Block1_Lab3_GroupK8

Nicolas Taba & Siddharth Saminathan & Salvador Marti Roman

13/12/2020

Statement of contribution

Nicolas Taba solved assignment 1, Salvador Marti Roman solved assignment 2 and SIddharth Saminathan solved assignment 3. All group members discussed the exercises and encountered issues with the others.

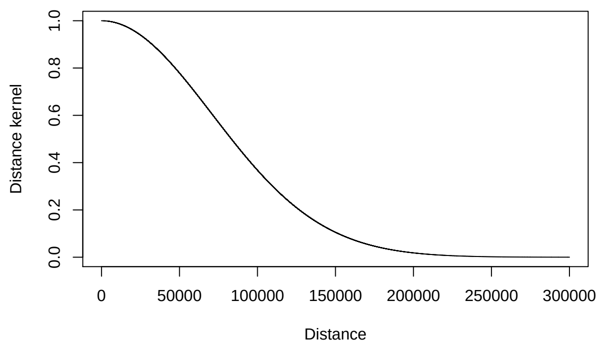
Assignment 1: Kernel Methods

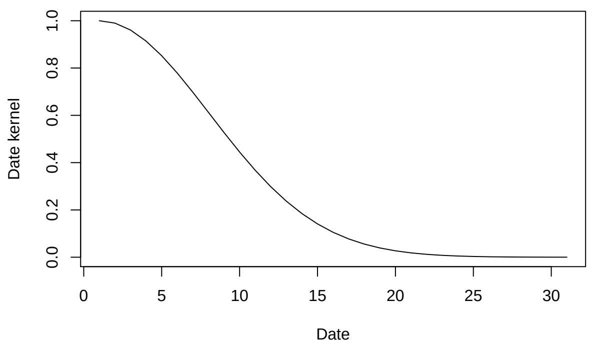
In this assignment, we are asked to implement a kernel method to predict the hourly temperature at a certain date and place in Sweden. The data is provided by the Swedish Meteorological and Hydrological Institute. We perform the prediction using a kernel that is the sum of gaussian kernels and another that is the product of gaussian kernels. The three kernels used are for distance, date and time. We can compute the kernels for physical distance and date as we loop over the kernel for time.

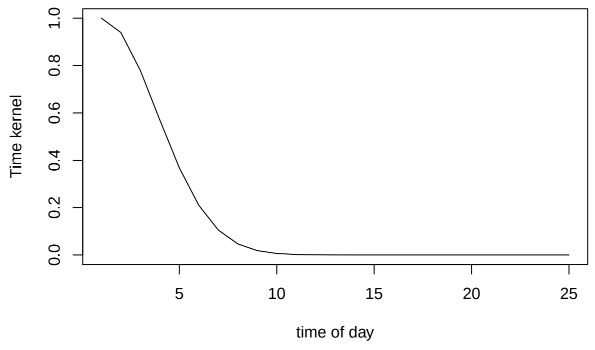
We also filter out the dates that happen after our target date. In this implementation of the kernel methods, we chose to change the distance units into kilometers in order to avoid computational errors with large floating point numbers.

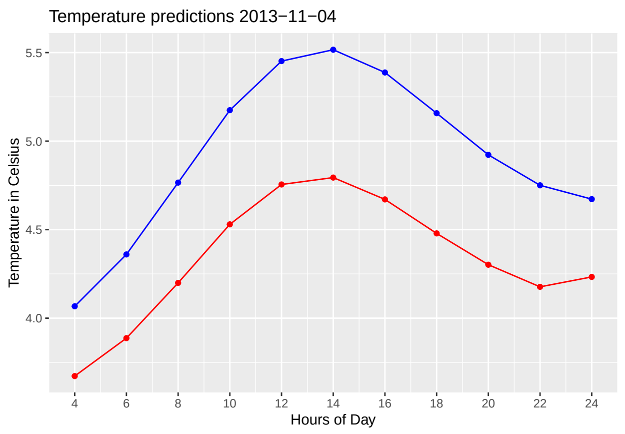
The width of the kernels were chosen heuristically. We have chosen a distance width of 100 km to cover large areas and have some overlap in our kernels. The date width is 10 days in order to capture variations within monthly temperature readings and time width is chosen to be 3h in order to have more strongly of the variation of temperature in the day.

```
## Warning: package 'geosphere' was built under R version 4.0.3
```









