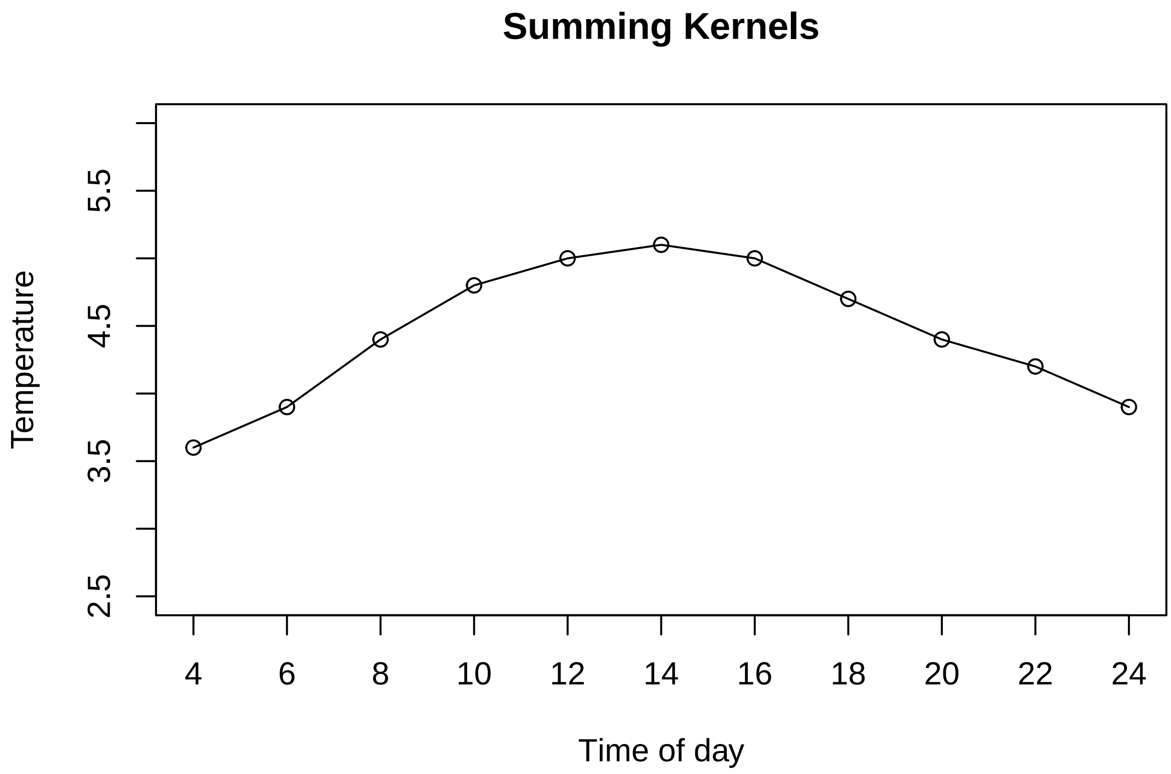
Lab3 Block 1

Ahmed Alhasan

12/16/2019

1. KERNEL METHODS:

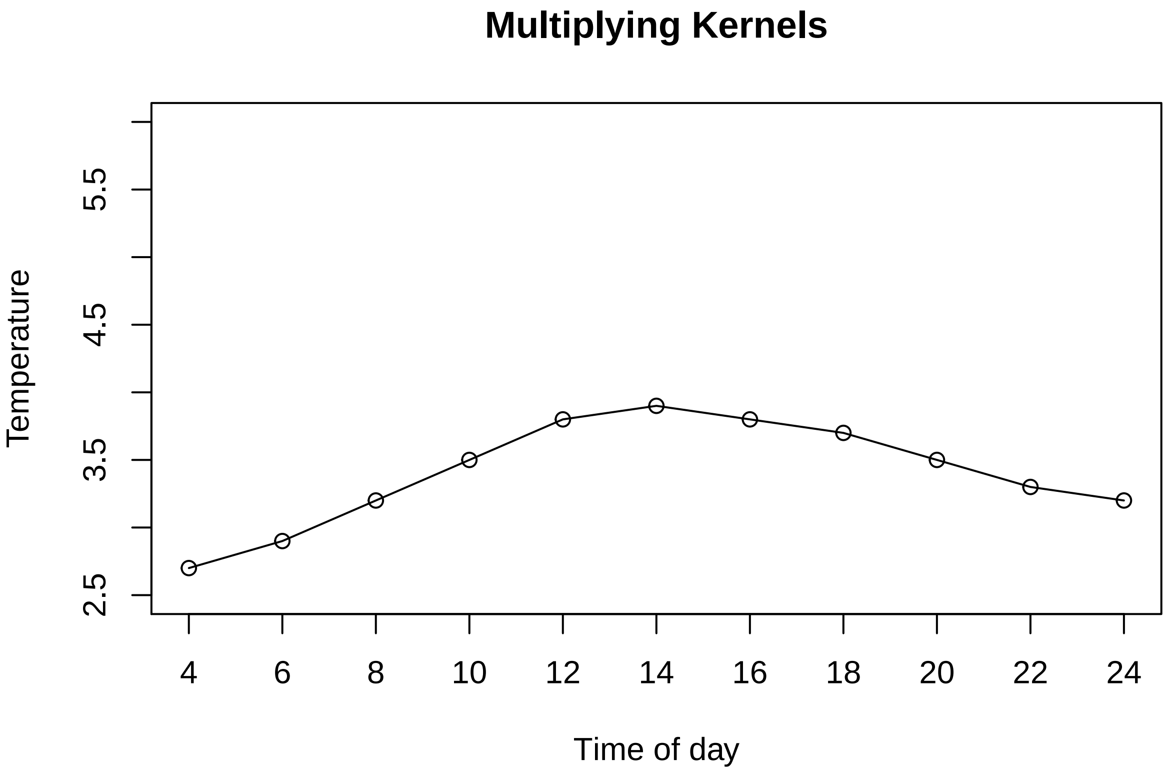
Summing	Multiplying
3.6	2.7
3.9	2.9
4.4	3.2
4.8	3.5
5.0	3.8
5.1	3.9
5.0	3.8
4.7	3.7
4.4	3.5
4.2	3.3
3.9	3.2



Show that your choice for the kernels' width is sensible, i.e. that it gives more weight to closer points. Discuss why your of definition closeness is reasonable.

- h = 10000, is the smoothing factor for the distance between stations which is 100 squared to adjust for the squared distance, meaning we are weighting the temperature every 100km which is reasonable range for the temperature to change.
- h = 900, is the smoothing factor for the difference between dates which is 30 days squared, weighting the temperature on 30 days basis.
- h = 36, is the smoothing factor for the difference between hours which is 6 hours squared, following the same concept above.
- Other things that can be noticed is that, the date and time follow cyclical trend and values should be adjusted before calculating the distance otherwise for example January and Decemeber will be very far apart and the same thing for 11PM and 1AM, howver due to lack of time adjustment is not made.
- In the selected times we have 24:00:00 while in the data set it is 00:00:00 this also could be changed but it will not matter much as long as the point above is not fixed. Because of this the predicted tempretures have a great variance at 24:00:00.

Instead of combining the three kernels into one by summing them up, multiply them. Compare the results obtained in both cases and elaborate on why they may differ.



