

Lab3

Andreas C Charitos -andch552

15 Dec 2018

Contents

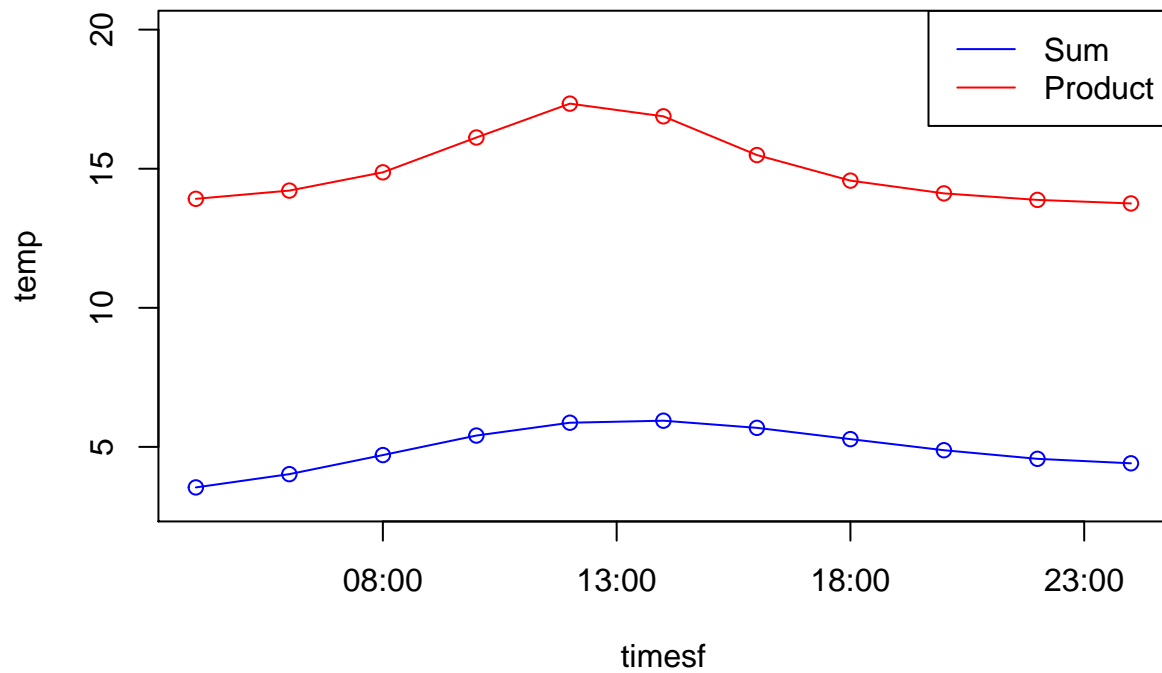
Assignment 1	1
Assignment 2	5
Appendix	6

