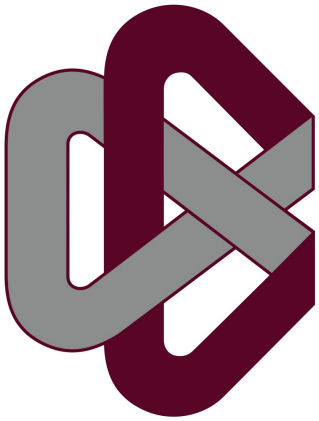
Ciencia de Datos

Tarea 2

Ejercicio 3



CIMAT

Miranda Belmonte Hairo Ulises

hairo.miranda@cimat.mx

Institución: Centro de Investigación en Matemáticas Unidad Monterrey

Dirección: Alianza Centro 502, 66629 N.L

Programa: Maestría en Computo Estadístico

Fecha: 15 de Febrero del 2019

Clase: Computo Estadístico

Contenido

Ciencia de Datos

Tarea 2

Ejercicio 3a

Clasificador de imágenes utilizando PCR

Miranda Belmonte Hairo Ulises

Resumen

Se utilizan cuatro base con imágenes de dígitos del 0 al 9, dos bases para entrenar el modelo y dos para probarlo. El clasificador se realiza utilizando regresión con componentes principales.

1 Problema

a) Implementa un clasi

cador para las imagenes que pertenecen a uno de los $k = 1al9$. Utiliza los datos de mnistXtrain.dat para ajustar el modelo y mnistXtest.dat para probarlo. Obten el error obtenido, tanto en los datos de entrenamiento como los de prueba, usando diferentes valores de p componentes principales. Realiza una gráfica de error vs p. ¿ Qué valor de p recomendarías usar? Nota: Puedes usar la función general para modelos lineales lm() de R, la cual puede usarse también para regresión lineal multivariada. Revisa la ayuda de la función.

2 Descripción de la base

Se utilizan cuatro bases en formato dat, con dígitos escritos a mano, digitalizados y normalizados de 28x28, en los datos que se presentan en mnistXtrain.dat y mnistXtest.dat se encuentran estos dígitos, El archivo mnistXtrain.dat se utiliza para entrenar el modelo, cuenta con 784 columnas (pixeles) y $6,000$ filas (imagenes), el archivo mnistYtrain.dat es un vector columna con $6,000$ registros que contienen las etiquetas de la base mnistXtrain.dat. El archivo mnistXtest.dat contieen los datos de prueba para el modelo, y cuenta con $10,000$ filas, con 748 columnas, su estiquetas se obtienen de se mnistYtest.dat. Las imagenes se encuentran estandarizadas y no cuentan con valores perdidos.

3 Descripción de la estimación

Se utiliza el software R (versión 3.5.2) para calcular todos los resultados que se presentan a continuación. Se trabaja con la paqueteria *princomp*, de la librería *stats* en R, el cual realiza el calculo de los componentes principales, está función se utiliza al conjunto de observaciones de entrenamiento y prueba (mnistXtrain.dat y mnistXtest.dat), y se extraen las proyecciones definidas como :

$$Ztrain_p = \phi Xtrain$$

con $Ztrain_p$ como los p-esimos scores o proyecciones, ϕ e vector de rotación o loadings, y $Xtrain$ la matriz de observaciones de entrenamiento.

Se realiza la regresión y estimación de coeficientes para los datos de entrenamiento utilizando la función lm; la variable dependiente es la matriz de ceros y unos que se construyen con mnistYtrain.dat, de acuerdo a la especificación de la tarea, y las covariables son los scores de mnistXtrain.dat. Se realiza la predicción de la siguiente forma:

$$\hat{Ytrain} = Ztrain_p\hat{B}$$

con $/B$ matriz de coeficientes estimados y $\hat{Ytrain}$ la predicción.

Se calcula el porcentaje de error de predicción como el número de veces que $\hat{Ytrain}$ sea igual al de la base mnistYtrain.dat; se gráfica cada uno de los errores para un cierto número de componente p y se selecciona aquel componente que proporcione un de error de prediccion pequeño; por último, se utilizan los coeficientes estimados, con el modelo seleccionado, se multiplican respecto al producto de los vector de pesos d respecto a los datos de prueba

$$Ztest_p = \phi Xtrain$$

$$\hat{Yest} = Ztest_p\hat{B}$$

Una vez realizado el cálculo, se gráfica cada uno de los errores para un cierto número de componente p y se selecciona aquel componente que proporcione un de error de prediccion pequeño. El error se obtiene como el número de veces que $\hat{Ytest}$ sea igual al de la base mnistYtest.dat

4 Resultados

A continuación se presentan los resultados de los errores de predicción para p componentes principales. En la figura 1, se tienen los errores de pronostico para los primeros 100 componentes. La linea azul son los errores de pronostico de los datos de prueba y los de rosa de entrenamiento; se observar es con 25 componentes para los datos de

entrenamiento, el error de predicción es menor al 20%; por otro lado, en los errores de prueba, se observa que al utilizar más de 50 componentes, el modelo se encuentra sobre entrenado.

