MATH208-A3

Christopher Zheng - 260760794

(a)

Write a function to compute p(x1, x2) for n observations which takes as arguments: i) A vector of three parameters θ = (θ1, θ2, θ3). ii) Two predictor vectors, x1 = (x1,1, …, xn,1) and x2 = (x1,2, …xn,2) and returns a length n vector corresponding to p(x11, p12), …p(xn1, xn2) for the corresponding θ values. Hint: You can do this without loops by subscripting for θ and using vectorized calculations for x1 and x2.

```
log_reg <- function(theta, x1, x2){
  output <- c()
  for(i in seq_along(x1)){
      p <- 1/(1 + exp(-x1[i]*theta[1] - x2[i]*theta[2] - theta[3]))
      output <- c(output, p)
  }
  return(output)
}

#testing
#theta <- c(0,1,1)
#x1 <- c(1,9)
#x2 <- c(1,1)
#log_reg(theta,x1,x2)
```

(b)

(b) Write a function to compute L(θ1, θ2, θ3) for n observations which takes as arguments:

i) A vector of three parameters θ = (θ1, θ2, θ3).

ii) Two predictor vectors, x1 = (x1,1, …, xn,1) and x2 = (x1,2, …xn,2)

iii) An outcome vector, y = (y1, …, yn) Hint: Use your function p(x1, x2) from part (a).

```
cross_entropy_loss <- function(theta, x1, x2, y){
  p <- log_reg(theta, x1, x2)
  output <- 0
  for(i in seq_along(y)){
    one_term <- y[i] * log(p[i], base = exp(1)) + (1-y[i]) * log(1-p[i], base
 = exp(1))
    output <- output - one_term
  }
  return(output)
}

#testing
#theta <- c(0,1,1)
```

```
#x1 <- c(1,9)
#x2 <- c(1,1)
#y <- c(1,0.7)
#cross_entropy_loss(theta,x1,x2,y)
```

(c)

(c) Fit a logistic regression classifier to the HTRU2 data, choosing Y to be the Class values (coded as 0 and 1), X1 to be the Mean IP values and X2 to be the Mean DMSNR values using the optim() function in R. Using optim and your loss function from part (b), find the values of theta[1], theta[2], theta[3] that minimize the cross-entropy loss. Report your estimates for (theta1.theta2, theta3) and the estimated loss (and be sure to include the code that allowed you to achieve it). Note, you do not need to write a new function to do this with associated arguments, you simply can write a block of R code accomplishes the task. Starting optim at par=c(0,0,0) works well this model.

```
library(readr)

loss_cal <- function(col1,col2){
  HTRU <- read_csv("D:/PERSONAL/McGill/McGill/McGill Current/MATH 208/assignments/A3/HTRU_2.csv",col_names = FALSE)
  names(HTRU) = c("Mean_IP", "SD_IP", "EK_IP", "SKW_IP","Mean_DMSNR", "SD_DMSNR", "EK_DMSNR", "SKW_DMSNR","Class")
  x1_data <- c(HTRU[[col1]])
  x2_data <- c(HTRU[[col2]])
  y_data <- c(HTRU$Class)
  result<-optim(par=c(0,0,0), fn=cross_entropy_loss, x1=x1_data, x2=x2_data, y=y_data)
}

result <- loss_cal(1, 5)

result

## $par
## [1] -0.10569326  0.01629013  7.28979911
##
## $value
## [1] 1991.015
##
## $counts
## function gradient
##      218       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
# Theta: -0.10569326  0.01629013  7.28979911
# Value: 1991.015
```

(d)

(d) For this part, you should write code using a for loop (or loops) to compute the minimized cross-entropy loss for each possible pair of predictors for the HTRU2 data (note there are 28 possible models) and then store the results in a tibble with each row containing the names of the two variables used in the modelling and their cross-entropy loss). You can then arrange the rows by the value of the loss to find create a table ordered from best pairs of predictors to worst pairs according to estimated loss. Display your ordered table using the kable(.) function. Include all the code used to generate your results. Note: starting optim at par=c(0,0,0) actually works well in all 28 models (this will not always be the case!). Hint: I found it easiest to first use the combn() function to generate a 2 × 28 matrix where the columns contain all possible pairs pairs of names.

```
library(tidyverse)
HTRU <- read_csv("D:/PERSONAL/McGill/McGill/McGill Current/MATH 208/assignments/A3/HTRU_2.csv",col_names = FALSE)
names(HTRU) = c("Mean_IP", "SD_IP", "EK_IP", "SKW_IP","Mean_DMSNR", "SD_DMSNR", "EK_DMSNR", "SKW_DMSNR","Class")
var_combs<-combn(names(HTRU[,-9]),2)

res = NULL


for(i in seq_along(names(HTRU[,-8]))){
  for( j in 1:8){
    if(j<=i){
      next
    }
    if(i == 8 & j == 8){
      break
    }
    cross_result <- loss_cal(i,j)
    cross_val <- cross_result$value
    little_tb <- tibble(
      "col1" = names(HTRU)[i],
      "col2" = names(HTRU)[j],
      "cross entropy loss" = cross_val
    )
    res = bind_rows(little_tb,res)
  }
}

library(knitr)
output_tibble <- res[order(res$`cross entropy loss`),]
kable(output_tibble)
```

