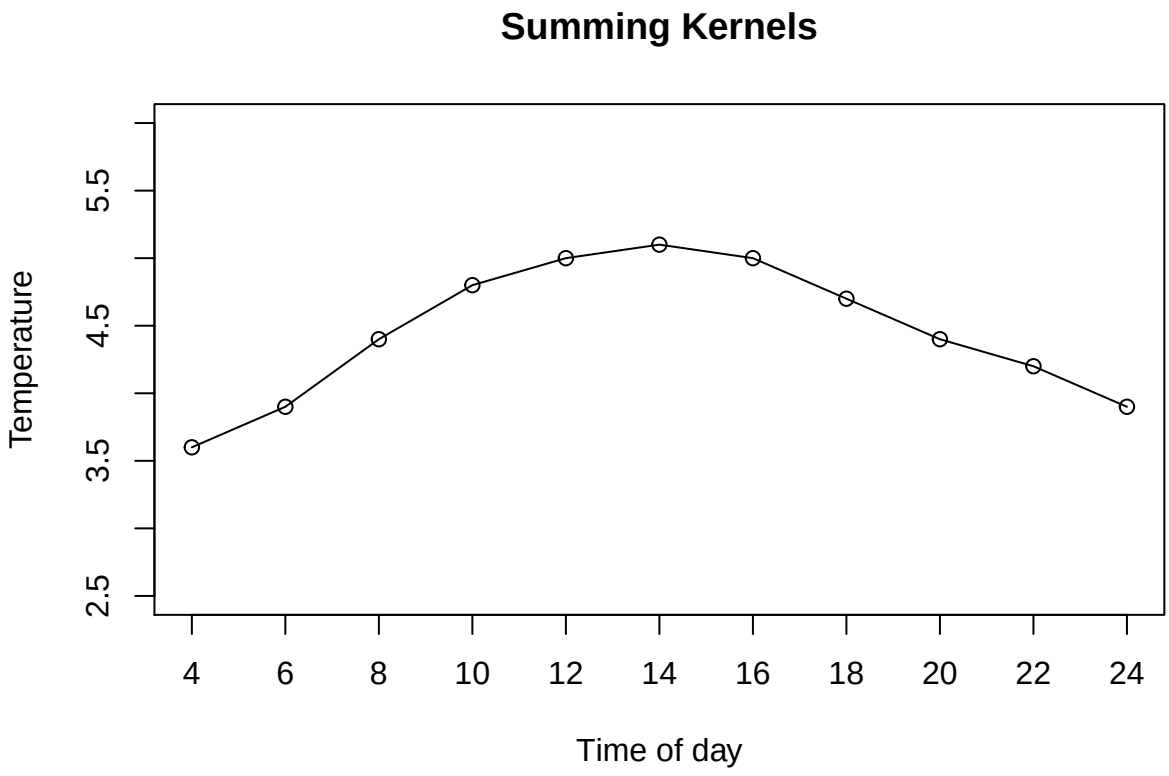
Lab 3 Block 2

Group A05

12/17/2019

1. KERNEL METHODS:

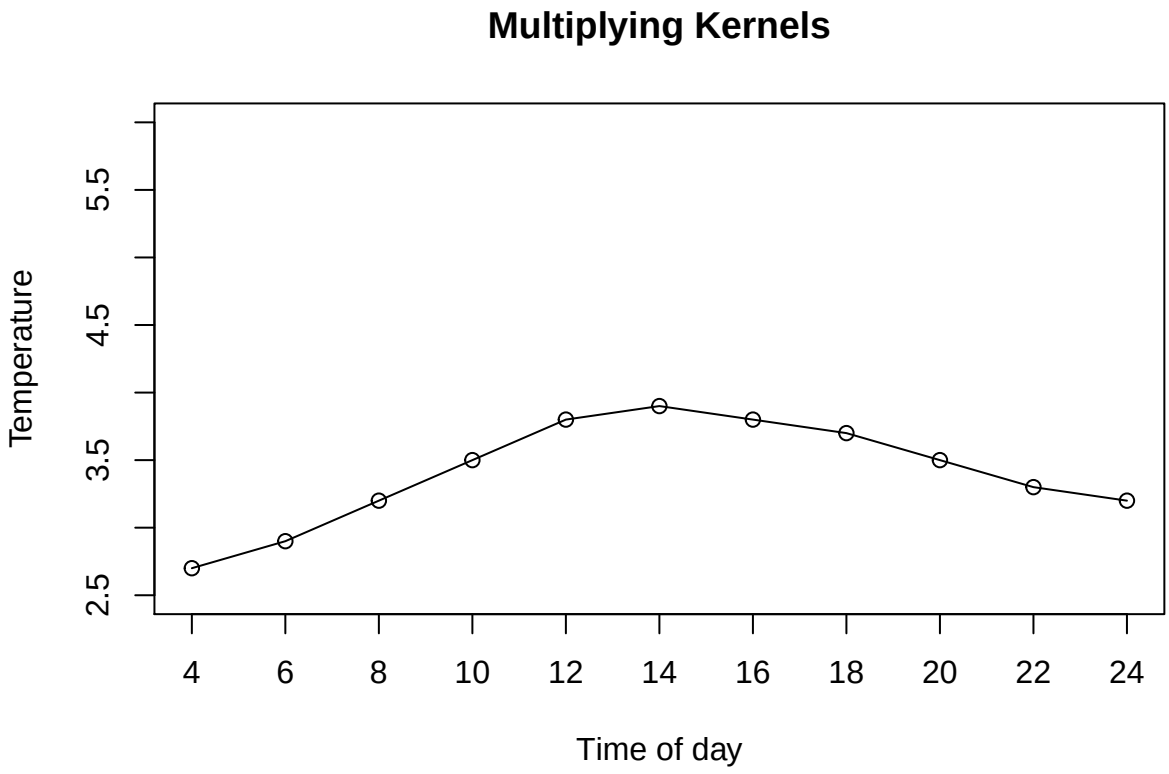
Summing	Multiplying
3.6	2.7
3.9	2.9
4.4	3.2
4.8	3.5
5.0	3.8
5.1	3.9
5.0	3.8
4.7	3.7
4.4	3.5
4.2	3.3
3.9	3.2



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

Use the function ksvm from the R package kernlab to learn a SVM for classifying the spam dataset that is included with the package. Consider the radial basis function kernel (also known as Gaussian) with a width of 0.05. For the C parameter, consider values 0.5, 1 and 5. This implies that you have to consider three models.

** Perform model selection, i.e. select the most promising of the three models (use any method of your choice except cross-validation or nested cross-validation).* **Estimate the generalization error of the SVM selected above (use any method of your choice except cross-validation or nested cross-validation).** Produce the SVM that will be returned to the user, i.e. show the code. ** What is the purpose of the parameter C ?

```
## Confusion Matrix and Statistics
##
##          Y hat
## Y         nonspam spam
##   nonspam     672   22
##   spam         86  371
##
##                Accuracy : 0.9062
##                  95% CI : (0.8878, 0.9224)
##     No Information Rate : 0.6586
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7992
##
##  Mcnemar's Test P-Value : 1.343e-09
##
##             Sensitivity : 0.8865
##             Specificity : 0.9440
##          Pos Pred Value : 0.9683
##          Neg Pred Value : 0.8118
##              Prevalence : 0.6586
##          Detection Rate : 0.5838
##    Detection Prevalence : 0.6030
##       Balanced Accuracy : 0.9153
##
##        'Positive' Class : nonspam
##
```

