

732A99_lab3_block1_A2

Anubhav Dikshit(anudi287), Lennart Schilling(lensc874), Thijs Quast(thiqu264)

17 December 2018

Contents

Assignment 1	2
1. Implement a kernel method to predict the hourly temperatures for a date and place in Sweden. To do so, you are provided with the files stations.csv and temps50k.csv. These files contain information about weather stations and temperature measurements in the stations at different days and times. The data have been kindly provided by the Swedish Meteorological and Hydrological Institute (SMHI).	2
Assignment 2	5
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.	5
Appendix	8

Contributions

During the lab, Anubhav focused on assignment 2, Lennart focused on assignment 3 and Thijs focused on assignment 4. All codes and analysis was indepedently done and is also reflected in the individual reports.

Loading The Libraries

```
if (!require("pacman")) install.packages("pacman")
pacman::p_load(geosphere, kernlab, geosphere, ggplot2, caret)

set.seed(12345)
options("jtools-digits" = 2, scipen = 999)

# colours (colour blind friendly)
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2",
               "#D55E00", "#CC79A7")
```

Assignment 1

1. Implement a kernel method to predict the hourly temperatures for a date and place in Sweden. To do so, you are provided with the files `stations.csv` and `temps50k.csv`. These files contain information about weather stations and temperature measurements in the stations at different days and times. The data have been kindly provided by the Swedish Meteorological and Hydrological Institute (SMHI).

```
rm(list=ls())
set.seed(1234567890)
stations <- read.csv("stations.csv")
temps <- read.csv("temps50k.csv")
st <- merge(stations, temps, by="station_number")
rm(temps, stations)
```

defining the function

```
kernel_method <- function(df, date, loc_long, loc_lat, h1, h2, h3) {

  set.seed(1234567890)
  start <- as.POSIXct(date)
  interval <- 60
  end <- start + as.difftime(1, units="days")
  time_seq <- seq(from=start, by=interval*120, to=end)
  time_seq <- as.data.frame(time_seq)
  colnames(time_seq) <- "new_date_time"
  time_seq$time_index <- rownames(time_seq)

  df_new <- merge.data.frame(df, time_seq, all=TRUE)
  rm(df)

  df_new$new_date <- as.Date(df_new$new_date_time)
  df_new$new_time <- format(df_new$new_date_time, "%H:%M:%S")
  df_new$loc_long <- loc_long
  df_new$loc_lat <- loc_lat
```

```

df_new$h_distance <- abs(distHaversine(p1 = df_new[,c("loc_long", "loc_lat")], p2 = df_new[,c("longitudi

df_new$h_date <- as.numeric(abs(difftime(df_new$new_date, df_new$date, units = c("days"))))

df_new$h_time <- as.numeric(abs(difftime(strptime(paste(df_new$new_date,
                                                    df_new$new_time), "%Y-%m-%d%H:%M:%S"),
                                                    strptime(paste(df_new$new_date, df_new$time),
                                                                    "%Y-%m-%d %H:%M:%S"),
                                                    units = c("hour"))))

df_new$date_time <- paste(df_new$date, df_new$time)
df_new$hd_dist <- as.numeric(difftime(df_new$new_date_time,
                                      df_new$date_time,
                                      units = c("hour")))

## removing any negative dates and time
df_new$posterior_flag <- as.factor(ifelse(df_new$h_distance > 0 & df_new$hd_dist > 0, "retain", "drop"))

## calculating kernel distance and choosing guassian kernel
df_new$h_distance_kernel <- exp(-(df_new$h_distance/h1)^2)
df_new$h_date_kernel <- exp(-(df_new$h_date/h2)^2)
df_new$h_time_kernel <- exp(-(df_new$h_time/h3)^2)
df_new$total_additive_dist <- (df_new$h_distance_kernel + df_new$h_date_kernel + df_new$h_time_kernel)
df_new$total_mul_dist <- (df_new$h_distance_kernel * df_new$h_date_kernel * df_new$h_time_kernel)

df_new$additive_num <- ifelse(df_new$posterior_flag == "retain",
                             df_new$h_distance_kernel*df_new$air_temperature +
                             df_new$h_date_kernel*df_new$air_temperature +
                             df_new$h_time_kernel*df_new$air_temperature,0)

df_new$mul_num <- ifelse(df_new$posterior_flag == "retain",
                        (df_new$h_distance_kernel*df_new$air_temperature) *
                        (df_new$h_date_kernel*df_new$air_temperature) *
                        (df_new$h_time_kernel*df_new$air_temperature),0)

df_new$additive_den <- ifelse(df_new$posterior_flag == "retain", df_new$total_additive_dist, 0)
df_new$mul_den <- ifelse(df_new$posterior_flag == "retain", df_new$total_mul_dist, 0)

time = unique(time_seq$time_index)
result <- NULL

for(i in time){
  temp <- df_new[df_new$time_index == i,]
  additive_temp <- sum(temp$additive_num)/sum(temp$additive_den)
  mult_temp <- sum(temp$mul_num)/sum(temp$mul_den)

  temp <- cbind(additive_temp, mult_temp, i)
  result <- rbind(temp,result)
}

result <- as.data.frame(result)

