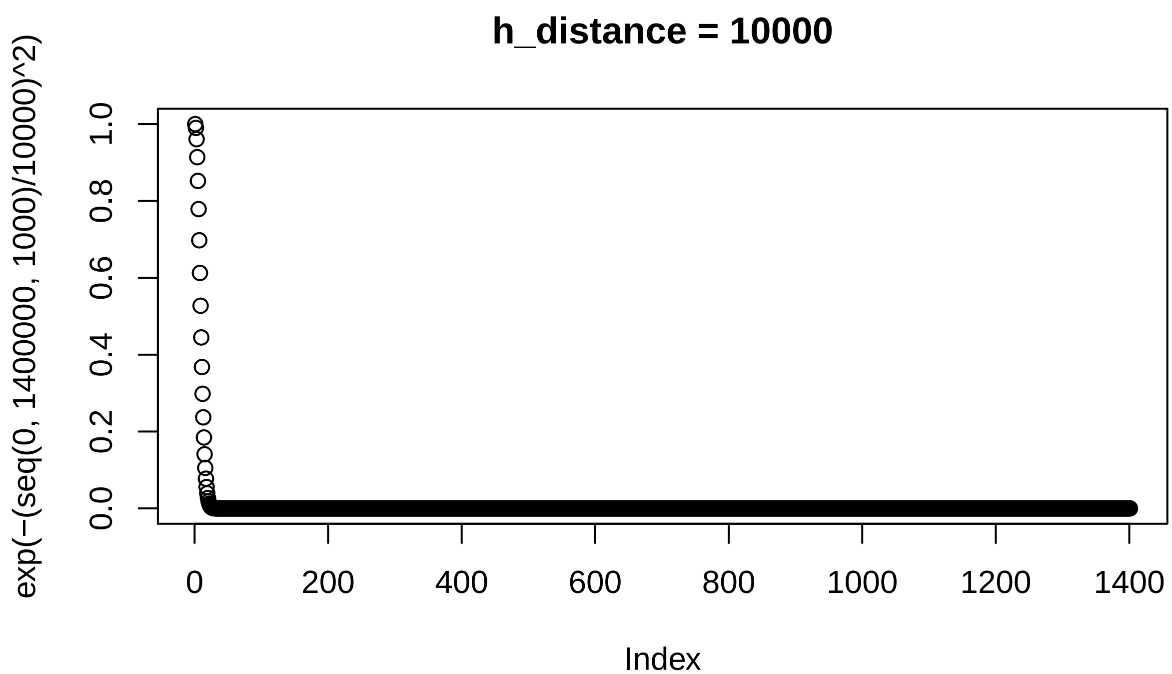
Machine Learning Lab3 Block 1

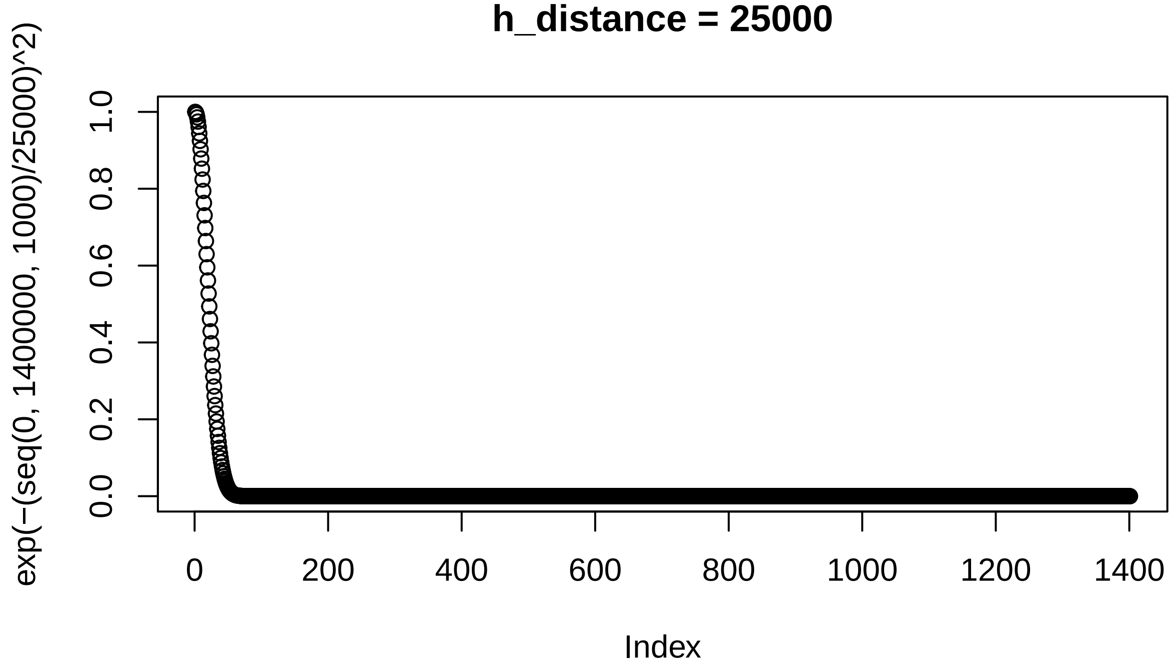
Karthikeyan Devarajan Karde799