Assignment 1

Implementation 1

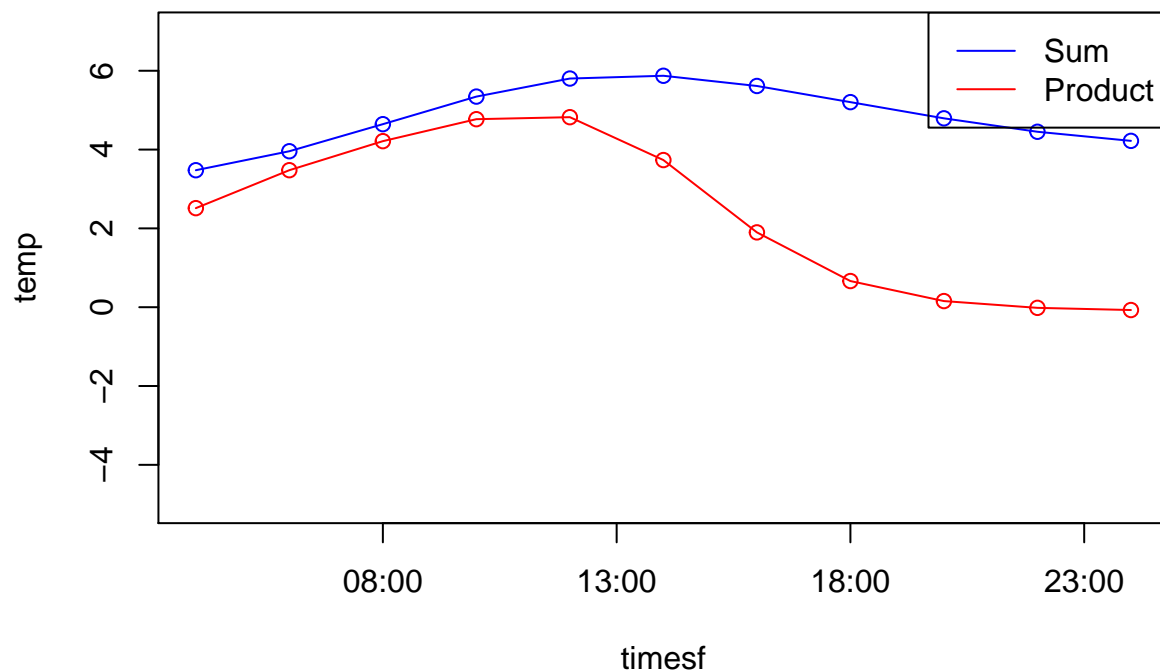
In this imlemenation we choose to take into account the difference in years between the dates so the temperature prediction is highly affected by the diffience in year our prediction date has with the obsereved dates on our data set.

Linkoping '2013-08-23' Sum kernel



The above plot shows the temperature predictions for Linkoping for the date of '2013-08-23' with the summation and the multiplication kernel.