```

```

result <- merge(x = result, y = time_seq, by.x = "i", by.y = "time_index", all.x = TRUE)
result$additive_temp <- as.numeric(result$additive_temp)
result$mult_temp <- as.numeric(result$mult_temp)

p1 <- ggplot(data=result, aes(x=new_date_time)) +
  geom_point(aes(y = additive_temp)) +
  geom_point(aes(y = mult_temp)) +
  geom_line(aes(y = additive_temp, color = "Additive")) +
  geom_line(aes(y = mult_temp, color = "Multiplicative")) +
  scale_color_manual(values=c("#E69F00", "#56B4E9")) +
  ylab("predicted temperature") +
  theme_bw() +
  ggtitle("Predicted Temperature using Kernels")

final <- list(p1)
return(final)
}

```

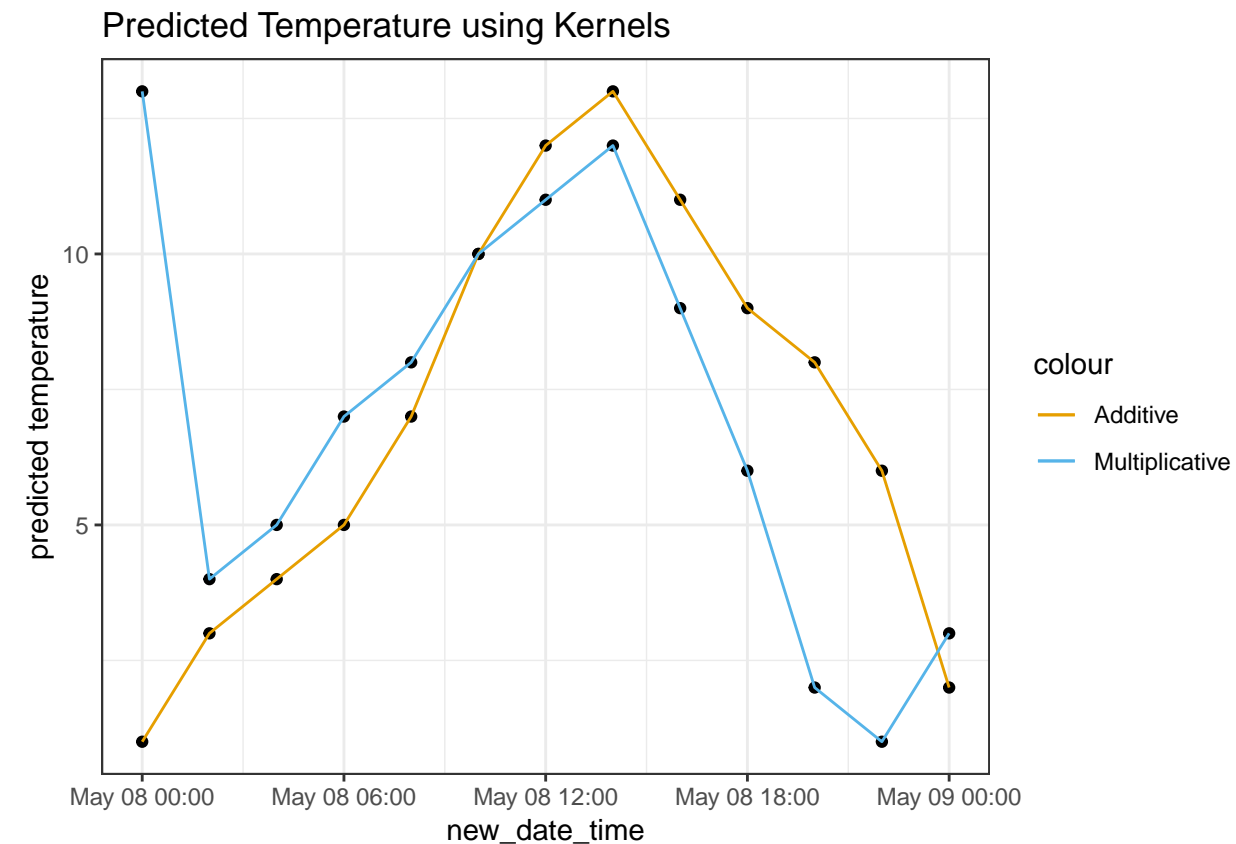
calling function

```

kernel_method(df=st, date = "2000-05-08", loc_long = 17.6935,
              loc_lat = 59.9953, h1 = 30000, h2 = 2, h3 = 5)

