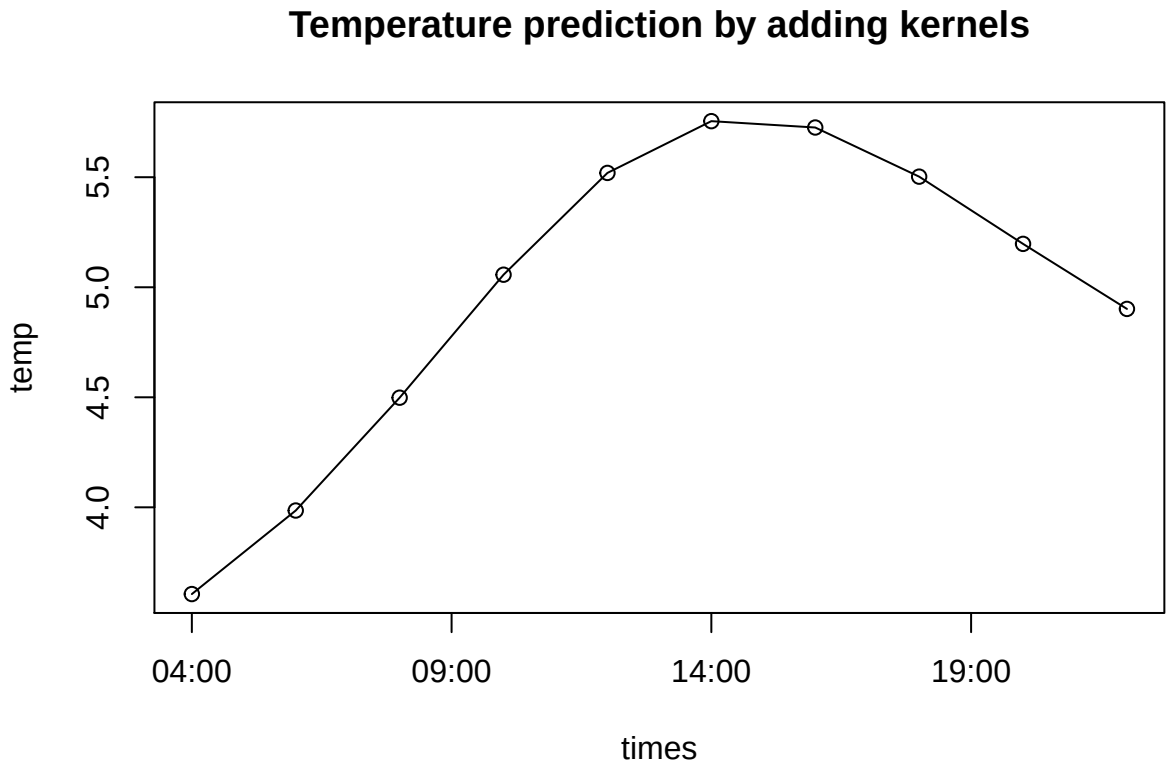
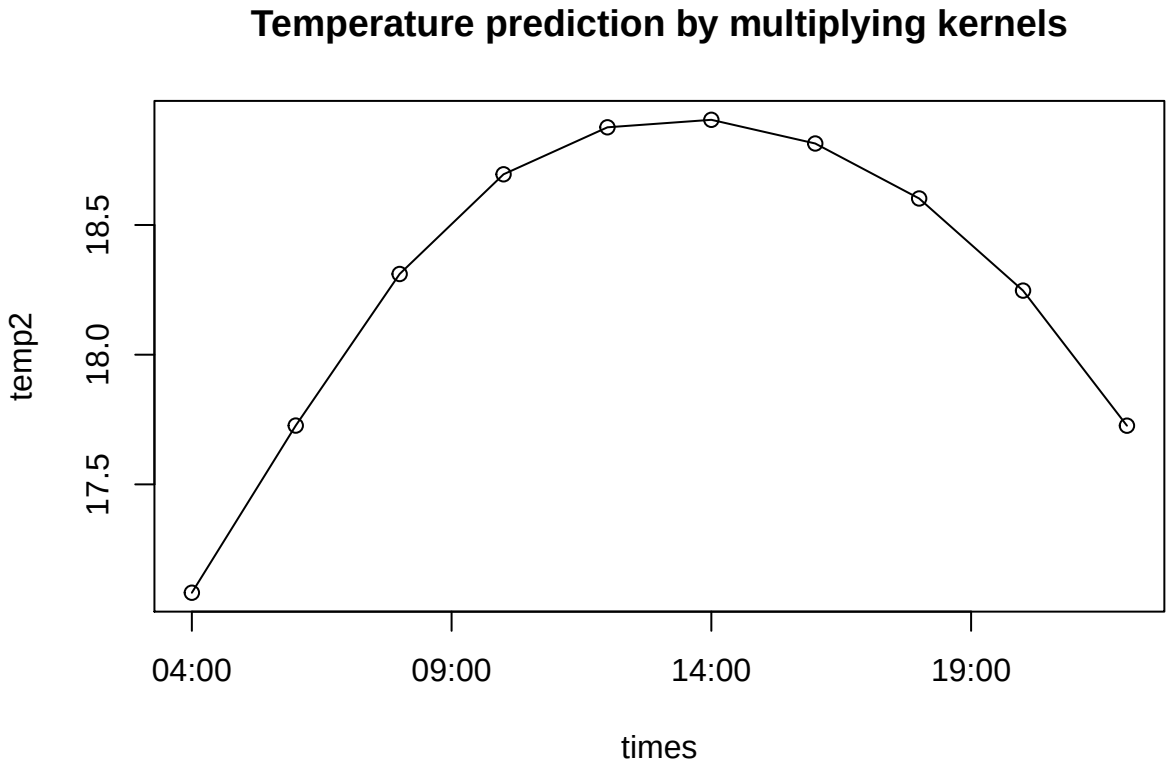
lab 3

