Block 1 - Lab 3

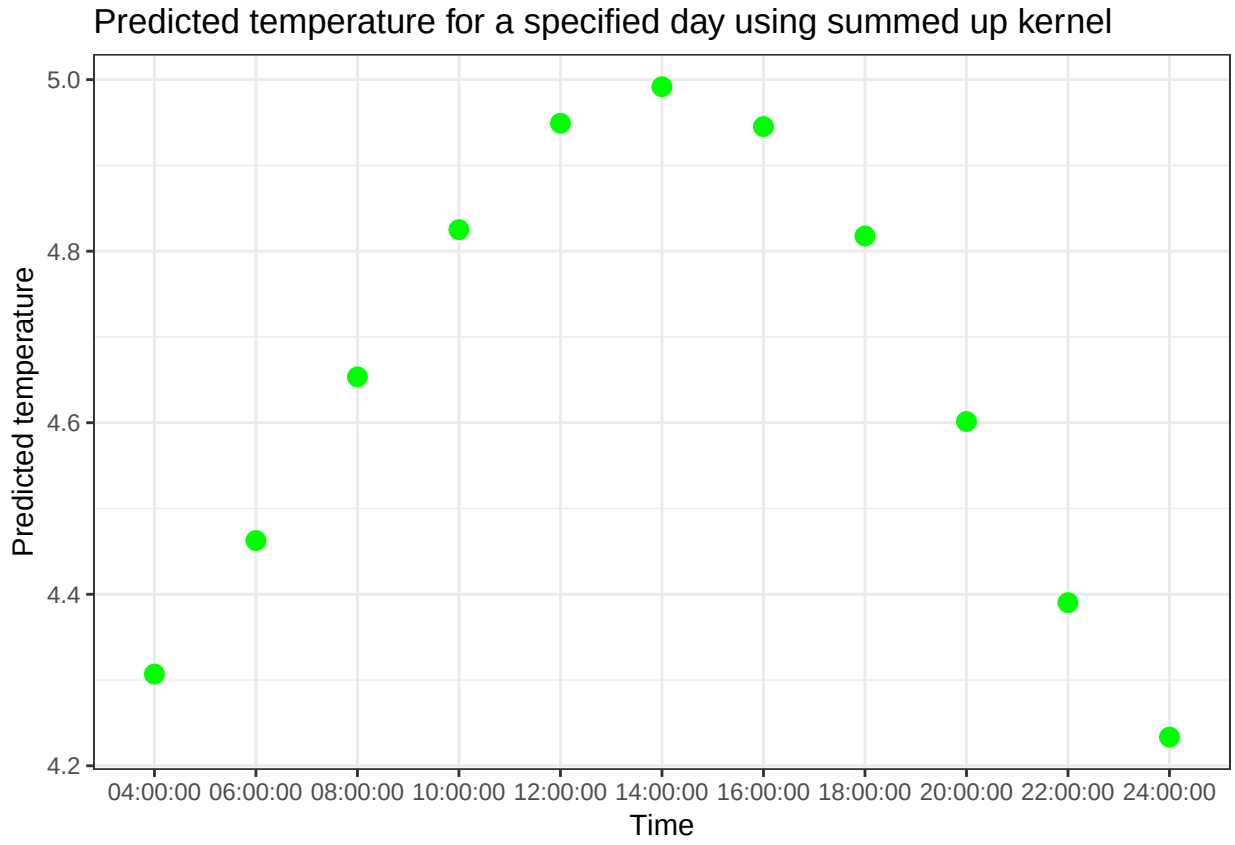
Roshni Sundaramurthy

18 December 2018

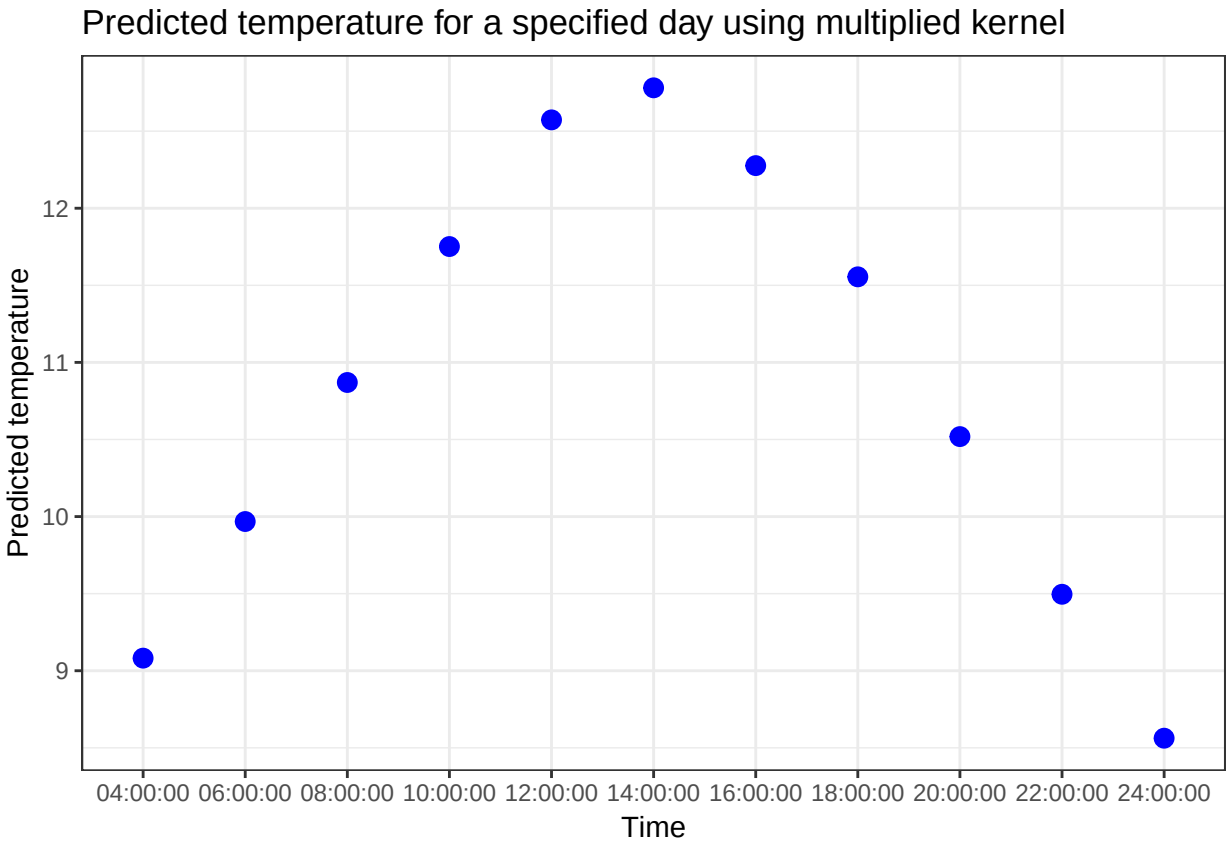
1. KERNEL METHODS

```
###################### Assignment 1 ##########################
set.seed(1234567890)
library(geosphere)
stations <- read.csv("stations.csv")
temps <- read.csv("temps50k.csv")
# merging two datasets using station number
st <- merge(stations,temps,by="station_number")
h_distance <- 40000
h_day <- 7
h_time <- 12
pred_lat <-  67.4432      #66.50502 # The point to predict (up to the students)
pred_long <-  21.4488     #19.67495
date <- as.Date("2016-5-30") # The date to predict (up to the students)
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")

temp <- length(times)

# Location formatting
# Distance between all stations and our location
newloc <- distHaversine(cbind(st$latitude,st$longitude), c(pred_lat,pred_long))

# Date formatting
#distance between the day a temperature measurement was made and the day of interest
st$date <- difftime(as.Date(st$date),as.Date(date),units = "days")
st$date <- as.numeric(abs(st$date))

# "U" calculation for location and days
u_loc <- newloc / h_distance
u_day <- st$date / h_day

# Kernels for location and days
K_loc <- exp(-(u_loc)^2)
K_day <- exp(-(u_day)^2)

# Time formatting for consecutive hours
times1 <- as.POSIXct(times,format="%H:%M:%S")
st$time <- as.POSIXlt(st$time, format="%H:%M:%S")