Linköping '2013-08-23' Sum kernel

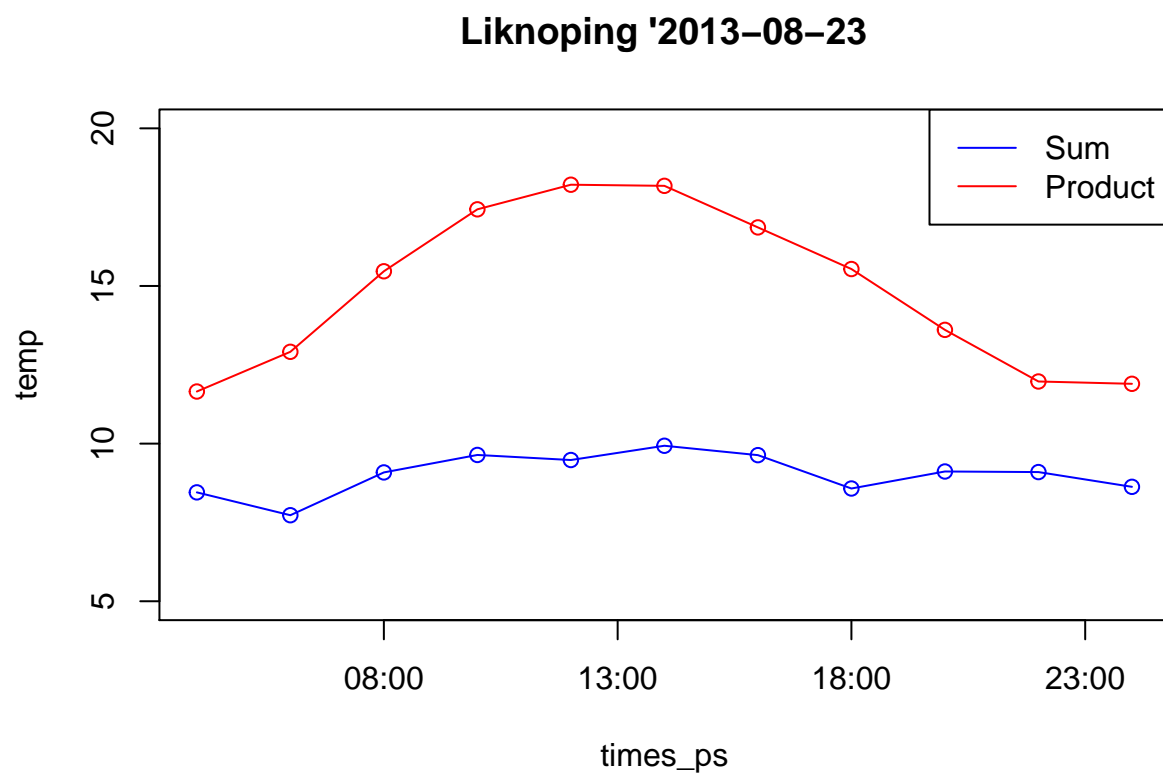


The above plot shows the temperature predictions for Linköping for the date of '2013-12-23' with the summation and the multiplication kernel.

From the 2 plots we can see that the predictions of the multiplication kernel seem to be reasonable for the 2 dates but for the summation kernel we can see that we basically achieve approximately the same predictions. The h values we used were $(20000, 20, 5)$ for (distance, days, time) respectively.

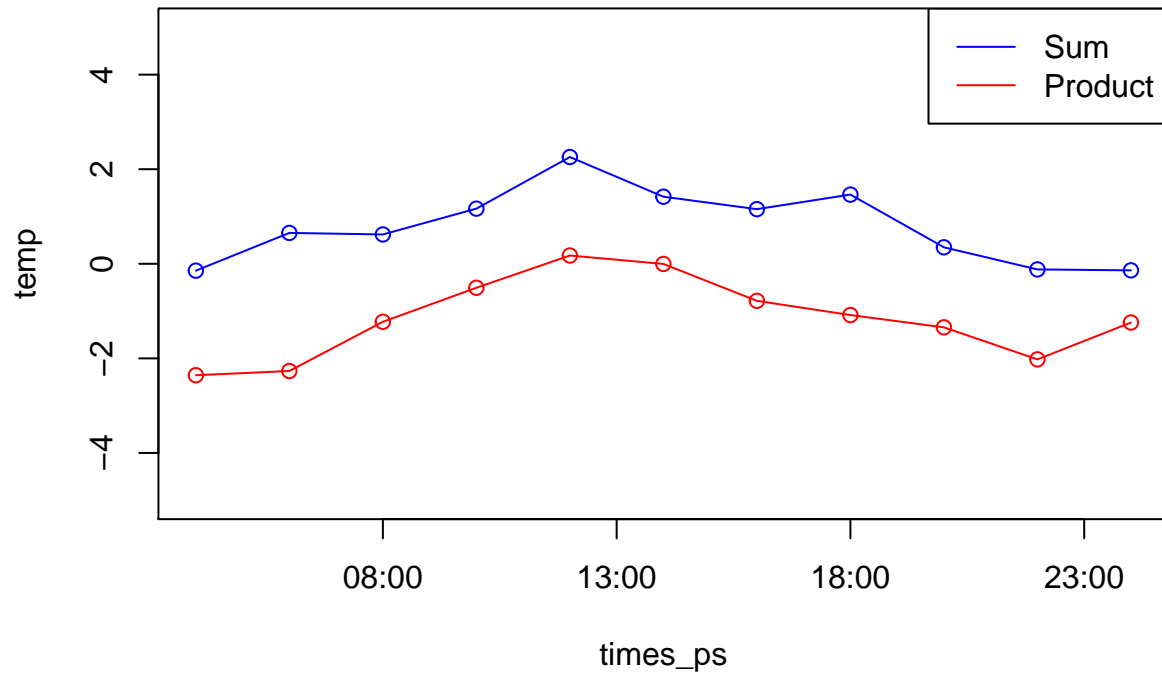
Implementation 2

In this implementation we choose to disregard the year information meaning we only care on the difference that the prediction date has with the observed but not in years but between the week so for example the distance between '2013-12-23' and '2012-12-23' will be zero. Also we modify the time distance on '24-hour-distance' so for example, the distance between 21:00 and 01:00 is treated as 4 hours. With these 2 changes we build a new model.