1.Kernel Methods

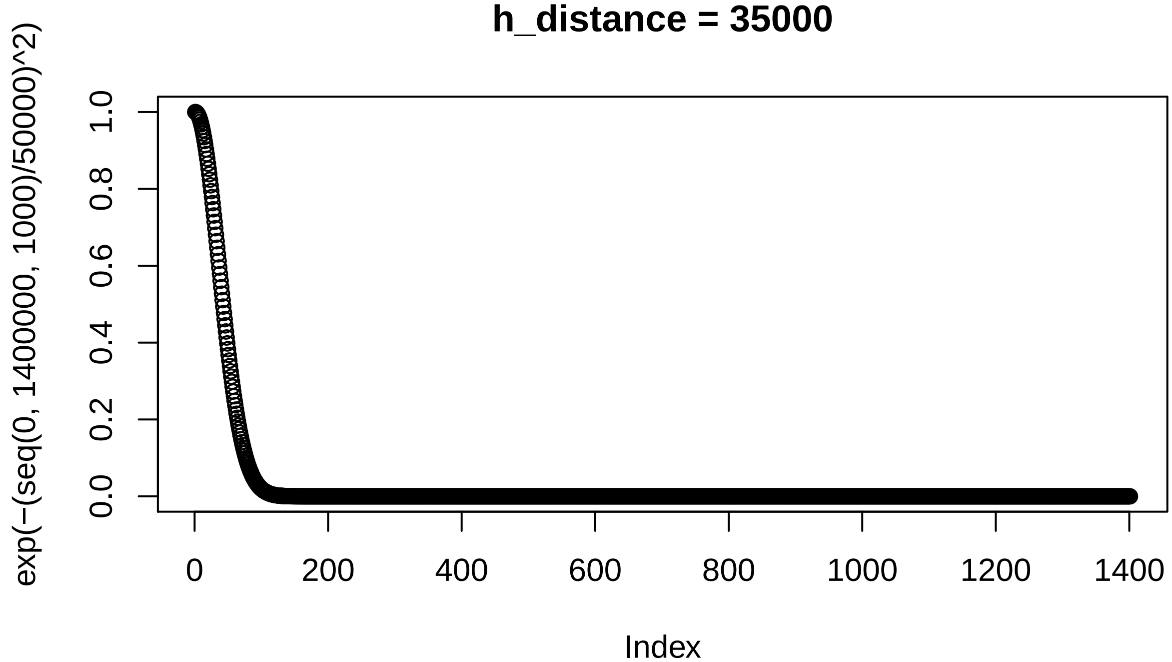
The kernal method function

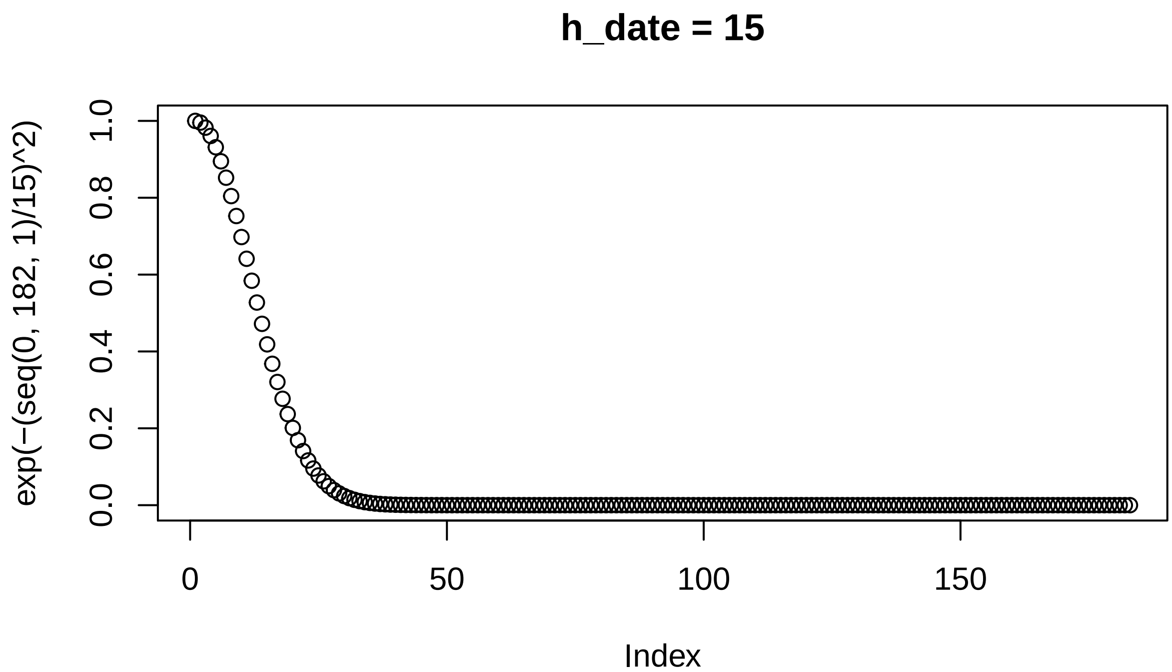