vinbe289

December 18, 2018


Temperature prediction by adding kernels
temp
4.0
4.5
5.0
5.5
04:00
09:00
14:00
19:00
times


Temperature prediction by multiplying kernels
temp2
17.5
18.0
18.5
04:00
09:00
14:00
19:00
times

Choice of Kernels:

Width for the distance kernel is chosen to be 50,000 meters i.e. 50 kilometers since it is reasonable to give more weightage to the stations which are within the radius of 50 kilometers from the point of interest. Width for the date kernel is chosen to be 7 days since it is reasonable to consider temperature measurements of 7 days prior to the date of interest to contribute more to prediction of temperature on the particular day of interest as the weather remains almost the same over a week. Width for the time kernel is chosen to be 6 hours since it seems to be reasonable to consider prior 6 hours to contribute more towards the prediction of temperature at a particular point of time as i have considered 1/4th part of the day.

Comparing the results obtained in both cases and elaborating on why they may differ:

By looking at the plot it is evident that the plot for additive kernel and multiplicative kernel is completely different. The predicted temperatures is different for additive and multiplicative kernels. The predicted temperature peaks to its highest which is around 5.5 at around 14'o clock for additive kernel curve, whereas the predicted temperature peaks to its highest which is around 19 at around 14'o clock for multiplicative kernel curve. However, in both the cases the temperature increses until noon i.e around 14'o clock and then the temerature starts to decrease again. The multiplicative kernel curve seems to predict the temperature reasonably well since any one of the Gaussian kernel's having more weight, due to less width or distance, would not affect the multiplicative kernel very much by decreasing or increasing its value, i.e. predicted temperature value. In the case of additive kernels the final value of additive kernel, i.e. predicted temperature value, can be affected even if one of the individual Gaussian kernels have a higher weightage due to lesser distance. Considering small Gaussian kernal distance makes the prediction more sensitive to the obsevations close to the point of interest and considering large Gaussian kernal distance makes the prediction less sensitive to the obsevations close to the point of interest but takes into account even the obsevations which are not closer to the point of interest.