The above plot shows the temperature predictions for Linköping for the date of '2013-08-23' with the summation and the multiplication kernel but with the modified model prediction.

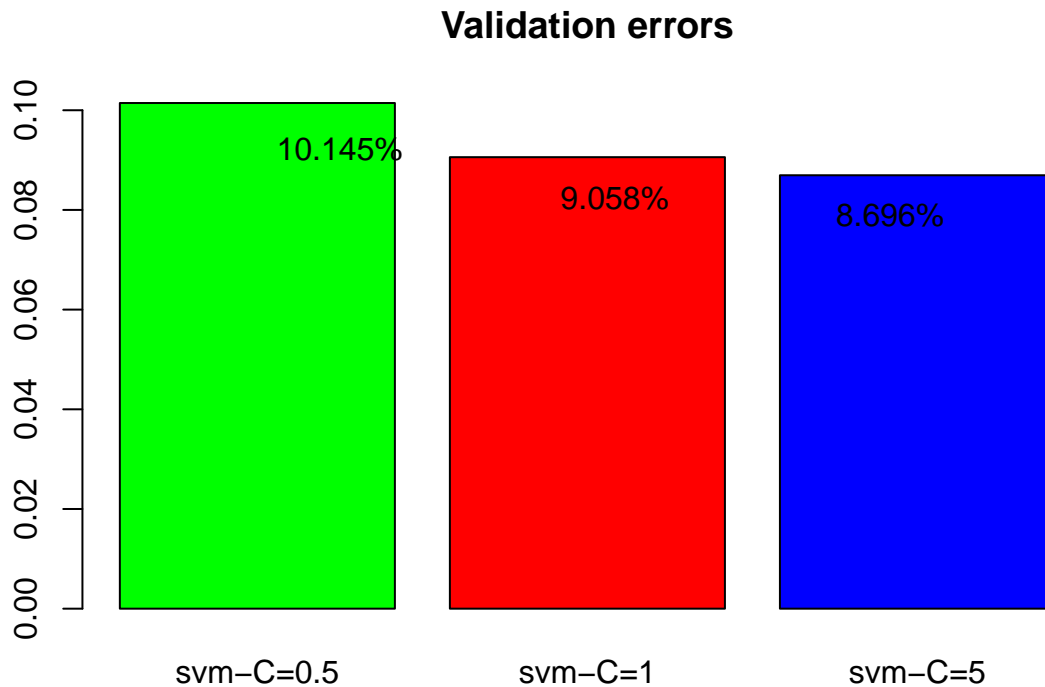
Liknoping '2013-08-23



The above plot shows the temperature predictions for Linköping for the date of '2013-12-23' with the summation and the multiplication kernel but with the modified model prediction.

Again the multiplication kernel model is achieving reasonable predictions for the 2 dates but the summation model only for winter has a reasonable prediction for the summer the prediction is slightly better than the one from the implementation 1. The h values we used where $(50000, 27, 3600 \times 1.5)$ for (distance, days, time) respectfully.

Assignment 2



We choose the model with $C=5$ because it has the lowest misclassification error among the other values of C . Then we use the value of the parameter C in order to calculate the generalization error on a portion of the data we kept as test.

```
## =====  
## The generalization error for the best C value is: 7.600434 %
```

The model that will be returned to the user is:

```
## =====  
## 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
```

Finally, the role of C is to control the regularization of the model. Once the C is large the model might have high variance. On the contrary a small C will lead to high bias for the model.