- Multiplying is more sensetive to changes in h values, because every kernel is affected by the others, which means choosing a higher value of h for one kernel can inflate the other kernels and require more tuning. In summation each kernel is not affected by the others so we can get reasonable results only from one kernel even if the other two perform poorly.

2. SUPPORT VECTOR MACHINES

Model Selection

Training Data

Table 2: C = 0.5

Error	FPR	TPR
0.056	0.023	0.891

Table 3: C = 1

Error	FPR	TPR
0.04	0.021	0.93

Table 4: C = 5

Error	FPR	TPR
0.021	0.011	0.963

Validation Data

Table 5: C = 0.5

Error	FPR	TPR
0.086	0.033	0.832

Table 6: C = 1

Error	FPR	TPR
0.07	0.035	0.874

Table 7: C = 5

Error	FPR	TPR
0.075	0.037	0.865

- From validation we can see that the model with C = 1 is the best model since error rate and FPR are the lowest and TPR is the highest, C = 0.5 is under fitted and C = 5 is overfitted.

2.2 Generalization Error

Table 8: C = 1

Error	FPR	TPR
0.088	0.046	0.854

2.3 Selected SVM

```
## 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 : 1007
##
## Objective Function Value : -467.7423
## Training error : 0.04
```

2.4 Purpose of Parameter C

- The C value controls the margins between support vectors and the classification line, this cost parameter penalizes large residuals. So a larger cost will result in a more flexible model with fewer misclassifications. In effect the cost parameter allows to adjust the bias/variance trade-off. The greater the cost parameter, the more variance in the model and the less bias.