y_Sum_kern <- c() # initializing kernels
y_Mult_kern <- c()
weather_data <- data.frame() # final dataset of predicted values

for(i in 1:temp){
```

```r
  # Distance between the hour of the day a temperature measurement was made and the hour of interest
  newtime <- as.numeric(difftime(st$time ,times1[i], units = "hours"))
  newtime <- abs(newtime)
  newtime[newtime > 12] = 24 - newtime[newtime > 12]

  # "U" for time
  u_time <- newtime / h_time
  # Kernel for time
  K_time <- exp(-(u_time)^2)

  # Sum and prod of 3 Gaussian kernels
  Sum_kern <- K_loc + K_day + K_time
  Mult_kern <- K_loc * K_day * K_time

  y_Sum_kern[i] <- sum(Sum_kern*st$air_temperature) / sum(Sum_kern)
  y_Mult_kern[i] <- sum(Mult_kern*st$air_temperature) / sum(Mult_kern)

  data <- data.frame(Time = times[i],y_Sum_kern = y_Sum_kern[i], y_Mult_kern = y_Mult_kern[i])
  weather_data <- rbind(weather_data, data)
}
library(ggplot2)
ggplot(weather_data)+
  geom_point(aes(y=y_Sum_kern,x=times),color="green",size=3)+
  ggtitle("Predicted temperature for a specified day using summed up kernel")+
  xlab("Time")+ ylab("Predicted temperature")+theme_bw()
```



```
ggplot(weather_data)+
  geom_point(aes(y=y_Mult_kern,x=times),color="blue",size=3)+
  ggtitle("Predicted temperature for a specified day using multiplied kernel")+
  xlab("Time")+ ylab("Predicted temperature")+ theme_bw()
```