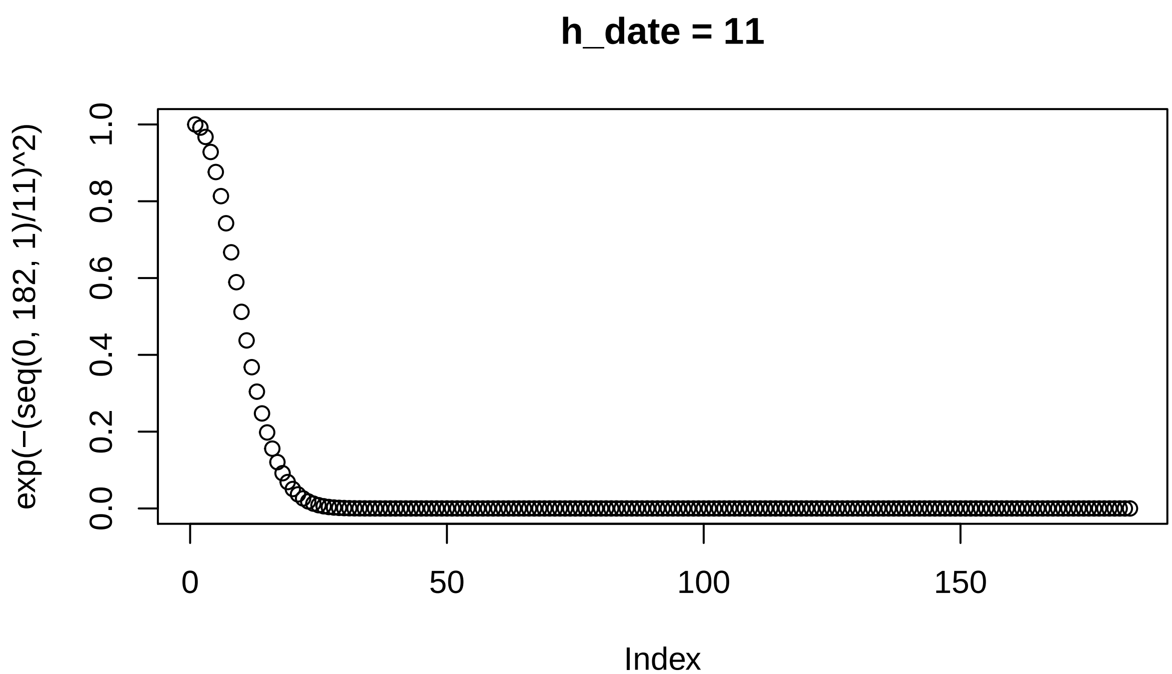
h_distance = 25000

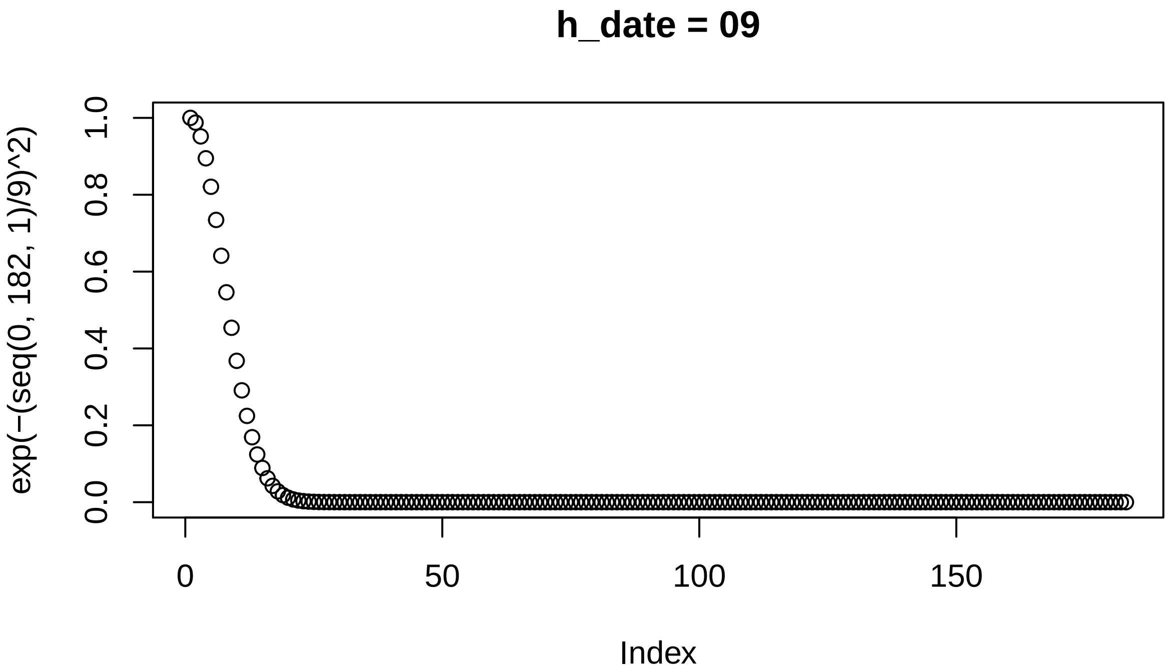


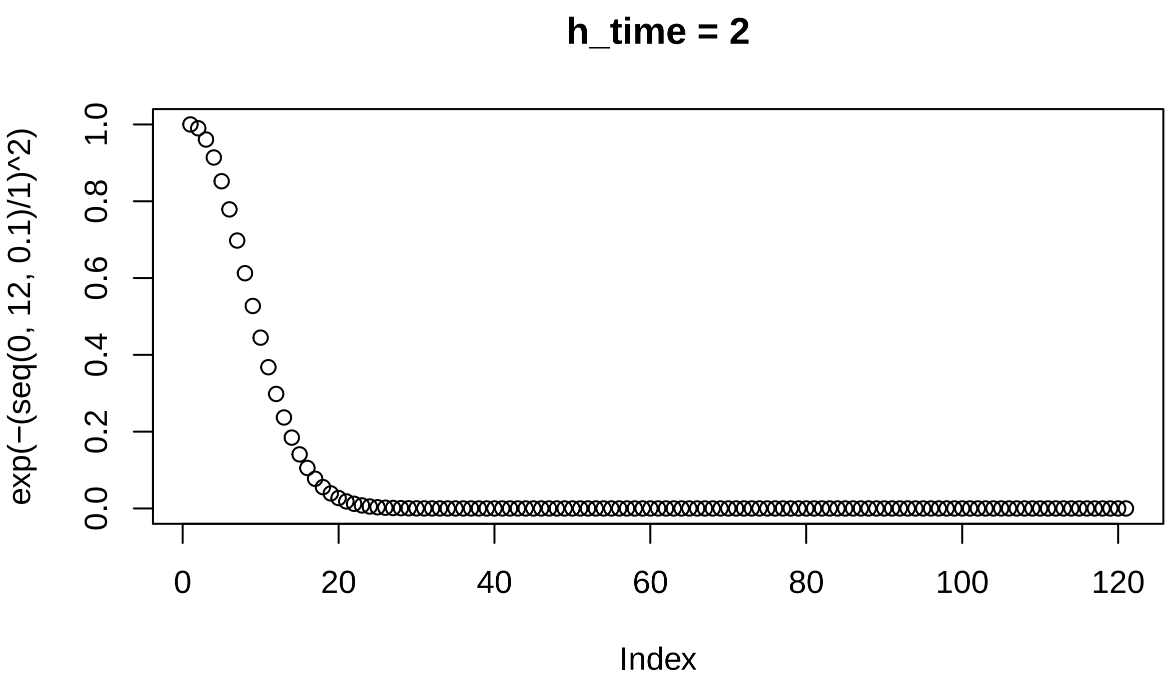
h_distance = 35000