Appendix

```
set.seed(1234567890)
library(geosphere)

stations <- read.csv("stations.csv",stringsAsFactors = F,fileEncoding = "latin1")
temps <- read.csv("temps50k.csv",stringsAsFactors = F)
st <- merge(stations,temps,by="station_number")
h_distance <- 30000      # These three values are up to the students
h_date <- -25
h_time <- -5
a <- 58.4137 #latitude # The point to predict (up to the students)
b <- 15.6235 #logitude
date1 <- "2013-08-23" # 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 <- vector(length=length(times))
# Students' code here

timesf<-as.POSIXct(times,format="%H:%M:%S")
pred_date<-as.POSIXct( date1 , format = "%Y-%m-%d")

st$date<-as.POSIXct(st$date,format="%Y-%m-%d")
st$time<-as.POSIXct(st$time,format="%H:%M:%S")

st_filtered<-st[st$date< pred_date ,]

d1<-distm(st_filtered[,c("longitude","latitude")], c(b,a),fun=distHaversine)
d2<-as.numeric(difftime(pred_date ,st_filtered$date,units = "days"))

func <-function( time ,h){
  dist <- as.numeric(difftime(st_filtered$time,time , units="hours"))
  dist [dist >12] <- 24 - dist [dist >12]
  exp(-(dist/h)^2)
}

k1<-exp(-(d1/h_distance)^2)
k2<-exp(-(d2/h_date)^2)

for(i in 1:length(timesf)){
  k3<-func(timesf[i],h_time)
  K<-as.vector(k1)+as.vector(k2)+as.vector(k3)
  temp[i]<-sum(K*st_filtered$air_temperature)/sum(K)
}
```

```

temp_mult<-vector(length=length(times))
for(i in 1:length(timesf)){
  k3<-func(timesf[i],h_time)
  K1<-as.vector(k1)*as.vector(k2)*as.vector(k3)
  temp_mult[i]<-sum(K1*st_filtered$air_temperature)/sum(K1)
}

plot(timesf,temp, type="o",col="blue",main="Linköping '2013-08-23' Sum kernel",ylim=c(3,20))
lines(timesf,temp_mult,type="o",col="red")
legend("topright", legend=c("Sum", "Product"), col=c("blue", "red"),lty = 1)

set.seed(1234567890)
library(geosphere)

stations <- read.csv("stations.csv",stringsAsFactors = F,fileEncoding = "latin1")
temps <- read.csv("temps50k.csv",stringsAsFactors = F)
st <- merge(stations,temps,by="station_number")
h_distance <- 30000 # These three values are up to the students
h_date <- -25
h_time <- -5
a <- 58.4137 #latitude # The point to predict (up to the students)
b <- 15.6235 #logitude
date1 <- "2013-12-23" # 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 <- vector(length=length(times))
# Students' code here

timesf<-as.POSIXct(times,format="%H:%M:%S")
pred_date<-as.POSIXct( date1 , format = "%Y-%m-%d")

st$date<-as.POSIXct(st$date,format="%Y-%m-%d")
st$time<-as.POSIXct(st$time,format="%H:%M:%S")

st_filtered<-st[st$date< pred_date ,]

d1<-distm(st_filtered[,c("longitude","latitude")], c(b,a),fun=distHaversine)
d2<-as.numeric(difftime(pred_date ,st_filtered$date,units = "days"))

func <-function( time ,h){
  dist <- as.numeric(difftime(st_filtered$time,time , units="hours"))
  dist [dist >12] <- 24 - dist [dist >12]
  exp(-(dist/h)^2)
}