2. SUPPORT VECTOR MACHINES

1

```
## [1] 3
```

```
## [1] 9.556907 8.514335 9.122502
```

2

```
# Considering the model that has lowest error rate
svm2 <- ksvm(type~., data=train, kernel="rbfdot", C=1, kpar=list(0.05))
predicted2 <- predict(svm2, newdata=valid)
conMat2 <- table(predicted2, valid$type)
errorRate2 <- 1-sum(diag(conMat2))/sum(conMat2)
(errorRate2)*100
```

```
## [1] 6.956522
```

3

Purpose of parameter C:

Parameter C controls the extent of regularization, which in turn controls the overfitting or under fitting of model to the training dataset. The value of C should neither be too small nor too large as it may lead to underfitting or overfittig. In this case the optimum value for parameter C is found to be 1 since it gives the lowest test error.

Appendix

```
#1. KERNEL METHODS

set.seed(1234567890)
library(geosphere)

stations <- read.csv("stations.csv")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")

# Defining width for each kernel
h_distance <- 50000 # in meters
h_date <- 7 # in days
h_time <- 6 # in hours

# The point to predict
a <- 58.4274
b <- 14.826

# The date to predict
date <- "2016-07-02"

# The time to predict
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")

temp <- vector(length=length(times))
temp2 <- vector(length=length(times))

# Gaussian distance kernel
gaus_dist <- function(loc_stat,loc_int){
  dist_diff <- distHaversine(loc_stat,loc_int)
  return (exp(-(dist_diff / h_distance)^2))
}

# Gaussian date kernel
gaus_date <- function(date_stat, date_int){
  date_diff <- as.numeric(difftime(date_int,date_stat,unit = "days"))
  return (exp(-(date_diff / h_date)^2))
}

# Gaussian time kernel
gaus_hours <- function(time_stat, time_int){
  time_diff <- as.numeric(difftime(time_int,time_stat,unit = "hours"))
  return (exp(-(time_diff / h_time)^2))
```

```
}

dist_gaus <- gaus_dist(st[,c("longitude", "latitude")], c(b,a))
date_gaus <- gaus_date(st$date,date)

# Considering system date and station time
con_time <- data.frame(date_time=as.POSIXct(paste(Sys.Date(), st$time),"%Y-%m-%d %H:%M:%S"))
#con_time2 <- con_time[1:nrow(con_time),]

# Considering system date and time of interest
times <- strptime(paste(Sys.Date(),times),"%Y-%m-%d %H:%M:%S")

for(i in 1:length(times)){
  # con_time3 <- as.data.frame(con_time2[which(con_time2<times[i])])
  # names(con_time3) <- c("date_time")
  hours_gaus <- gaus_hours(con_time$date_time,times[i])

  # Adding all kernels
  sum_kernels <- dist_gaus+date_gaus+hours_gaus
  temp[i] <- sum(sum_kernels*st$air_temperature)/sum(sum_kernels)

  # Multiplying all kernels
  mul_kernels <- dist_gaus*date_gaus*hours_gaus#which of these are close to 0,that is making the mul_ke
  temp2[i] <- sum(mul_kernels*st$air_temperature)/sum(mul_kernels)
}

plot(times, temp, type="o", main="Temperature prediction by adding kernels")
plot(times, temp2, type="o", main="Temperature prediction by multiplying kernels")

#2. SUPPORT VECTOR MACHINES

##1

library(kernlab)
data("spam")

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,]

errorRate <- c()
var1 <- c(0.5,1,5)
length(var1)
for(i in 1:length(var1)){
  svm <- ksvm(type~., data=train, kernel="rbfdot", C=var1[i], kpar=list(0.05))
  predicted <- predict(svm, newdata=test)
```

```
  conMat <- table(predicted, test$type)
  errorRate[i] <- 1-sum(diag(conMat))/sum(conMat)
}
(errorRate)*100

##2

# Considering the model that has lowest error rate
svm2 <- ksvm(type~., data=train, kernel="rbfdot", C=1, kpar=list(0.05))
predicted2 <- predict(svm2, newdata=valid)
conMat2 <- table(predicted2, valid$type)
errorRate2 <- 1-sum(diag(conMat2))/sum(conMat2)
(errorRate2)*100
```