We have plotted the three kernels as a function of the quantity they are mapped from (physical distance, date and time). We find large kernel values close to 0 and small far from it as expected. This is the desired shape of the kernel. We then plotted our temperature estimates during the day from 4:00 to 24:00 on the 2013-11-04 both with the sum of kernels (in blue) and the product of kernels (in red).

The methods differ in that summation of kernels leads to larger kernel terms weighing more than others in their expression and thus contribute much more to the prediction than the other terms of the sum. This means that the sum of kernels only gives importance to larger terms for predictions. This is not desirable. For a product of kernels, all elements contribute to the prediction. The small values are accounted for in the calculation and yield a smaller estimate of the temperature than when using the sum of kernels. This accounts for more variations in the predictions and is a more desirable behavior. We would prefer to use the product of kernels for predictions.

2. Support Vector Machines

The code in the file Lab3Block1 2020 SVMs.R performs SVM model selection by using the function ksvm from the R package kernlab, in order to learn a SVM for classifying the spam dataset that is included with the package. All the models to select from use the radial basis function kernel (also known as Gaussian) with a width of 0.05. The C parameter varies between the models.

```
## Warning: package 'kernlab' was built under R version 4.0.3

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha

## [1] 0.07

## [1] 0.08489388

## [1] 0.08364544

## [1] 0.03370787
```

1. Which filter do we return to the user ? filter0, filter1, filter2 or filter3 ? Why?

We would return filter 3 to the user. This is the model that uses the most data points to return a filter and is thus the preferred solution to return to the user since it has trained on the most cases.

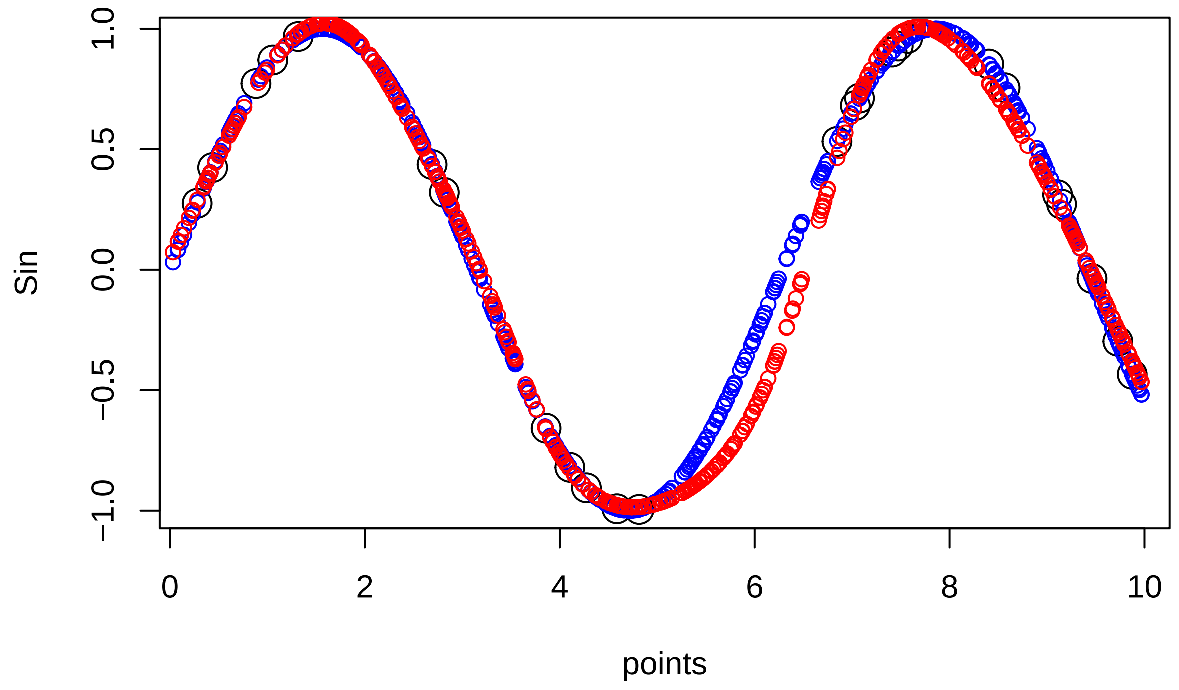
2. What is the estimate of the generalization error of the filter returned ? err0, err1, err2 or err3 ? Why ?