```

```

k1<-exp(-(d1/h_distance)^2)

k2<-exp(-(d2/h_date)^2)

for(i in 1:length(timesf)){
  k3<-func(timesf[i],h_time)
  K<-as.vector(k1)+as.vector(k2)+as.vector(k3)
  temp[i]<-sum(K*st_filtered$air_temperature)/sum(K)
}

temp_mult<-vector(length=length(times))
for(i in 1:length(timesf)){
  k3<-func(timesf[i],h_time)
  K1<-as.vector(k1)*as.vector(k2)*as.vector(k3)
  temp_mult[i]<-sum(K1*st_filtered$air_temperature)/sum(K1)
}

plot(timesf,temp, type="o",col="blue",main="Linkoping '2013-08-23' Sum kernel",ylim=c(-5,7))
lines(timesf,temp_mult,type="o",col="red")
legend("topright", legend=c("Sum", "Product"), col=c("blue", "red"),lty = 1)

set.seed(1234567890)
library(geosphere)

stations <- read.csv("stations.csv",stringsAsFactors = F,fileEncoding = "latin1")
temps <- read.csv("temps50k.csv",stringsAsFactors = F)
st <- merge(stations,temps,by="station_number")
h_distance <- 50000 # These three values are up to the students
h_date <- 27
h_time <- 3600 * 1.5
a <- 58.413738 #latitude # The point to predict (up to the students)
b <- 15.623510 #logitude
date1 <- "2015-08-23" # 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 <- vector(length=length(times))
# Students' code here

timesf<-as.difftime(times , units = "sec")
pred_date<-as.POSIXlt(as.Date(date1))$yday

st$yday <-as.POSIXlt(st$date)$yday
st$sectime <-as.difftime(as.character(st$ time), units= "secs" )

```



```

st_filtered<-st[st$date< pred_date ,]

d1<-distm(st_filtered[,c("longitude","latitude")], c(b,a),fun=distHaversine)

func_diff <- function(a,b,p ) {
  dif <- abs(as.numeric(a - b))
  ifelse( 2*dif > p , p- dif , dif )
}

func <-function(time ,h){
  dist <- func_diff(time , st$sectime , 86400)
  exp(-(dist/h)^2)
}

k1<-exp(-(d1/h_distance)^2)

k2<-exp(-(func_diff(pred_date,st_filtered$yday,365)/h_date)^2)

for(i in 1:length(timesf)){
  k3<-func(timesf[i],h_time)
  K<-as.vector(k1)+as.vector(k2)+as.vector(k3)
  temp[i]<-sum(K*st_filtered$air_temperature)/sum(K)
}

temp_mult<-vector(length=length(times))
for(i in 1:length(timesf)){
  k3<-func(timesf[i],h_time)
  K1<-as.vector(k1)*as.vector(k2)*as.vector(k3)
  temp_mult[i]<-sum(K1*st_filtered$air_temperature)/sum(K1)
}

times_ps<-as.POSIXct(times,format="%H:%M:%S")
plot(times_ps,temp, type="o",main="Liknoping '2013-08-23",col="blue",ylim=c(5,20))
lines(times_ps,temp_mult,col="red",type ="o")
legend("topright", legend=c("Sum", "Product"), col=c("blue", "red"),lty = 1)

set.seed(1234567890)
library(geosphere)

stations <- read.csv("stations.csv",stringsAsFactors = F,fileEncoding = "latin1")
temps <- read.csv("temps50k.csv",stringsAsFactors = F)
st <- merge(stations,temps,by="station_number")
h_distance <- 50000 # These three values are up to the students
h_date <-27
h_time <-3600 * 1.5

