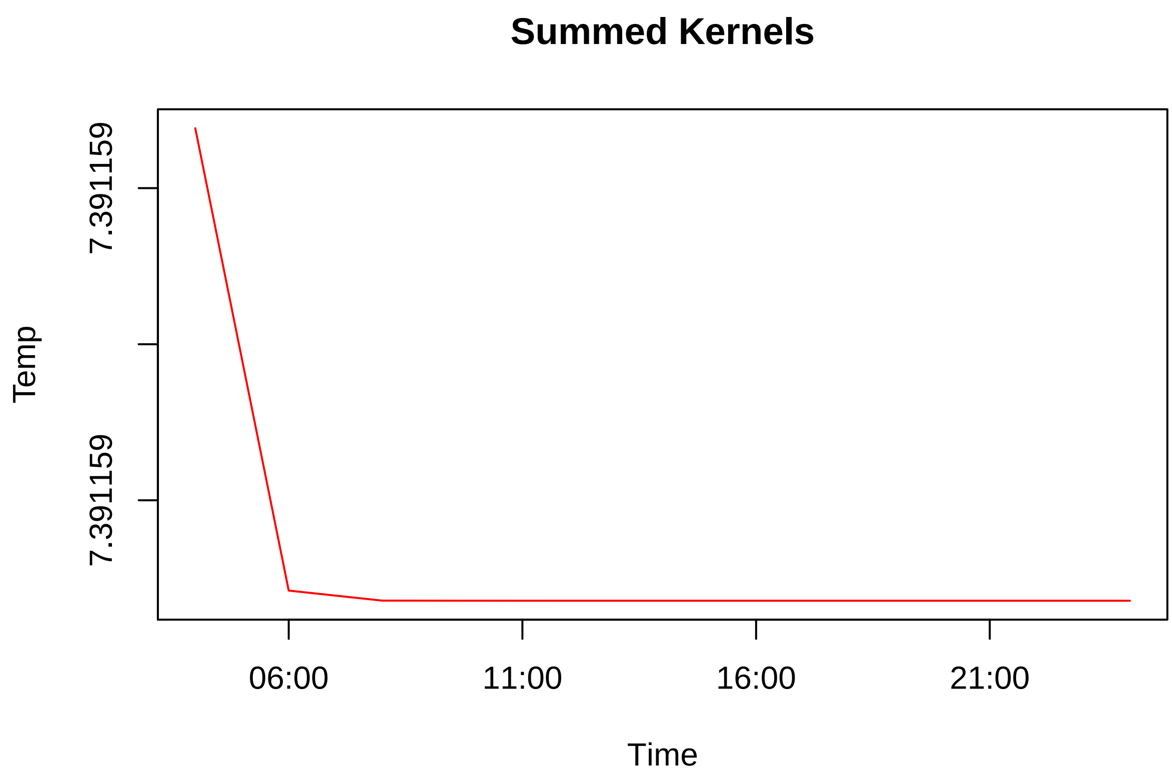
lab03-block01

Mohammed Bakheet

16/12/2019

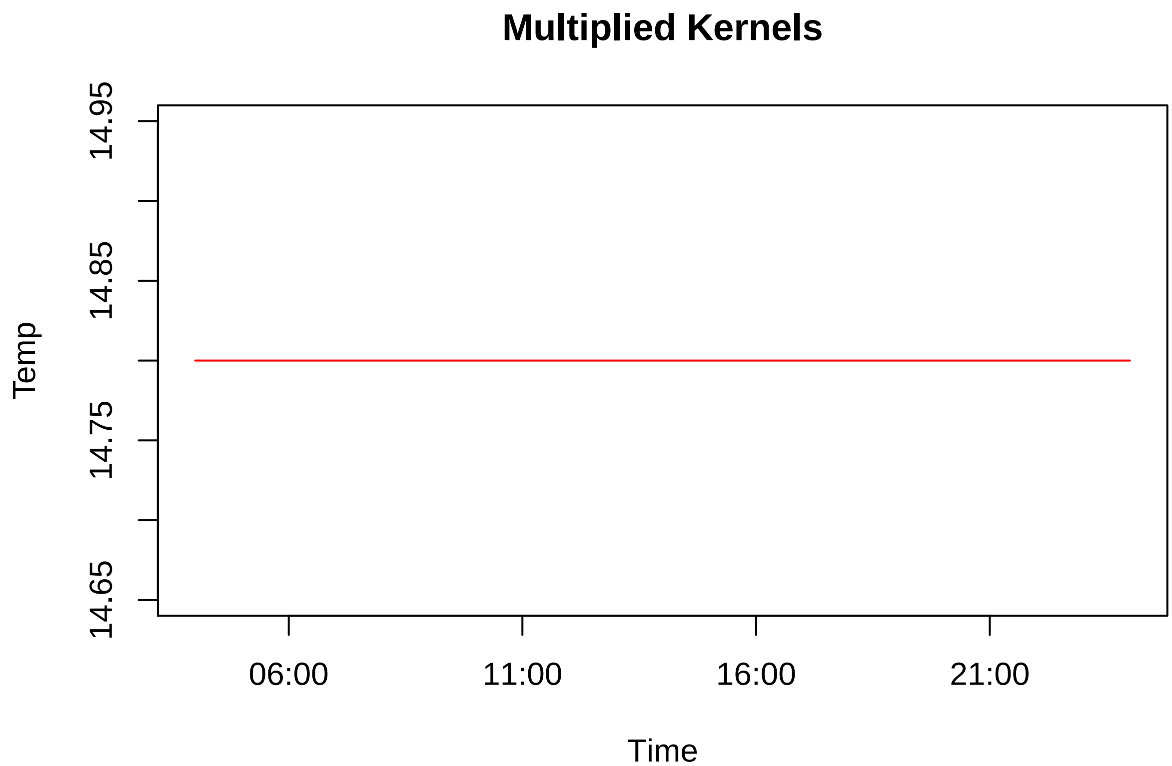
Assignment 1: KERNEL METHODS

After counting the distance from a station to the point of interest (2016-07-02), the distance between the day a temperature measurement was made and the day of interest, and the distance between the hour of the day a temperature measurement was made and the hour of interest, the following results were acquired for the summed kernels and multiplied kernels respectively.



Kernels' width choice:

When selecting a date of "2016-07-02", the kernal method gives a high weight to "2016-07-01" and "2016-06-30" respectively in the dataset which is the closest date to this date. We can, therefore, conclude that our choice of kernel's width is sensible.



#Kernels multiplication is one of the kernel tricks. When adding two kernels instead of summing them we don't see the function that takes a vector and adds an extra coordinate of sqrt and then project the resulting vector into the unit sphere, and the resulting kernel will have high value if and only if the two kernels have high values, this will result in a function that is quadratic, the gaussian kernel function, in this case takes the mupltiplied kernels and applies the calculation on it. Moreover, when multiplying k with a positive definite, this leads to the positive definiteness of the Gaussian kernel. Whereas when a linear kernel is periodic, this will result in a periodic function, and this explains why we got a straight line with one temprature when multiplying kernels.

Assignment2: SUPPORT VECTOR MACHINES

#When c is equal to 0.5

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 0.5
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.05
##
## Number of Support Vectors : 1063
##
## Objective Function Value : -304.0238
## Training error : 0.044783
```

#When c is equal to 1

```