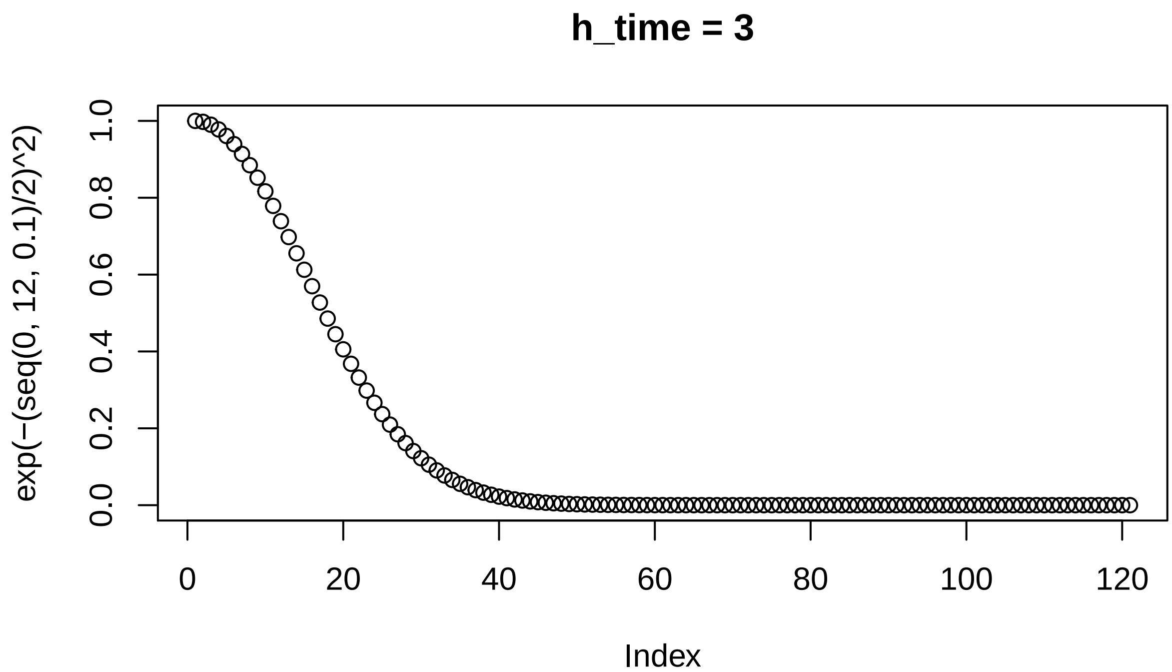


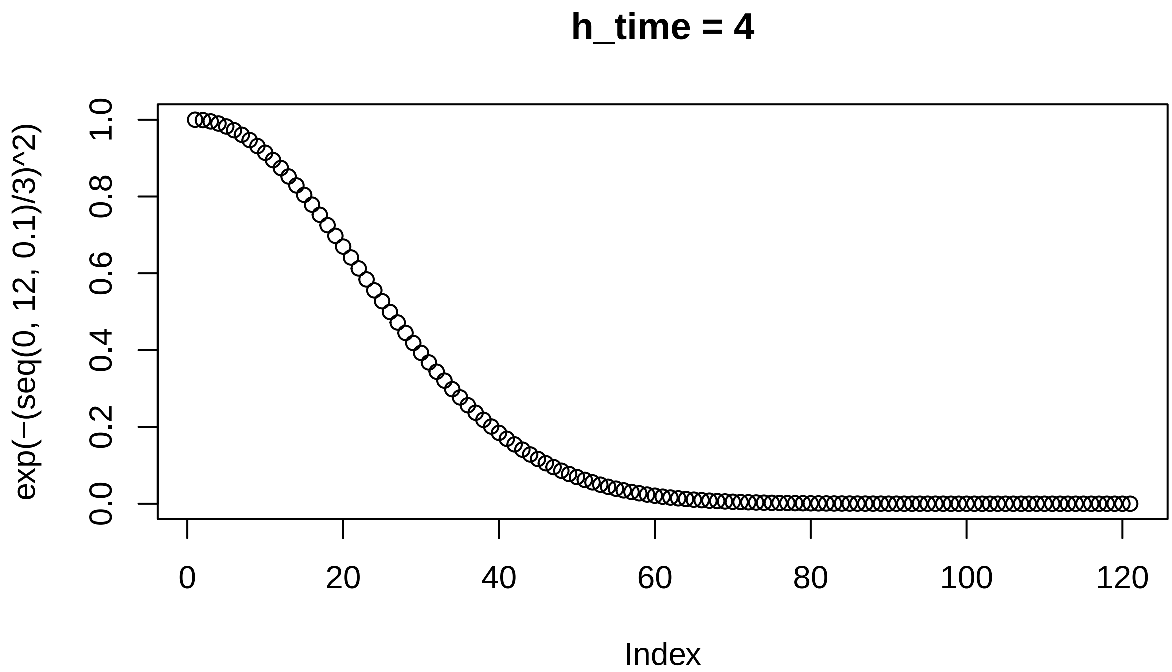

h_date = 15
exp(−(seq(0, 182, 1)/15)^2)
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
Index


h_date = 11
exp(−(seq(0, 182, 1)/11)^2)
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
Index