Appendix

```r
RNGversion("3.5.1")
library(geosphere)

set.seed(1234567890)

stations <- read.csv("Data/stations.csv",fileEncoding = "Latin1")
temps    <- read.csv("Data/temps50k.csv")
st       <- merge(stations,temps,by="station_number")
st$time  <- as.POSIXct(st$time, format="%H:%M:%S")


h_distance <- 10000
h_date     <- 900
h_time     <- 36
```

```r
a <- 14.826
b <- 58.4274
station_poi <- c(a,b)

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")
times <- as.POSIXct(times, format="%H:%M:%S")

temp           <- matrix(0,length(times),2)
colnames(temp) <- c("Summing", "Multiplying")

k_station <- function(obs, poi)
{
  dist <- abs(distHaversine(obs, poi) / 1000)
  k    <- exp(-(dist^2)/h_distance)
  return(k)
}
k1 <- k_station(st[,c("longitude","latitude")], station_poi)


#Re-arranging the date data so only months and days is considered
dates <- as.POSIXct(st$date, format="%Y-%m-%d")
mons  <- as.numeric(format(dates,"%m"))
days  <- as.numeric(format(dates,"%d"))
dates <- cbind(mons, days)

date <- "2017-11-03"
date <- as.POSIXct(date)
mon  <- as.numeric(format(date,"%m"))
day  <- as.numeric(format(date,"%d"))
date <- cbind(mon, day)

k_date <- function(obs, poi)
{
  diff <- (abs(obs[,1] - poi[1]) * 30) + abs(obs[,2] - poi[2])
  k    <- exp(-(diff^2)/h_date)
  return(k)
}
k2 <- k_date(dates, date)


k_time <- function(obs, poi)
{
  diff <- abs(as.numeric(difftime(obs, poi, unit = "hours")))
  k    <- exp(-(diff^2)/h_time)
  return(k)
}
k3 <- matrix(0,50000,11)
for(j in 1:length(times)){
  k3[,j] <- k_time(st$time, times[j])
}
```

```
for(j in 1:length(times)){
  temp[j,1] <- sum(k1 * st$air_temperature + k2 * st$air_temperature + k3[,j] * st$air_temperature)
  / sum(k1 + k2 + k3[,j])
  temp[j,2] <- sum(k1 * k2 * k3[,j] * st$air_temperature) / sum(k1 * k2 * k3[,j])
}

temp <- round(temp, 1)
knitr::kable(temp)

plot (temp[,1],
      type ="o",
      xaxt = "n",
      xlab ="Time of day",
      ylab = "Temperature",
      main = "Summing Kernels",
      ylim = c(2.5,6))
axis (1, at =1:11, labels = seq (04 ,24 ,2))

plot (temp[,2],
      type ="o",
      xaxt = "n",
      xlab ="Time of day",
      ylab = "Temperature",
      main = "Multiplying Kernels",
      ylim = c(2.5,6))
axis (1, at =1:11, labels = seq (04 ,24 ,2))

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*0.3))
valid <- spam[id2,]

id3   <- setdiff(id1,id2)
test  <- spam[id3,]


svm <- function(data, c){
  model     <- ksvm(type ~ .,
                    data   = train,
                    type   = "C-svc",
                    kernel = "rbfdot",
                    kpar   = list(sigma = 0.05),
                    C      = c)

  pred      <- predict(model, newdata = data)
```

```r
  con_mat   <- table("Predictions" = pred, "Actuals" = data$type)
  miss_rate <- 1 - sum(diag(con_mat)) / sum(con_mat)

  TN <- con_mat[1,1]
  TP <- con_mat[2,2]
  FN <- con_mat[1,2]
  FP <- con_mat[2,1]
  TPR <- TP / (TP + FN)
  FPR <- FP / (FP + TN)

  res <- round(cbind(miss_rate, FPR, TPR), 3)
  colnames(res) <- c("Error", "FPR", "TPR")

  if(c == 0.5){
    return(knitr::kable(res, caption = "C = 0.5"))
  }

  if(c == 1){
    return(knitr::kable(res, caption = "C = 1"))
  }

  if(c == 5){
    return(knitr::kable(res, caption = "C = 5"))
  }
}

svm(train, c = 0.5)
svm(train, c = 1)
svm(train, c = 5)

svm(valid, c = 0.5)
svm(valid, c = 1)
svm(valid, c = 5)

##2.2
svm(test, c = 1)

##2.3
print(model     <- ksvm(type ~ .,
                        data   = train,
                        type   = "C-svc",
                        kernel = "rbfdot",
                        kpar   = list(sigma = 0.05),
                        C      = 1))
```