```

```

a <- 58.413738 #latitude # The point to predict (up to the students)
b <- 15.623510 #logitude
date1 <- "2015-12-23" # 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 <- vector(length=length(times))
# Students' code here

timesf<-as.difftime(times , units = "sec")
pred_date<-as.POSIXlt(as.Date(date1))$yday

st$yday <-as.POSIXlt(st$date)$yday
st$sectime <-as.difftime(as.character(st$ time), units= "secs" )

st_filtered<-st[st$date< pred_date ,]

d1<-distm(st_filtered[,c("longitude", "latitude")], c(b,a),fun=distHaversine)

func_diff <- function(a,b,p ) {
  dif <- abs(as.numeric(a - b))
  ifelse( 2*dif > p , p- dif , dif )
}

func <-function(time ,h){
  dist <- func_diff(time , st$sectime , 86400)
  exp(-(dist/h)^2)
}

k1<-exp(-(d1/h_distance)^2)

k2<-exp(-(func_diff(pred_date,st_filtered$yday,365)/h_date)^2)

for(i in 1:length(timesf)){
  k3<-func(timesf[i],h_time)
  K<-as.vector(k1)+as.vector(k2)+as.vector(k3)
  temp[i]<-sum(K*st_filtered$air_temperature)/sum(K)
}

temp_mult<-vector(length=length(times))
for(i in 1:length(timesf)){
  k3<-func(timesf[i],h_time)
  K1<-as.vector(k1)*as.vector(k2)*as.vector(k3)
  temp_mult[i]<-sum(K1*st_filtered$air_temperature)/sum(K1)
}

```

```

}

times_ps<-as.POSIXct(times,format="%H:%M:%S")
plot(times_ps,temp, type="o",main="Liknoping '2013-08-23",col="blue",ylim=c(-5,5))
lines(times_ps,temp_mult,col="red",type="o")
legend("topright", legend=c("Sum", "Product"), col=c("blue", "red"),lty = 1)

library(kernlab)

#load buid-in dataset from kernlab
data("spam")

#Split data to train,valid,test
n=dim(spam)[1]
set.seed(1234567890)
id=sample(1:n, floor(n*0.5))
train_spam=spam[id,]
id1=setdiff(1:n, id)
set.seed(1234567890)
id2=sample(id1, floor(n*0.3))
valid_spam=spam[id2,]
id3=setdiff(id1,id2)
test_spam=spam[id3,]

#fit svm models for every C value
svm1<-ksvm(type~.,data=train_spam,
           kernel = "rbfdot", kpar =list(sigma = 0.05),C = 0.5)

svm2<-ksvm(type~.,data=train_spam,
           kernel = "rbfdot", kpar =list(sigma = 0.05),C = 1)

svm3<-ksvm(type~.,data=train_spam,
           kernel = "rbfdot", kpar =list(sigma = 0.05),C = 5)

index_class_column<-which(names(spam)=="type") #index of the class column
#make predictions on valid data with the 3 models
svm1_preds_valid<-predict(svm1,valid_spam[,-index_class_column],
                          type="response")

svm2_preds_valid<-predict(svm2,valid_spam[,-which(names(spam)=="type")],
                          type="response")

svm3_preds_valid<-predict(svm3,valid_spam[,-index_class_column],
                          type="response")
#calculate misclassification errors on valid data
error1<-mean(valid_spam$type!=svm1_preds_valid)
error2<-mean(valid_spam$type!=svm2_preds_valid)
error3<-mean(valid_spam$type!=svm3_preds_valid)
#combine data

```

```

errors_valid<-c(error1,error2,error3)
#plot the misclassification errors
barplot(errors_valid,names.arg=c("svm-C=0.5", "svm-C=1", "svm-C=5"),col=c("green","red","blue"),
        main = "Validation errors")
text((errors_valid/1.1),labels=paste0(round(errors_valid*100,digits=3),"%"))

#combine train and valid data
dt<-rbind(train_spam,valid_spam)
#fit svm with train and valid data
svm_best<-ksvm(type~.,data=dt,
               kernel = "rbfdot", kpar =list(sigma = 0.05),C = 5)
#predict on test data
svm3_preds_test<-predict(svm_best,test_spam[,-58],type="response")
#calculate misclassification error
error6<-mean(test_spam$type!=svm3_preds_test)
cat("=====\n",
    "The generalization error for the best C value is: ",error6*100,"%")

cat("=====\n")
svm_final<-ksvm(type~.,data=spam,
                kernel = "rbfdot", kpar =list(sigma = 0.05),C = 5)
svm_final

```