```

```
## [[1]]
```



Analysis: A good width for the distance is 30Kms, the reasoning behind this is that temperature in Linköping

and Norrköping tend to be similar but they vary by a few degrees, given that Sweden is way up north the temperature fluctuations will be less sensitive to distance than compared to the equator, thus 30Kms tend to be reasonable.

The width for the distance for day is 2, because I have personally experienced days where one day it's freezing and next day I am sweating, thus 2 days is what I have chosen for my width.

For the width of time, considering the shorter winter days I do expect 3 hours of the time to be an ideal window for temperature.

Assignment 2

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.

```
rm(list=ls())
set.seed(1234567890)

data(spam)

## create test and training set
index <- sample(1:dim(spam)[1])
spamtrain <- spam[index[1:floor(dim(spam)[1]/2)], ]
spamttest <- spam[index[(ceiling(dim(spam)[1]/2)) + 1]:dim(spam)[1]], ]

## train a support vector machine
model_0.05 <- ksvm(type~., data=spamtrain,
                  kernel="rbfdot",
                  kpar=list(sigma=0.05),
                  C=0.5)

model_1.0 <- ksvm(type~., data=spamtrain,
                  kernel="rbfdot",
                  kpar=list(sigma=1.0),
                  C=0.5)

model_5.0 <- ksvm(type~., data=spamtrain,
                  kernel="rbfdot",
                  kpar=list(sigma=5),
                  C=0.5)

model_0.05

## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc (classification)
## parameter : cost C = 0.5
##
```

```
## Gaussian Radial Basis kernel function.
## Hyperparameter : sigma = 0.05
##
## Number of Support Vectors : 1063
##
## Objective Function Value : -304.0238
## Training error : 0.044783
```

```
model_1.0
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc (classification)
## parameter : cost C = 0.5
##
## Gaussian Radial Basis kernel function.
## Hyperparameter : sigma = 1
##
## Number of Support Vectors : 2098
##
## Objective Function Value : -615.9819
## Training error : 0.22913
```

```
model_5.0
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc (classification)
## parameter : cost C = 0.5
##
## Gaussian Radial Basis kernel function.
## Hyperparameter : sigma = 5
##
## Number of Support Vectors : 2150
##
## Objective Function Value : -649.5897
## Training error : 0.274348
```

```
x <- as.matrix(spamtrain[,-58])
```

```
# confusion table
```

```
conf_model_0.05 <- table(spamtest[,58], predict(model_0.05,spamtest[,-58]))
names(dimnames(conf_model_0.05)) <- c("Actual Test", "P2redicted Test")
caret::confusionMatrix(conf_model_0.05)
```

```
## Confusion Matrix and Statistics
##
##               P2redicted Test
## Actual Test nonspam spam
##      nonspam   1345    56
##      spam      155   744
##
##               Accuracy : 0.9083
##               95% CI : (0.8957, 0.9197)
```

```
##      No Information Rate : 0.6522
##      P-Value [Acc > NIR] : < 0.00000000000000022
##
##              Kappa : 0.8035
##      McNemar's Test P-Value : 0.00000000001514
##
##      Sensitivity : 0.8967
##      Specificity : 0.9300
##      Pos Pred Value : 0.9600
##      Neg Pred Value : 0.8276
##      Prevalence : 0.6522
##      Detection Rate : 0.5848
##      Detection Prevalence : 0.6091
##      Balanced Accuracy : 0.9133
##
##      'Positive' Class : nonspam
##
```

```
conf_model_1.0 <- table(spamtest[,58], predict(model_1.0,spamtest[, -58]))
names(dimnames(conf_model_1.0)) <- c("Actual Test", "P2redicted Test")
caret::confusionMatrix(conf_model_1.0)
```

```
## Confusion Matrix and Statistics
##
##              P2redicted Test
## Actual Test nonspam spam
##      nonspam   1401    0
##      spam      685   214
##
##      Accuracy : 0.7022
##      95% CI : (0.683, 0.7208)
##      No Information Rate : 0.907
##      P-Value [Acc > NIR] : 1
##
##              Kappa : 0.2757
##      McNemar's Test P-Value : <0.0000000000000002
##
##      Sensitivity : 0.6716
##      Specificity : 1.0000
##      Pos Pred Value : 1.0000
##      Neg Pred Value : 0.2380
##      Prevalence : 0.9070
##      Detection Rate : 0.6091
##      Detection Prevalence : 0.6091
##      Balanced Accuracy : 0.8358
##
##      'Positive' Class : nonspam
##
```

```
conf_model_0.05 <- table(spamtest[,58], predict(model_5.0,spamtest[, -58]))
names(dimnames(conf_model_0.05)) <- c("Actual Test", "P2redicted Test")
caret::confusionMatrix(conf_model_0.05)
```

```
## Confusion Matrix and Statistics
##
```

```
##           P2redicted Test
## Actual Test nonspam spam
##   nonspam   1401    0
##   spam      769   130
##
##           Accuracy : 0.6657
##           95% CI : (0.646, 0.6849)
##   No Information Rate : 0.9435
##   P-Value [Acc > NIR] : 1
##
##           Kappa : 0.1708
## Mcnemar's Test P-Value : <0.0000000000000002
##
##           Sensitivity : 0.6456
##           Specificity : 1.0000
##   Pos Pred Value : 1.0000
##   Neg Pred Value : 0.1446
##           Prevalence : 0.9435
##   Detection Rate : 0.6091
##   Detection Prevalence : 0.6091
##   Balanced Accuracy : 0.8228
##
##   'Positive' Class : nonspam
##
```