```
## Confusion Matrix and Statistics
##
##          Y hat
## Y         nonspam spam
##   nonspam     668   26
##   spam         63  394
##
##                Accuracy : 0.9227
##                  95% CI : (0.9057, 0.9374)
##     No Information Rate : 0.6351
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8362
##
##  Mcnemar's Test P-Value : 0.0001356
##
##             Sensitivity : 0.9138
##             Specificity : 0.9381
##          Pos Pred Value : 0.9625
##          Neg Pred Value : 0.8621
##              Prevalence : 0.6351
##          Detection Rate : 0.5804
##    Detection Prevalence : 0.6030
##       Balanced Accuracy : 0.9260
##
##        'Positive' Class : nonspam
##

## Confusion Matrix and Statistics
##
##          Y hat
## Y         nonspam spam
##   nonspam     666   28
##   spam         69  388
##
##                Accuracy : 0.9157
##                  95% CI : (0.8982, 0.9311)
##     No Information Rate : 0.6386
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8213
##
##  Mcnemar's Test P-Value : 4.878e-05
##
##             Sensitivity : 0.9061
##             Specificity : 0.9327
##          Pos Pred Value : 0.9597
##          Neg Pred Value : 0.8490
##              Prevalence : 0.6386
##          Detection Rate : 0.5786
##    Detection Prevalence : 0.6030
##       Balanced Accuracy : 0.9194
##
##        'Positive' Class : nonspam
##
```