The best estimate of the generalization error will come from filter 2, because that is the one that actually tests for unseen data. This is a good enough estimate of the performance of the filter on unseen data and only represents a lower bound for our performance. This is also the filter that uses the most data out of those that use unseen data to test the model. The filter given to the user (filter 3) might perform better, but it will at least perform as well as filter 2. We prefer to be conservative when reporting the performance of the model. Thus, the estimate of the generalization error is 16.4% (filter 2 error)

3.NEURAL NETWORKS

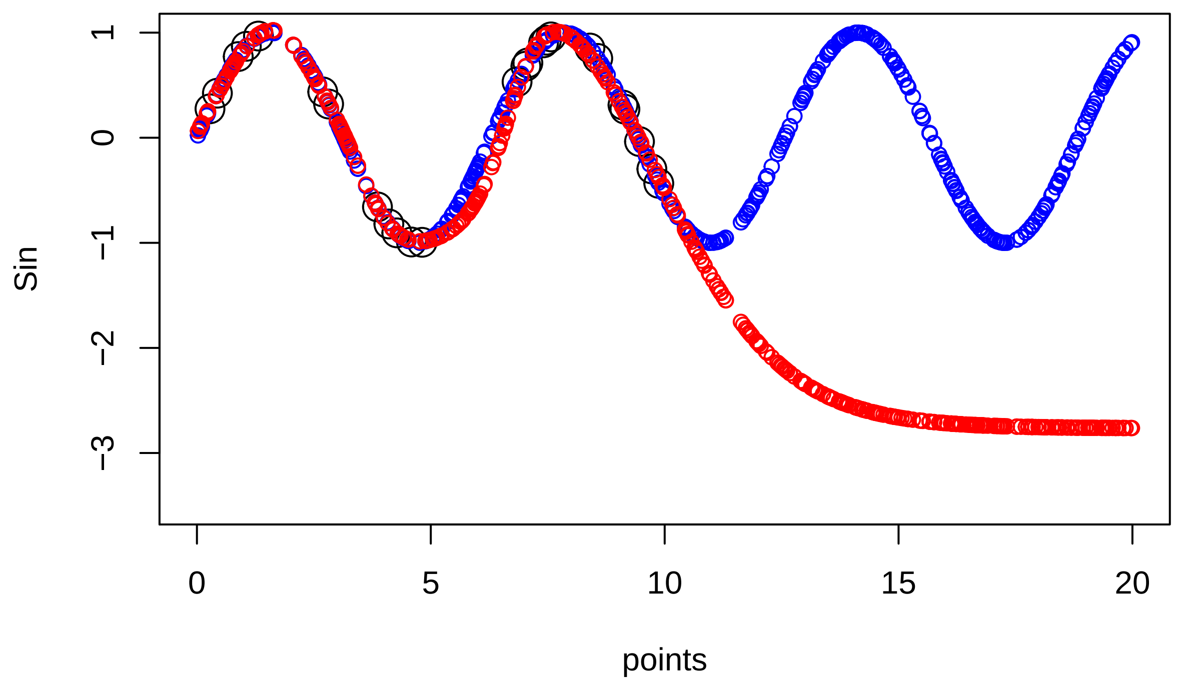
Training with points from U~[0,10]

```
## Warning: package 'neuralnet' was built under R version 4.0.3
```