Analysis:

From the summary of the three models build we can see that the accuracy of models are 90.83%, 70.22%, 66.57% respectively. Accuracy is only half the story, as a good spam detection should never classify a good mail has 'spam', which is something that model2 and model3 are doing. However in model 3 the accuracy is least thus, given a choice i would select model2 has the best model despite the lower accuracy.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
if (!require("pacman")) install.packages("pacman")
pacman::p_load(xlsx, glmnet, MASS, jtools, huxtable, ggplot2,
               ggthemes, gridExtra, ROCR, broom, caret, e1071,
               kknn, tidyr, dplyr, reshape2, glmnet)

options("jtools-digits" = 2, scipen = 999)

if (!require("pacman")) install.packages("pacman")
pacman::p_load(geosphere, kernlab, geosphere, ggplot2, caret)

set.seed(12345)
options("jtools-digits" = 2, scipen = 999)

# colours (colour blind friendly)
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2",
               "#D55E00", "#CC79A7")

rm(list=ls())
```



```

set.seed(1234567890)
stations <- read.csv("stations.csv")
temps <- read.csv("temps50k.csv")
st <- merge(stations, temps, by="station_number")
rm(temps, stations)
kernel_method <- function(df, date, loc_long, loc_lat, h1, h2, h3) {

  set.seed(1234567890)
  start <- as.POSIXct(date)
  interval <- 60
  end <- start + as.difftime(1, units="days")
  time_seq <- seq(from=start, by=interval*120, to=end)
  time_seq <- as.data.frame(time_seq)
  colnames(time_seq) <- "new_date_time"
  time_seq$time_index <- rownames(time_seq)

  df_new <- merge.data.frame(df, time_seq, all=TRUE)
  rm(df)

  df_new$new_date <- as.Date(df_new$new_date_time)
  df_new$new_time <- format(df_new$new_date_time, "%H:%M:%S")
  df_new$loc_long <- loc_long
  df_new$loc_lat <- loc_lat

  df_new$h_distance <- abs(distHaversine(p1 = df_new[,c("loc_long", "loc_lat")], p2 = df_new[,c("longitudi

  df_new$h_date <- as.numeric(abs(difftime(df_new$new_date, df_new$date, units = c("days"))))

  df_new$h_time <- as.numeric(abs(difftime(strptime(paste(df_new$new_date,
                                                         df_new$new_time), "%Y-%m-%d%H:%M:%S"),
                                                         strptime(paste(df_new$new_date, df_new$time),
                                                         "%Y-%m-%d %H:%M:%S"),
                                                         units = c("hour")))))

  df_new$date_time <- paste(df_new$date, df_new$time)
  df_new$hd_dist <- as.numeric(difftime(df_new$new_date_time,
                                       df_new$date_time,
                                       units = c("hour"))))