Conslusions:

We split the data into train, validation and test data (50, 25, 25). The idea is that we train three models with varying values of hyperparameter C (0.5, 1.0, 5.0). After training all models it is seen from the output above that the best model is obtained when C = 1.0. This will be used below as the final svm model.

```
# choosing the best model with C=1.0, train with full dataset.
train.test = rbind(train, test)
```

```
final_svm_model = ksvm(type~ ., data=train.test,
               kernel="rbfdot",
               kpar=list(sigma=0.05),
               C=1.0)
final_confmat = table(valid[,58], predict(final_svm_model, valid[,-58]))
names(dimnames(final_confmat)) = c("Y", "Y hat")
caret::confusionMatrix(final_confmat)
```

```
## Confusion Matrix and Statistics
##
##          Y hat
## Y         nonspam spam
##   nonspam     653   30
##   spam         54  413
##
##                Accuracy : 0.927
##                  95% CI : (0.9104, 0.9413)
##     No Information Rate : 0.6148
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8473
##
##  Mcnemar's Test P-Value : 0.01209
##
##             Sensitivity : 0.9236
##             Specificity : 0.9323
##          Pos Pred Value : 0.9561
##          Neg Pred Value : 0.8844
##              Prevalence : 0.6148
##          Detection Rate : 0.5678
##    Detection Prevalence : 0.5939
##       Balanced Accuracy : 0.9280
##
##        'Positive' Class : nonspam
##
```

What is the pupose of C? The Purpose of hyperparameter C is to penalizes large residuals, it is effictivly a cost parameter which controls the relationship between bias and variance. A larger C value will give good accuracy on the training data set but because the margin between the classes is small the accuracy on the test data set will suffer. In other words, a larger C will contribute to overfitting. Subsequently a lower C value can lead to underfitting.

Appendix

```
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

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))


## Assignment 2
library("kernlab")
data("spam")

set.seed(12345)

# separate data to training, validation, and test - 50/25/25
```

```r
n = dim(spam)[1]
set.seed(12345)
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,]

## train a support vector machine models
svm_model0.5 = ksvm(type~., data=train,
                kernel="rbfdot",
                kpar=list(sigma=0.05),
                C=0.5)
svm_model1.0 = ksvm(type~.,data=train,
                kernel="rbfdot",
                kpar=list(sigma=0.05),
                C=1.0)
svm_model5.0 = ksvm(type~.,data=train,
                kernel="rbfdot",
                kpar=list(sigma=0.05),
                C=5.0)
# confusion table
confmat0.5 = table(test[,58], predict(svm_model0.5,test[,-58]))
names(dimnames(confmat0.5)) <- c("Y", "Y hat")
caret::confusionMatrix(confmat0.5)
confmat1.0 = table(test[,58], predict(svm_model1.0,test[,-58]))
names(dimnames(confmat1.0)) <- c("Y", "Y hat")
caret::confusionMatrix(confmat1.0)
confmat5.0 = table(test[,58], predict(svm_model5.0,test[,-58]))
names(dimnames(confmat5.0)) <- c("Y", "Y hat")
caret::confusionMatrix(confmat5.0)

# choosing the best model with C=1.0, train with full dataset.
train.test = rbind(train, test)
final_svm_model = ksvm(type~ ., data=train.test,
                kernel="rbfdot",
                kpar=list(sigma=0.05),
                C=1.0)
final_confmat = table(valid[,58], predict(final_svm_model, valid[,-58]))
names(dimnames(final_confmat)) = c("Y", "Y hat")
caret::confusionMatrix(final_confmat)
```