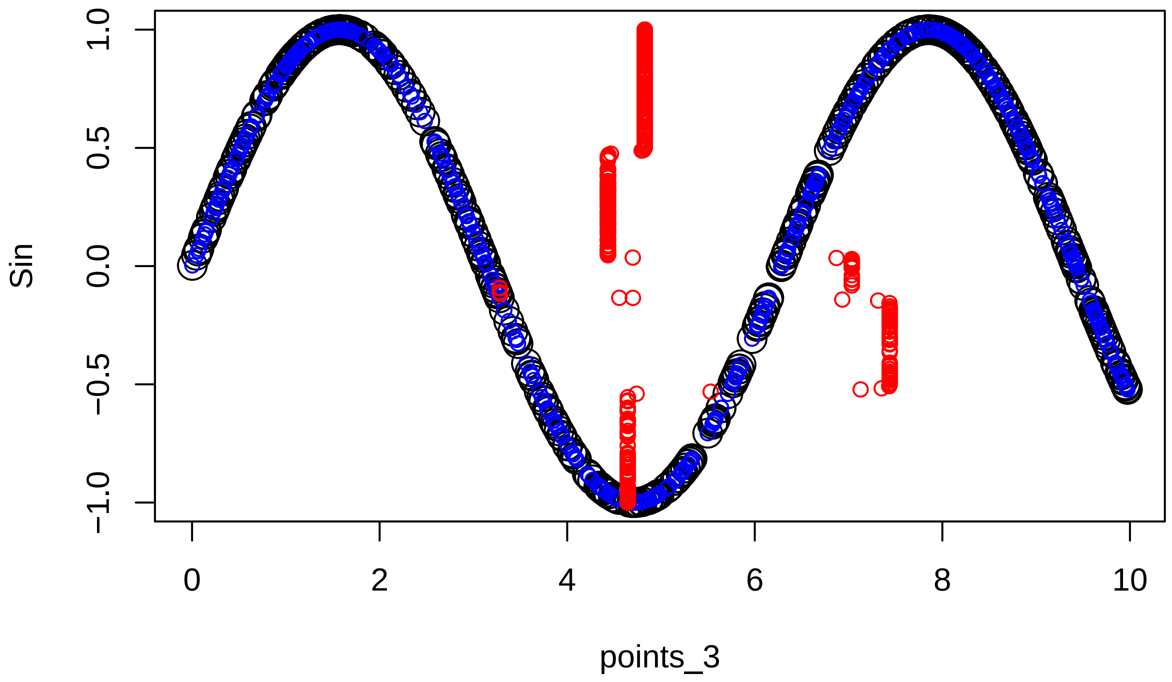
The results seems to be good with hidden layers=3.The neural network is able to predict the sine function with little error. The periodic nature of the function is maintained.

Training the Model using points from U~[0,20]



Since we used our previous model with points from [0,10], the accuracy of our model is high on the interval [0,10]. However the test data points range from [0,20]. The predictions seems to be accurate in this case too. The predictions stop being accurate after points>10.

Training a model with points U~[0,20], and predicting Points(x) from Sin instead of Sin from Points(x).



This model does not give us promising results as the previous models. This is because we are trying to predict Points(x) from Sine function. The predicted values do not follow our target.This is because sine function is a periodic function and for any output of sin(x)=y, there are several values of x that satisfy this equation. The neural network cannot predict which of these values satisfies the equation. The neural network would perform better on intervals shorter than the period of the function.