Analysis:

My choice for the kernels' width is as shown below, where h is smoothing factor.

```
tabl <- "
| Variables   |   Parameter h   |
|-------------|:---------------:|
| Location    |     40000       |
| Days        |     7           |
| Time        |     12          |
"
cat(tabl)
```

Variables	Parameter h
Location	40000
Days	7
Time	12

The gaussian kernel is used with the following equation,

$k(u) = exp(-||u||^2)$ where $||.||$ is the Euclidean norm and the kernel function $k(\frac{x-x'}{h})$ is used.

The date for prediction is chosen as 30th may and the distance of our location as 40000 (40 kms) considering nearby stations in Sweden. The reason for choosing 7 days is considering a week (having 7 days) for forecasting. And also 12 hours of time is chosen by considering half a day for weather forecasting. The variance will be large with low bias if the smoothing parameter is too small. This may overfit the prediction. The kernel widths chosen here is sensible since it gives more weight to the closer points. And practically these values makes sense.

The resulting temperature values seems to be positive for both summed up and multiplicative kernels. Both kernels differ because, if one width becomes zero, it will affect the multiplicative kernel much than the summed up kernel since, in multiplication, the term becomes zero. It will affect the result. Actually, the large variation is noticed when some random values are given to h-values. So, appropriate h parameters should be selected for proper prediction.