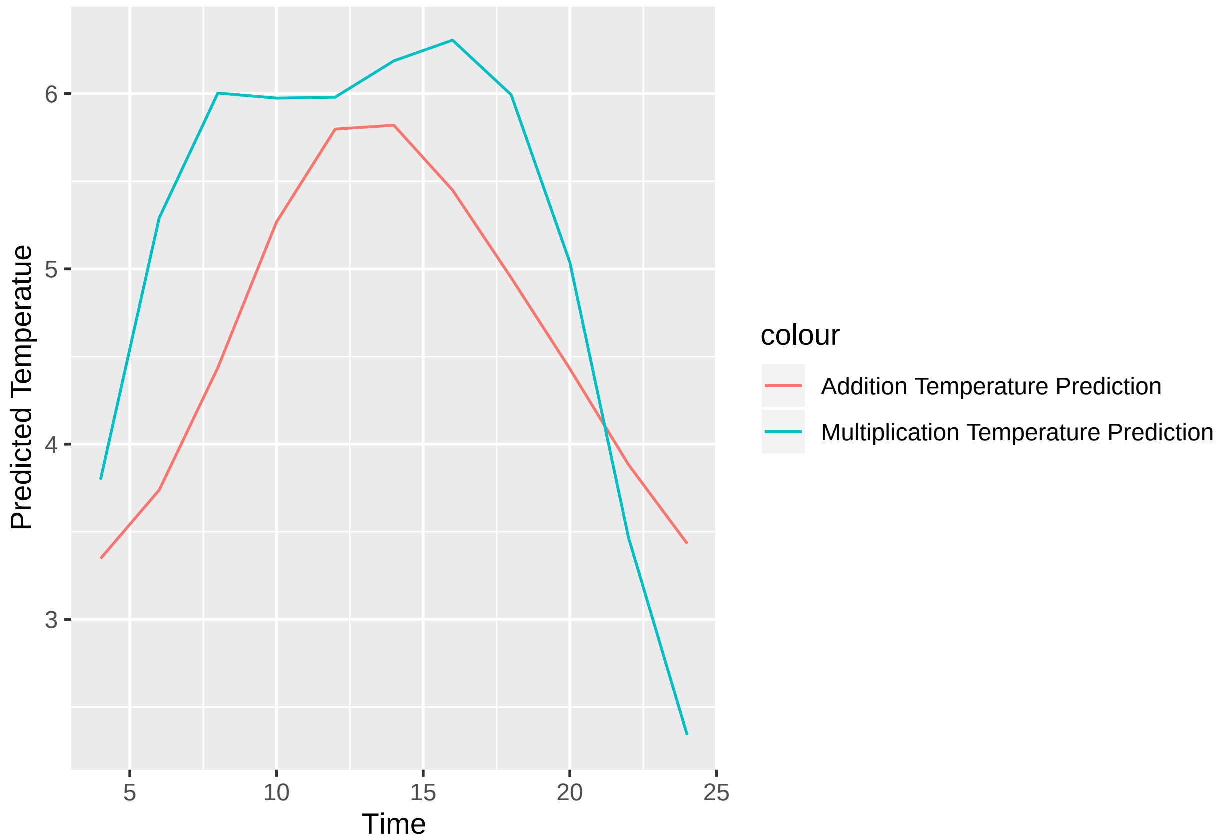

h_date = 09
exp(−(seq(0, 182, 1)/9)^2)
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
Index


h_time = 2
exp(−(seq(0, 12, 0.1)/1)^2)
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100
120
Index





The above graphs are plotted based on the gaussian kernal for h_distance,h_time,h_date. Based on the graph, the h_distance,h_time,h_date can be selected as 35000,4 and 15. This is because all the points in the respective graphs are closely plotted and close to mean.



For date kernel, the reasonable value for bandwidth h could be taken around 4. The h value here depends on the previous 4 hour value and the temperature would not increase or decrease within that range. For stations, the h_distance could be 35000(i.e 35km) which includes all points in the 35km. For h_date and h_time, the bandwidth are 15 and 4. These values include all points while plotting in kernel. The Multiple Kernel rises from low value to high value during morning and decreases after mid-day. It maintains for a few hours from morning to midday. whereas mulitiplication kernel is increases and then deceases without any gradual or holding points,which is unsual for temperature.The Multiplication kernal is multiplication of two number which will give a bigger or smaller value depending upon the input. This will lead to projection of a small change into account. In Addition Kernal, the small change cannot be counted.

Hence for this weather prediction task, Addition of Kernel functions is more suited than Multiplication of Kernel Functions.

2.Support Vector Mechanisms

Model Selection

Classifying into train/test - 50/25/25.

i)C = 0.5, kernel = rbfdot

```
## The confusion matrix is:

##          prediction1
##           nonspam spam
```

```
##   nonspam     695   25
##   spam         83  347
```

```
## Misclassification rate is: 0.09391304
```

ii)C = 1, kernel = rbfdot

```
## The confusion matrix is:
```

```
##          prediction2
##           nonspam spam
##   nonspam     690   30
##   spam         71  359
```

```
## Misclassification rate is: 0.08782609
```

iii)C = 5, kernel = rbfdot

```
## The confusion matrix is:
```

```
##          prediction3
##           nonspam spam
##   nonspam     688   32
##   spam         70  360
```

```
## Misclassification rate is: 0.08869565
```

From the above misclassification rates, C = 1 is producing small misclassification rate. So, for generalisation error we can use C = 1.