Appendix: All code for this report

```r
knitr::opts_chunk$set(echo = TRUE)
set.seed(1234567890)
library(geosphere)
stations <- read.csv("stations.csv")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")

# Location chosen
x_location <- c(14.826, 58.4274)
x_date <- "2013-11-04"
x_times <- paste(02:12*2, "00","00", sep = ":")

# Choice of width
h_distance <- 100000 #100km as distance to get some overlap and cover some area
h_date <- 10 #To capture seasonality and monthly trends
h_time <- 4 #To capture fluctuations of temperature in a day
weights <- c(h_distance, h_date, h_time)

# Plot distance kernel
distance <- seq(0,300000,1)/100000
k_dist <- exp(-(distance)^2)
plot(k_dist, type = "l", xlab = "Distance", ylab = "Distance kernel")
# abline(v=100, col="red")
# Plot date kernel
date <- seq(0,30,1)/10
k_date <- exp(-(date)^2)
plot(k_date, type= "l", xlab= "Date", ylab = "Date kernel")
# abline(v=10, col="red")
# Plot time kernel
time <- seq(0,24,1)/4
k_time <- exp(-(time)^2)
plot(k_time, type = "l", xlab = "time of day", ylab = "Time kernel")
# abline(v=4,col="red")


# Functions
# Kernels are f(u)=(exp(-1/2*u^2)) with u = distance/h
# gaussian kernel distance
gaussian_phys_dist <- function(a,b,h){
  #a must be a data frame and b must be valid vectors of length = 2 representing the long. and lat.
  #returns distance in manageable units to avoid computational errors
  # dis <- (distHaversine(data.frame(a$longitude, a$latitude),b)/1000)/h
  dis <- (distHaversine(data.frame(a$longitude, a$latitude),b))/h
  k <- exp(-(1/2)*dis^2)
  return(k)
}

# Gaussian kernel day
gaussian_date_dist <- function(a,b,h){
  #a is a data frame and b is a valid format date
  # difftime() yields NaN for prod kernel
  diff_day <- ifelse( (as.numeric(as.Date(a$date) - as.Date(b), unit="days") %% 365.25) > (365.25/2),
                      365 - floor(as.numeric(as.Date(a$date) - as.Date(b), unit="days")%%365.25),
                      floor(as.numeric(as.Date(a$date) - as.Date(b), unit="days")%%365.25)) # This filt
  # d_dist <- (as.numeric(as.Date(a$date) - as.Date(b), unit="days"))/h
   d_dist <- diff_day/h
  k <- exp(-(1/2)*d_dist^2)
  return(k)
}

# Gaussian kernel time
gaussian_time_dist <- function(a,b,h){
  # a is a data frame and b is the proper format time.
  time_diff <- difftime(strptime(a$time , format = "%H:%M:%S"), strptime(b , format = "%H:%M:%S"))
  time_diff <- abs(as.numeric(time_diff/(3600)))
  t_dist <- time_diff/h
  k <- exp(-(1/2)*t_dist^2)
  return(k)
}



# Filter dates in data posterior to target date
filter <- function(df, date, time) {
  return (df[!as.Date(df$date) > as.Date(date),])
}

# Estimates
estimates <- function(st, weights){
  filtered_data <- filter(st, x_date)
  # filtered_time <- filtered_data
  k_distance <- gaussian_phys_dist(filtered_data, x_location, weights[1])
  k_distance
  k_day <- gaussian_date_dist(filtered_data, x_date, weights[2])

  estimate_sum <- c()
  estimate_prod <- c()

  for(i in 1:length(x_times)){
    k_time <- gaussian_time_dist(filtered_data, x_times[i], weights[3])
    kernel_sum <- k_distance + k_day + k_time
    kernel_prod <- k_distance * k_day * k_time

    estimate_sum[i] <- sum(kernel_sum %*% filtered_data$air_temperature) / sum(kernel_sum)
    estimate_prod[i] <- sum(kernel_prod %*% filtered_data$air_temperature) / sum(kernel_prod)