## 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 : 964
##
## Objective Function Value : -446.3466
## Training error : 0.037826
```

#When c is equal to 5

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 5
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.05
##
## Number of Support Vectors : 918
##
## Objective Function Value : -1016.625
## Training error : 0.017826
```

#Using prediction to check the model errors against the testing data:

Table 1: Testing Data Missclassification Comparison Table

c05	c1	c5
0.0916993	0.0838766	0.0830074

Table 2: Training Data Missclassification Comparison Table

c05	c1	c5
0.0916993	0.0838766	0.0830074

#The most promising model is when c is equal to 5, where the missclassification rate is minimal for the training and testing datasets.

```
## confusionMatrix When c is equal to 0.5
```

```
##           Reference
## Prediction nonspam spam
```

```
##    nonspam    1346  155
##    spam         56  744
```

```
## confusionMatrix When c is equal to 1
```

```
##           Reference
## Prediction nonspam spam
##    nonspam    1340  131
##    spam         62  768
```

```
## confusionMatrix When c is equal to 5
```

```
##           Reference
## Prediction nonspam spam
##    nonspam    1336  125
##    spam         66  774
```

#The model that should be returned to the user:

```
returnedC5Model =  ksvm(type~., data = spam, C = 5,kpar=list(sigma=0.05))
     returnedC5Model
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 5
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.05
##
## Number of Support Vectors : 1547
##
## Objective Function Value : -2082.468
## Training error : 0.022169
```

#C is the cost of constraints violation, it takes a default value of 1, this is the constant of the regularization term in the Lagrange formulation. The constraint force C could be seen as the tension in the rod. When the C value is very high the model's error increases and it's not advices to have a low value of C as well.