  ## removing any negative dates and time
  df_new$posterior_flag <- as.factor(ifelse(df_new$h_distance > 0 & df_new$hd_dist > 0, "retain", "drop"))

  ## calculating kernel distance and choosing gaussian kernel
  df_new$h_distance_kernel <- exp(-(df_new$h_distance/h1)^2)
  df_new$h_date_kernel <- exp(-(df_new$h_date/h2)^2)
  df_new$h_time_kernel <- exp(-(df_new$h_time/h3)^2)
  df_new$total_additive_dist <- (df_new$h_distance_kernel + df_new$h_date_kernel + df_new$h_time_kernel)
  df_new$total_mul_dist <- (df_new$h_distance_kernel * df_new$h_date_kernel * df_new$h_time_kernel)

  df_new$additive_num <- ifelse(df_new$posterior_flag == "retain",

```

```

        df_new$h_distance_kernel*df_new$air_temperature +
        df_new$h_date_kernel*df_new$air_temperature +
        df_new$h_time_kernel*df_new$air_temperature,0)

df_new$mul_num <- ifelse(df_new$posterior_flag == "retain",
                        (df_new$h_distance_kernel*df_new$air_temperature) *
                        (df_new$h_date_kernel*df_new$air_temperature) *
                        (df_new$h_time_kernel*df_new$air_temperature),0)

df_new$additive_den <- ifelse(df_new$posterior_flag == "retain", df_new$total_additive_dist, 0)
df_new$mul_den <- ifelse(df_new$posterior_flag == "retain", df_new$total_mul_dist, 0)

time = unique(time_seq$time_index)
result <- NULL

for(i in time){
  temp <- df_new[df_new$time_index == i,]
  additive_temp <- sum(temp$additive_num)/sum(temp$additive_den)
  mult_temp <- sum(temp$mul_num)/sum(temp$mul_den)

  temp <- cbind(additive_temp, mult_temp, i)
  result <- rbind(temp,result)
}

result <- as.data.frame(result)
result <- merge(x=result, y = time_seq, by.x = "i", by.y = "time_index", all.x = TRUE)
result$additive_temp <- as.numeric(result$additive_temp)
result$mult_temp <- as.numeric(result$mult_temp)

p1 <- ggplot(data=result, aes(x=new_date_time)) +
  geom_point(aes(y = additive_temp)) +
  geom_point(aes(y = mult_temp)) +
  geom_line(aes(y = additive_temp, color = "Additive")) +
  geom_line(aes(y = mult_temp, color = "Multiplicative")) +
  scale_color_manual(values=c("#E69F00", "#56B4E9")) +
  ylab("predicted temperature") +
  theme_bw() +
  ggtitle("Predicted Temperature using Kernels")

final <- list(p1)
return(final)
}

kernel_method(df=st, date = "2000-05-08", loc_long = 17.6935,
              loc_lat = 59.9953, h1 = 30000, h2 = 2, h3 = 5)
rm(list=ls())
set.seed(1234567890)

data(spam)

## create test and training set
index <- sample(1:dim(spam)[1])
spamtrain <- spam[index[1:floor(dim(spam)[1]/2)], ]

```

```

spamtest <- spam[index[((ceiling(dim(spam)[1]/2)) + 1):dim(spam)[1]], ]

## train a support vector machine
model_0.05 <- ksvm(type~., data=spamtrain,
  kernel="rbfdot",
  kpar=list(sigma=0.05),
  C=0.5)

model_1.0 <- ksvm(type~.,data=spamtrain,
  kernel="rbfdot",
  kpar=list(sigma=1.0),
  C=0.5)

model_5.0 <- ksvm(type~.,data=spamtrain,
  kernel="rbfdot",
  kpar=list(sigma=5),
  C=0.5)

model_0.05
model_1.0
model_5.0

x <- as.matrix(spamtrain[, -58])

# confusion table

conf_model_0.05 <- table(spamtest[,58], predict(model_0.05,spamtest[, -58]))
names(dimnames(conf_model_0.05)) <- c("Actual Test", "P2redicted Test")
caret::confusionMatrix(conf_model_0.05)

conf_model_1.0 <- table(spamtest[,58], predict(model_1.0,spamtest[, -58]))
names(dimnames(conf_model_1.0)) <- c("Actual Test", "P2redicted Test")
caret::confusionMatrix(conf_model_1.0)

conf_model_5.0 <- table(spamtest[,58], predict(model_5.0,spamtest[, -58]))
names(dimnames(conf_model_5.0)) <- c("Actual Test", "P2redicted Test")
caret::confusionMatrix(conf_model_5.0)

```