Generalisation error

```
## The confusion matrix is:
```

```
##          prediction4
##           nonspam spam
##   nonspam     658   24
##   spam         61  408
```

```
## Misclassification rate is: 0.07384883
```

When full data is used

```
## The confusion matrix is:
```

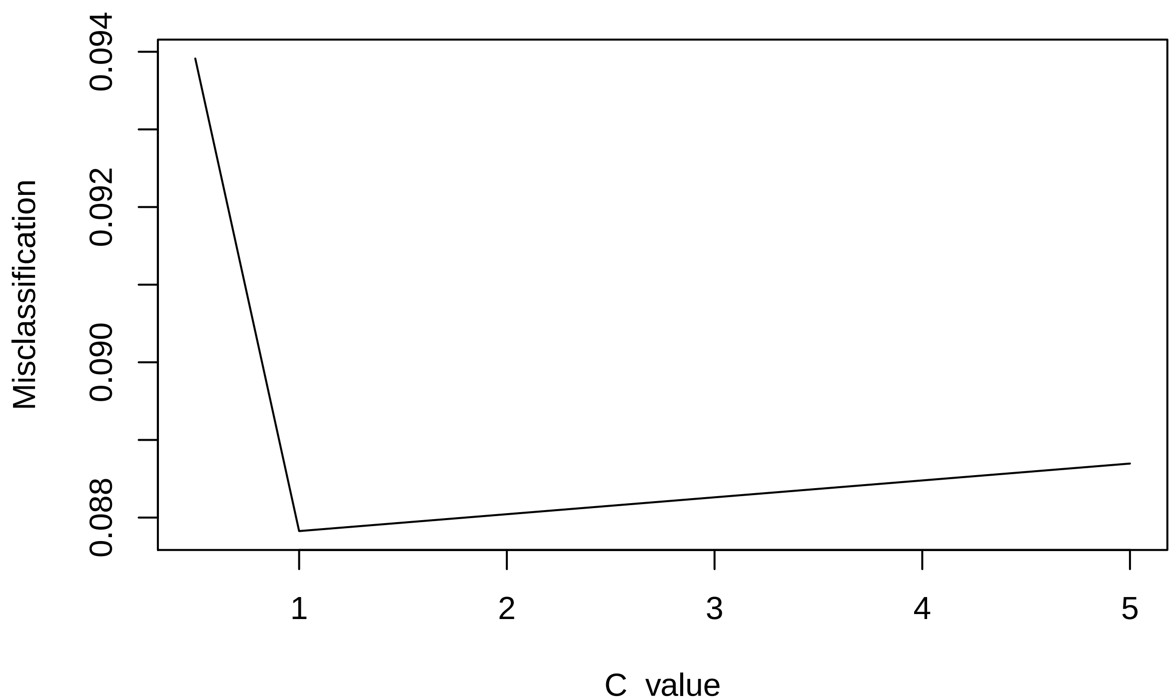
```
##          prediction_full
##           nonspam spam
##   nonspam    2724   64
##   spam        149 1664
```

```
## Misclassification rate is: 0.07384883
```

The generalization error and misclassification rate for whole data is similar.

Purpose of parameter

The C argument is known as cost of constraints.Here,Cost of Constraints means the how it influences the margin of the hyperplane.The C is 1 in default.It is constant of the regularization term in the Lagrange formulation. It will decide the width between the hyperplane. If the margin hyperplanes is small, then it will take all points into consideration properly. The margin hyperplane will be small when C value is large. The small C value will produce large margin hyperplane, which make the Residual mean error more. The Residual Mean square Error will be reducing a little bit when C is increasing.



Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(kernlab)
library(geosphere)
library(ggplot2)
stations = read.csv(file.choose())
temps = read.csv(file.choose())
data("spam")
sf1 <- spam
kernal_methods <- function(st,a,b,h_distance,date,h_date,times,h_time)
  {
  day_distance <- as.numeric(difftime(date, st$date, units = c("days")))
  criteria <- which(day_distance < 0)
```

```r
  day_distance <- day_distance[-criteria]
  day_distance <- day_distance %% 365
  for(i in 1:length(day_distance))
  {
    day_distance[i] <- min(365-day_distance[i],day_distance[i])
  }
  k_day <- exp(-(day_distance / h_date)^2)
  st <- st[-criteria,]

  station_distance <- abs(distHaversine(p1 = c(a,b), p2 = st[,c("latitude","longitude")]))
  k_dist <- exp(-(station_distance / h_distance)^2)


hr_distance <- matrix(data = NA, nrow =dim(st)[1], ncol = 11)

for(i in 1:length(times))
{

  hr_distance[,i] <- as.numeric(abs(difftime(strptime(times[i],"%H"),strptime(as.character(st[,"time"])
  hr_distance[,i] <- sapply(hr_distance[,i], function(x) min((24-x), x))
}

k_time <- exp(-(hr_distance / h_time)^2)
add_prediction <- matrix(NA,nrow=dim(st)[1],ncol=11)
for(i in 1:11)
{
add_prediction[,i] <- (k_dist + k_day) + k_time[,i]
}
colnames(add_prediction) <- c("04:00:00", "06:00:00","08:00:00","10:00:00","12:00:00","14:00:00","16:00
temp_pred_addition <- vector()
for(i in 1:11)
{
numerator_addition <- sum(add_prediction[,i] * st[,11])
denominator_addition <- sum(add_prediction[,i])
temp_pred_addition[i] <- numerator_addition/ denominator_addition
}
multiplication_prediction <- matrix(NA,nrow=dim(st)[1],ncol=11)
for(i in 1:11)
{
  multiplication_prediction[,i] <- (k_dist * k_day) * k_time[,i]
}
colnames(multiplication_prediction) <- c("04:00:00", "06:00:00","08:00:00","10:00:00","12:00:00","14:00