Code Appendix

```
RNGversion('3.5.1')
knitr::opts_chunk$set(echo = TRUE)
packages <- c("ggplot2", "plotly","readxl","tree", "MASS", "e1071", "boot", "fastICA","mgcv","akima","p
options(tinytex.verbose = TRUE)
library(chron)
library(geosphere)
stations <- read.csv("D:/Desktop/Machine Learning/Machine Learning/lab03 block 1/stations.csv", fileEnc
temps <- read.csv("D:/Desktop/Machine Learning/Machine Learning/lab03 block 1/temps50k.csv")
```

```r
      set.seed(1234567890)

      st <- merge(stations,temps,by="station_number")
      h_distance <- 25000 # These three values are up to the students
        h_date <- 11
        h_time <- 5
        a <- 58.4274 # The point to predict (up to the students)
        b <- 14.826
      date <- "2016-07-02" # The date to predict (up to the students)
      times <- c("4:00:00","6:00:00","8:00:00","10:00:00","12:00:00","14:00:00",
                 "16:00:00","18:00:00","20:00:00","22:00:00","24:00:00")
      temp <- vector(length=length(times))
# Student code here
#a) Calculating the distance difference

      cbind(st,"gaussianDistance")
      for(i in 1:nrow(st)){
        x <- distHaversine(c(b,a),c(st[i,5],st[i,4]))
        gaussianCalculation <- exp(-(x/h_distance)^2)
        st[i,"gaussianDistance"]<- gaussianCalculation
      }
#b) Calculating the dates difference
      cbind(st,"dateDifference")
      date <- as.Date(date, format = "%Y-%m-%d")
      for (i in 1:nrow(st)){
        stDate <- as.Date(st$date[i], format ="%Y-%m-%d" )
        x <- as.numeric(date-stDate)
        dateGaussian <- exp(-(x/h_date)^2)
        st[i,"dateDifference"]<- dateGaussian
      }
#c) Calculating the time difference

      timeDiff <- matrix(nrow = nrow(st), ncol = length(times))
      colnames(timeDiff) <- times
      for(i in 1:nrow(st)){
        for(j in times){
        x<- as.numeric(difftime(strptime(paste(date,j), format = "%Y-%m-%d%H:%M:%S"),
                    strptime(paste(st[i,"date"],st[i,"time"]), format = "%Y-%m-%d%H:%M:%S"),units="hou
        xExp <- exp(-(x/h_time)^2)
        timeDiff[i,j]<- xExp
        }
      }

#d) The coice of kernel


#Adding the kernels

      for(i in 1:length(times)){
        individualkernalSum <- st$gaussianDistance + st$dateDifference + timeDiff[,i]
        typeof(individualkernalSum)
        totalKernalSum <- sum(st$air_temperature*individualkernalSum, na.rm = T)/sum(individualkernalSu
        temp[i] <- totalKernalSum
```

```
      }
      orderedTime <- as.POSIXct(strptime(paste(date,times), format="%Y-%m-%d%H:%M:%S"))
plottedData <- data.frame(temp = temp, time = orderedTime)

plot(plottedData$time, plottedData$temp, xlab="Time", ylab = "Temp", typ = "l", col = "red", main = "Su
#e) Multiplying the kernels
        temp2 <- vector(length=length(times))
        for(i in 1:length(times)){
          individualkernalSum <- st$gaussianDistance * st$dateDifference * timeDiff[,i]
          typeof(individualkernalSum)
          totalKernalSum <- sum(st$air_temperature*individualkernalSum, na.rm = T)/sum(individualkernal
          temp2[i] <- totalKernalSum
        }
        plottedData2 <- data.frame(temp = temp2, time = orderedTime)

        plot(plottedData2$time, plottedData2$temp, xlab="Time", ylab = "Temp", typ = "l", col = "red",

library(kernlab)
library(caret)


data(spam)
n =  nrow(spam)
rand = sample(1:n,floor(n*0.5))
#Dividing data into training and testing
training = spam[rand,]
testing = spam[-rand,]

      c0.5Model =  ksvm(type~., data = training, C = 0.5,kpar=list(sigma=0.05))
      c0.5Model
      c1Model =  ksvm(type~., data = training,kpar=list(sigma=0.05))
      c1Model
      c5Model =  ksvm(type~., data = training, C = 5,kpar=list(sigma=0.05))
      c5Model
library(knitr)
predict.C05.train = predict(c0.5Model,newdata = training)
predict.C1.train = predict(c1Model,newdata = training)
predict.C5.train = predict(c5Model,newdata = training)


predict.C05 = predict(c0.5Model,newdata = testing)
predict.C1 = predict(c1Model,newdata = testing)
predict.C5 = predict(c5Model,newdata = testing)
#Calculating the missclassification rate
missClassification = data.frame(c05 = mean(predict.C05 != testing$type),
                     c1 = mean(predict.C1 != testing$type),
                     c5 = mean(predict.C5 != testing$type))
missClassificationTrain = data.frame(c05 = mean(predict.C05 != testing$type),
                     c1 = mean(predict.C1 != testing$type),
                     c5 = mean(predict.C5 != testing$type))
kable(missClassification, caption="Testing Data Missclassification Comparison Table")
kable(missClassificationTrain, caption="Training Data Missclassification Comparison Table")
cat("confusionMatrix When c is equal to 0.5", sep = "\n")
```

```r
confusionMatrix(predict.C05,testing$type)$table
cat("confusionMatrix When c is equal to 1", sep = "\n")
confusionMatrix(predict.C1,testing$type)$table
cat("confusionMatrix When c is equal to 5", sep = "\n")
confusionMatrix(predict.C5,testing$type)$table
 returnedC5Model =  ksvm(type~., data = spam, C = 5,kpar=list(sigma=0.05))
      returnedC5Model
```