the classification error for this model is very close to the model with $C = 5$.

```
model.new <- ksvm(type~., data = train, kernel = "rbfdot", kpar=list(sigma=0.05), C = 1)
model.new
```

```
## 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 : 1277
##
## Objective Function Value : -600.9689
## Training error : 0.037267
```

The purpose of C?

For Support Vector Machines method using Kernel classification, C is a parameter termed as "cost of constraints violation". *ksvm* function solves a sequence of sub problems using SMO Algorithm. For n samples, there aree n(n-1)/2 number of possible sub problems. Every sub problem include pair of variables. Usually, there is no need to solve all the sub problems as in some of them have variables that violate the constraints. We define the cost of constraints violation value to filter out the sub problems that have the variables with violations and an optimum model is obtained based on accuracy.

In terms of bias-variance trade off, C acts as an adjustment parameter. Larger values of C, results in more variance and consecutively less bias. C penalizes larger residuals so when its value is increased, less residuals are allowed. Thus, variance is increasesd, allowing less misclassifications.

Appendix

```
library(geosphere)
library(kernlab)
library(ggplot2)

knitr::opts_chunk$set(echo = TRUE)

kernal_methods <- function(st, date, lat, lon, time_seq, h_date, h_time, h_distance){
  st$h_dist = abs(distHaversine(p1 = c(lon, lat), p2 = st[,c("longitude", "latitude")]))
  st$h_dist = exp(-(st$h_dist/h_distance)^2)
  st$h_date = as.numeric(difftime(date, st$date, units = c("days")))
  st$h_date = ifelse(st$h_date>0, st$h_date, 0)
  st = subset(st, st$h_date!=0)
  st$h_date = exp(-(st$h_date/h_date)^2)
  times = c()
  for(t in 1:length(time_seq)){
    d = as.Date(time_seq[t])
    c_time = format(time_seq[t], "%H:%M:%S")
    times = append(times, c(c_time))
    st[c_time] =  as.numeric(abs(difftime(strptime(paste(d, c_time),
                                                   "%Y-%m-%d%H:%M:%S"),
                                          strptime(paste(d, st$time),
                                                   "%Y-%m-%d%H:%M:%S"),
                                          units = c("hour"))))
    st[c_time] = exp(-(st[c_time]/h_time)^2)
  }
  temp = st[times]
  temp_add_d = temp + (st$h_date + st$h_dist)
  temp_mul_d = temp * (st$h_date * st$h_dist)
  temp_add_n = temp_add_d*st$air_temperature
  temp_mul_n = temp_mul_d*(st$air_temperature)

  temp_add = colSums(temp_add_n)/colSums(temp_add_d)
  temp_mul = colSums(temp_mul_n)/colSums(temp_mul_d)

  d = data.frame(Index = 1:length(times), Time = times, Add = temp_add, Mul = temp_mul)
  return(d)
}

set.seed(1234567890)
stations = read.csv("C:/Users/romina/Desktop/ali/Machine Learning/stations.csv")
temps = read.csv("C:/Users/romina/Desktop/ali/Machine Learning/temps50k.csv")
st <- merge(stations,temps,by="station_number")
#colnames(st)
h_distance <- 30000
h_date <- 12
h_time <- 4
lat <- 58.4274
lon <- 14.826
date <- "2014-05-04"
start <- as.POSIXct(date)
interval <- 60
```

```
end <- start + as.difftime(1, units="days")
time_seq <- seq(from=start, by=interval*120, to=end)
time_seq = time_seq[3:length(time_seq)]

pred = kernal_methods(st, date, lat, lon, time_seq, h_date, h_time, h_distance)


ggplot(pred) + geom_line(aes(Index, Add, col="Add"))


ggplot(pred) + geom_line(aes(Index, Mul, col="Mul"))


ggplot(pred) + geom_line(aes(Index, Add, col="Add")) + geom_line(aes(Index, Mul, col="Mul"))


data(spam)
data <- spam
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.70))
train=data[id,]
test=data[-id,]


ksvm.model <- function(x) {

model <- ksvm(type~., data = train, kernel = "rbfdot", kpar=list(sigma=0.05), C = x)
pred.model <- predict(model, test)
confmat <- table(test$type, pred.model)
misclass <- 1 - sum(diag(confmat)/sum(confmat))
cat("\nModel misclassification rate for C = ", x, "is: ", misclass)
confmat
}

ksvm.model(0.5)
ksvm.model(1)
ksvm.model(5)


model.new <- ksvm(type~., data = train, kernel = "rbfdot", kpar=list(sigma=0.05), C = 1)
model.new
```