  }
  return(data.frame(sum_kernels = estimate_sum, prod_kernels = estimate_prod))
}

# Calculate estimates
temperatures <- estimates(st, weights)

#plot estimates
library(ggplot2)
plot <- ggplot() +
```

```r
  geom_point(aes(x=2:12*2, y=temperatures$sum_kernels), color="blue") +
  geom_line(aes(x=2:12*2, y=temperatures$sum_kernels), color="blue") +
  geom_point(aes(x=2:12*2, y=temperatures$prod_kernels), color="red") +
  geom_line(aes(x=2:12*2, y=temperatures$prod_kernels), color="red") +
  scale_x_continuous(breaks = seq(2,24,2))+
  labs(title = "Temperature predictions 2013-11-04", x = "Hours of Day", y = "Temperature in Celsius")
plot

# Lab 3 block 1 of 732A99/TDDE01 Machine Learning
# Author: jose.m.pena@liu.se
# Made for teaching purposes

library(kernlab)
set.seed(1234567890)

data(spam)

index <- sample(1:4601)
tr <- spam[index[1:3000], ]
va <- spam[index[3001:3800], ]
trva <- spam[index[1:3800], ]
te <- spam[index[3801:4601], ]

by <- 0.3
err_va <- NULL
for(i in seq(by,5,by)){
  filter <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=i)
  mailtype <- predict(filter,va[,-58])
  t <- table(mailtype,va[,58])
  err_va <-c(err_va,(t[1,2]+t[2,1])/sum(t))
}

# Training in TR, predicting on validation
filter0 <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
mailtype <- predict(filter0,va[,-58])
t <- table(mailtype,va[,58])
err0 <- (t[1,2]+t[2,1])/sum(t)
err0

# Training on TR, prediction on te
filter1 <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
mailtype <- predict(filter1,te[,-58])
t <- table(mailtype,te[,58])
err1 <- (t[1,2]+t[2,1])/sum(t)
err1

# Training on training and validation dataset, prediction on test
filter2 <- ksvm(type~.,data=trva,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
mailtype <- predict(filter2,te[,-58])
t <- table(mailtype,te[,58])
err2 <- (t[1,2]+t[2,1])/sum(t)
err2
```

```r
#  Training on whole data, prediction on test
filter3 <- ksvm(type~.,data=spam,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
mailtype <- predict(filter3,te[,-58])
t <- table(mailtype,te[,58])
err3 <- (t[1,2]+t[2,1])/sum(t)
err3

# Questions

# 1. Which filter do we return to the user ? filter0, filter1, filter2 or filter3 ? Why ?

# 2. What is the estimate of the generalization error of the filter returned ? err0, err1, err2 or err3

library(neuralnet)

#Generating points from uniform distribution
set.seed(1234567890)
points<-runif(500,min=0,max=10)

#aaplying sin fucntion
data<-data.frame(points,Sin=sin(points))

train<-data[1:25,]
test<-data[26:500,]

winit<-runif(16,-1,1)

nn<-neuralnet(Sin~points,train,hidden=5,startweights  = winit)
plot(train, cex=2)
points(test, col = "blue", cex=1)
points(test[,1],predict(nn,test), col="red", cex=1)
#2
#Generating points from uniform dsitribution

points<-runif(500,min=0,max=20)

#aaplying sin fucntion
data_2<-data.frame(points,Sin=sin(points))


plot(train, cex=2,xlim=c(0,20),ylim=c(-3.5,1))
points(data_2, col = "blue", cex=1)
points(data_2[,1],predict(nn,data_2), col="red", cex=1)

#3
points_3<-runif(500,0,10)
data_3<-data.frame(points_3,Sin=sin(points_3))
nn_2<-neuralnet(points_3~Sin,data_3,hidden=5,startweights  = winit)
pred_2<-predict(nn_2,data_3)
plot(data_3, cex=2)
points(data_3, col = "blue", cex=1)
points(pred_2,data_3$Sin, col="red", cex=1)
```