col1	col2	cross entropy loss
EK_IP	SD_DMSNR	1427.745
EK_IP	EK_DMSNR	1429.591
EK_IP	SKW_DMSNR	1434.257
EK_IP	SKW_IP	1450.829
Mean_IP	EK_IP	1483.505
SD_IP	EK_IP	1490.764
EK_IP	Mean_DMSNR	1502.008
Mean_IP	SD_DMSNR	1759.214
Mean_IP	EK_DMSNR	1763.425
Mean_IP	SKW_DMSNR	1790.573
SKW_IP	SD_DMSNR	1834.243
SKW_IP	EK_DMSNR	1839.221
SKW_IP	SKW_DMSNR	1875.364
Mean_IP	SKW_IP	1918.023
Mean_IP	Mean_DMSNR	1991.015
SKW_IP	Mean_DMSNR	2021.685
Mean_IP	SD_IP	2052.101
SD_IP	SKW_IP	2305.642
SD_IP	EK_DMSNR	2777.460
SD_IP	SD_DMSNR	2877.531
SD_IP	SKW_DMSNR	2953.056
SD_IP	Mean_DMSNR	3365.135
Mean_DMSNR	EK_DMSNR	3772.916
SD_DMSNR	SKW_DMSNR	3800.222
EK_DMSNR	SKW_DMSNR	3808.527
SD_DMSNR	EK_DMSNR	3809.508
Mean_DMSNR	SKW_DMSNR	3869.097
Mean_DMSNR	SD_DMSNR	3971.733

(e)

(e) Finally, produce the same tibble as in part (d), only using the var_combs matrix above and map_dfr(.). Hint: You may find it useful to convert var_combs to a data.frame or tibble first.

```
library(readr)
library(dplyr)
library(purrr)
HTRU <- read_csv("D:/PERSONAL/McGill/McGill/McGill Current/MATH 208/assignmen
```

```
ts/A3/HTRU_2.csv",col_names = FALSE)
names(HTRU) = c("Mean_IP", "SD_IP", "EK_IP", "SKW_IP","Mean_DMSNR", "SD_DMSNR
", "EK_DMSNR", "SKW_DMSNR","Class")
var_combs<-combn(names(HTRU[,-9]),2) ## -9 excludes the 9th column, the Class
 variable

loss_cal2 <- function(cols){
  x1_data <- c(HTRU[[toString(cols[[1]])]])
  x2_data <- c(HTRU[[toString(cols[[2]])]])
  y_data <- c(HTRU$Class)
  result2<-optim(par=c(0,0,0), fn=cross_entropy_loss, x1=x1_data, x2=x2_data,
 y=y_data)

  little_tb <- tibble(
      "col1" = cols[[1]],
      "col2" = cols[[2]],
      "cross entropy loss" = result2$value
    )

  return(little_tb)
}

#result2 <- loss_cal2("X1", "X2")
var_df <- as.data.frame(var_combs)
#c1 <- droplevels(var_df[[1,1]])
#HTRU[[c1]]

result_final <- map_dfr(var_df,loss_cal2)

library(knitr)
output_tibble2 <- result_final[order(result_final$`cross entropy loss`),]
kable(output_tibble2)
```

col1	col2	cross entropy loss
EK_IP	SD_DMSNR	1427.745
EK_IP	EK_DMSNR	1429.591
EK_IP	SKW_DMSNR	1434.257
EK_IP	SKW_IP	1450.829
Mean_IP	EK_IP	1483.505
SD_IP	EK_IP	1490.764
EK_IP	Mean_DMSNR	1502.008
Mean_IP	SD_DMSNR	1759.214
Mean_IP	EK_DMSNR	1763.425
Mean_IP	SKW_DMSNR	1790.573
SKW_IP	SD_DMSNR	1834.243

SKW_IP	EK_DMSNR	1839.221
SKW_IP	SKW_DMSNR	1875.364
Mean_IP	SKW_IP	1918.023
Mean_IP	Mean_DMSNR	1991.015
SKW_IP	Mean_DMSNR	2021.685
Mean_IP	SD_IP	2052.101
SD_IP	SKW_IP	2305.642
SD_IP	EK_DMSNR	2777.460
SD_IP	SD_DMSNR	2877.531
SD_IP	SKW_DMSNR	2953.056
SD_IP	Mean_DMSNR	3365.135
Mean_DMSNR	EK_DMSNR	3772.916
SD_DMSNR	SKW_DMSNR	3800.222
EK_DMSNR	SKW_DMSNR	3808.527
SD_DMSNR	EK_DMSNR	3809.508
Mean_DMSNR	SKW_DMSNR	3869.097
Mean_DMSNR	SD_DMSNR	3971.733