2. SUPPORT VECTOR MACHINES

2.1 Performing model selection (non linear SVM)

```
####################### Assignment 2 ##########################
library(kernlab)
set.seed(12345)
data(spam)

# Dividing into training and test sets
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,]

# Model with the rbf kernel with a width of 0.05 for different C values

#sigma inverse kernel width for the Radial Basis kernel function "rbfdot"
fit1 <- ksvm(type~., data = train, kernel = "rbfdot", kpar=list(sigma=0.05), C = 0.5)
fit2 <- ksvm(type~., data = train, kernel = "rbfdot", kpar=list(sigma=0.05), C = 1)
fit3 <- ksvm(type~., data = train, kernel = "rbfdot", kpar=list(sigma=0.05), C = 5)


# predict with valid data for c=0.5
pred1 <- predict(fit1, valid, type="response")
matrixx1 <- table(valid$type, pred1)
paste("Table for model 1")
```

```
## [1] "Table for model 1"
```

```
matrixx1
```

```
##          pred1
##           nonspam spam
##   nonspam     658   25
##   spam         76  391
```

```
err_rate1 <- 1 - sum(diag(matrixx1))/sum(matrixx1)
paste("Test error rate when C = 0.5 : ", err_rate1)
```

```
## [1] "Test error rate when C = 0.5 :  0.0878260869565217"
```

```
# predict with valid data for c=1
pred2 <- predict(fit2, valid, type="response")
matrixx2 <- table(valid$type, pred2)
paste("Table for model 2")
```

```
## [1] "Table for model 2"
```

```
matrixx2
```

```
##          pred2
##           nonspam spam
##   nonspam     655   28
##   spam         61  406
```

```
err_rate2 <- 1 - sum(diag(matrixx2))/sum(matrixx2)
paste("Test error rate when C = 1 : ", err_rate2)
```

```
## [1] "Test error rate when C = 1 :  0.077391304347826"
```

```
# predict with valid data for c=5
pred3 <- predict(fit3, valid, type="response")
matrixx3 <- table(valid$type, pred3)
paste("Table for model 3")
```

```
## [1] "Table for model 3"
```

```
matrixx3
```

```
##          pred3
##           nonspam spam
##   nonspam     653   30
##   spam         64  403
```

```
err_rate3 <- 1 - sum(diag(matrixx3))/sum(matrixx3)
paste("Test error rate when C = 5 : ", err_rate3)
```

```
## [1] "Test error rate when C = 5 :  0.0817391304347826"
```

The spam data is divided as train (50%), valid (25%), test (25%) and the model is trained with inverse kernel width by specifying *sigma=0.05*. From the confusion matrix, it is observed that out of 467 spam emails, 406 emails are correctly classified as spam and out of 683, 655 emails are correctly classified as nonspam emails for model with C=1. And this count is greater than other two models. Also the misclassification error rate for this fit is 0.077 again which is lower than the other two models with C=0.5 and C=5. Hence, from these results, I conclude that the model with C=1 is the best fit among all the three models.

2.2 Estimating the generalization error of the SVM selected (*fit2*)

```
# 2.2
newtrain<-rbind(train,valid)
newfit <- ksvm(type~., data = newtrain, kernel = "rbfdot", kpar=list(sigma=0.05), C = 1)
newpred <- predict(newfit, test, type="response")
newmatrixx <- table(test$type, newpred)
paste("Table for model with C=1")
```

```
## [1] "Table for model with C=1"
```

```
newmatrixx
```

```
##          newpred
##           nonspam spam
##   nonspam     668   26
##   spam         57  400
```

```
err_rate <- 1 - sum(diag(newmatrixx))/sum(newmatrixx)
paste("Generalization error when C = 1 : ", err_rate)
```

```
## [1] "Generalization error when C = 1 :  0.0721112076455256"
```

The spam data is now divided as train (75%) and test (25%) and the model is trained with train data. Now, out of 457 emails, 400 are correctly classified as spam emails. The count for classification as nonspam emails has been increased when compared to the previous ones. The accuracy of this model with error rate 0.072 is higher than the previous models.

2.3 Produce the SVM that will be returned to the user

The best fit is model with C=1 that is estimated in step 2.2.

2.4 Parameter C purpose in SVM

For SVM, these 2 things are mostly required:

1. Setting a larger margin
2. getting low misclassification rate (how much a model misqualifies a data)

C is the parameter for the soft margin cost function, which controls the influence of each individual support vector. In other words C behaves as a regularization parameter in the SVM.

Large Value of parameter C => small margin (gives us low bias and high variance)

Small Value of parameter C => large margin (gives us high bias and low variance)

The decision function with a lower value of C favor the models in such a way that they use less memory and that are faster to predict. The low bias is because we penalize the cost of missclasification a lot. And also, the C can be used to alter the "sensitivity" of the algorithm.

Appendix

ref.label=knitr::all_labels(), echo=TRUE, eval=FALSE