temp_pred_multiplication <- vector()
for(i in 1:11)
{
  numerator_multiplication <- sum(multiplication_prediction[,i] * st[,11])
  denominator_multiplication <- sum(multiplication_prediction[,i])
  temp_pred_multiplication[i] <- numerator_multiplication / denominator_multiplication
}

d <- data.frame(Add = as.matrix(temp_pred_addition), Mul = as.matrix(temp_pred_multiplication))
return(d)
```

```
}
plot(exp(-(seq(0,1400000,1000) / 10000)^2), main = "h_distance = 10000")
plot(exp(-(seq(0,1400000,1000) / 25000)^2), main = "h_distance = 25000")
plot(exp(-(seq(0,1400000,1000) / 50000)^2), main = "h_distance = 35000")
plot(exp(-(seq(0,182, 1) / 15)^2), main = "h_date = 15")
plot(exp(-(seq(0,182, 1) / 11)^2), main = "h_date = 11")
plot(exp(-(seq(0,182, 1) / 09)^2), main = "h_date = 09")
plot(exp(-(seq(0,12,0.1 ) / 1)^2), main = "h_time = 2")
plot(exp(-(seq(0,12,0.1 ) / 2)^2), main = "h_time = 3")
plot(exp(-(seq(0,12,0.1 ) / 3)^2), main = "h_time = 4")
set.seed(1234567890)
st <- merge(stations,temps,by="station_number")
h_distance <- 35000
h_date <- 15
h_time <- 4
a <- 54.245
b <- 14.854
date <- "2000-04-20"
time_seq <- 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","2
prediction  = kernal_methods(st = st, date = date, a=a, b=b, times = time_seq, h_date = h_date, h_time =
ggplot(prediction) + geom_line(aes(seq(4,24,2),Add, col="Addition Temperature Prediction")) + geom_line
n=dim(sf1)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=sf1[id,]
id1=setdiff(1:n,id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=sf1[id2,]
id3=setdiff(id1,id2)
test=sf1[id3,]
model1 <- ksvm(type~., data=train,kernel="rbfdot",kpar=list(sigma=0.05),C=0.5)
prediction1 <- predict(model1,newdata=valid,type="response")
confusion_matrix1 <- table(valid$type,prediction1)
cat("The confusion matrix is:\n")
confusion_matrix1
misclassification_rate1 <- 1-sum(diag(confusion_matrix1))/sum(confusion_matrix1)
cat("Misclassification rate is:",misclassification_rate1)
model2 <- ksvm(type~., data=train,kernel="rbfdot",kpar=list(sigma=0.05),C=1)
prediction2 <- predict(model2,newdata=valid,type="response")
confusion_matrix2 <- table(valid$type,prediction2)
cat("The confusion matrix is:\n")
confusion_matrix2
misclassification_rate2 <- 1-sum(diag(confusion_matrix2))/sum(confusion_matrix2)
cat("Misclassification rate is:",misclassification_rate2)
model3 <- ksvm(type~., data=train,kernel="rbfdot",kpar=list(sigma=0.05),C=5)
prediction3 <- predict(model3,newdata=valid,type="response")
confusion_matrix3 <- table(valid$type,prediction3)
cat("The confusion matrix is:\n")
confusion_matrix3
misclassification_rate3 <- 1-sum(diag(confusion_matrix3))/sum(confusion_matrix3)
cat("Misclassification rate is:",misclassification_rate3)
new_train <- rbind(train,valid)
```

```
model4 <- ksvm(type~., data=new_train,kernel="rbfdot",kpar=list(sigma=0.05),C=1)
prediction4 <- predict(model4,newdata=test,type="response")
confusion_matrix4 <- table(test$type,prediction4)
cat("The confusion matrix is:\n")
confusion_matrix4
misclassification_rate4 <- 1- sum(diag(confusion_matrix4))/sum(confusion_matrix4)
cat("Misclassification rate is:",misclassification_rate4)
model_full <- ksvm(type~., data=sf1,kernel="rbfdot",kpar=list(sigma=0.05),C=1)
prediction_full <- predict(model4,newdata=sf1,type="response")
confusion_matrix_full <- table(sf1$type,prediction_full)
cat("The confusion matrix is:\n")
confusion_matrix_full
misclassification_rate_full <- 1- sum(diag(confusion_matrix_full))/sum(confusion_matrix_full)
cat("Misclassification rate is:",misclassification_rate4)
misclassi <- data.frame(C_value = c(0.5,1,5),Misclassification = c(misclassification_rate1,misclassifica
plot(misclassi,type = 'l')
```