Figure 1: Errores de predicción, 0 a 100 componentes

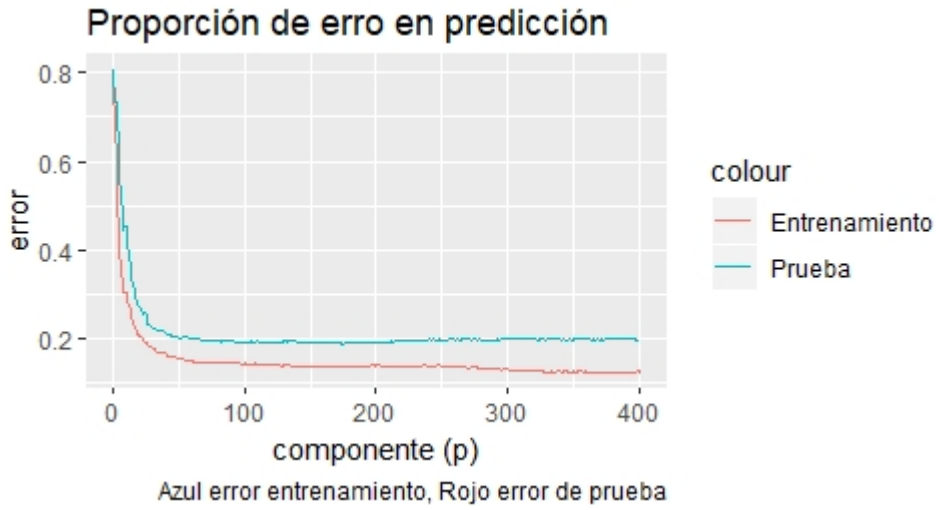
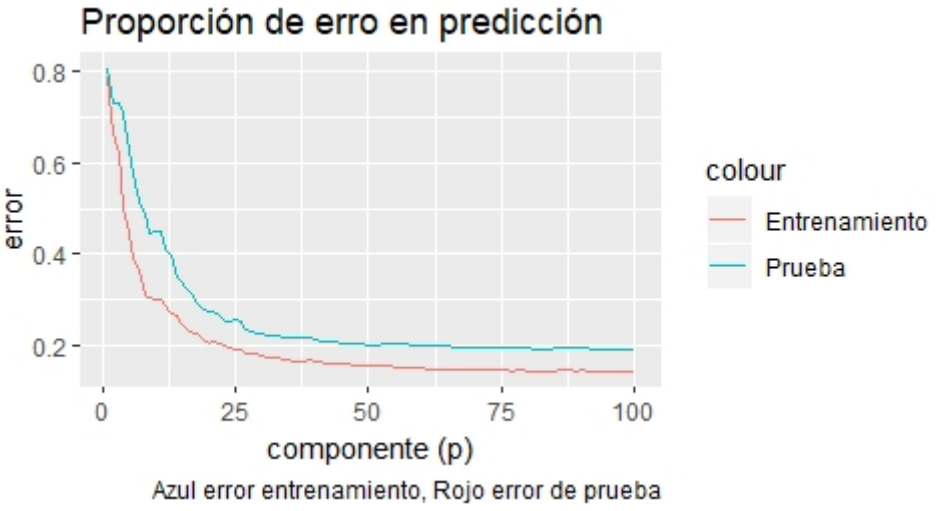


Figure 2: Errores de predicción, 0 a 50 componentes



De esta manera, se sugiere utilizar entre 20 a 25 componentes, con el fin de registrar un error de predicción menor al 40%, y obtener un modelo parsimonioso.

5 Función

```
regresion <- function(j){
entrenamientoReg <-lm(ceroMatrix ~ entrenarScores[,1:j]-1, data=as.data.frame(entren
entrenamientoCof <- entrenamientoReg$coefficients

# realiza predicción
entrenamientoY <- entrenarScores[,1:j] %*% entrenamientoCof
# toma valor mayor en la fila
entrenamientoYmayor <- apply(entrenamientoY, 1, which.max)
# convertir a matriz
entrenamientoYmayor <- entrenamientoYmayor %>%  as.matrix

# Proporción de erro, datos de entrnamiento
entrenamientoError <- 1-mean((entrenamientoYmayor-1)==Ytrain)

# datos de prueba
loadin <- entrenar$loadings[,1:j]
xtest_loading <- as.matrix(Xtest)%*%loadin
pruebaY <- xtest_loading%*% entrenamientoCof # coeficientesde entrenamiento



# selecciona el valor mayor
pruebaYmayor <- apply(pruebaY, 1, which.max)

pruebaYmayor <- pruebaYmayor %>%  as.matrix
pruebaYmayor %>%  dim
a <-Ytest$x
# proporción de error
pruebaError <- 1-mean((pruebaYmayor-1)==a)
```

```
return(c(entrenamientoError,pruebaError,j))
}# end regresion
```

Ciencia de Datos

Tarea 2

Ejercicio 3b

Aplicación en Shiny: clasificador

Miranda Belmonte Hairo Ulises

Resumen

Se utiliza el clasificador del ejercicio 3a, con el fin de realizar una aplicación que captures dígitos del 0 al 9, y sea capaz de predecirlos.

6 Ejercicio 3b Aplicación Clasificador

Se modifica los archivos los tres archivos que se brindaron en la clase para el desarrollo de la aplicación; el código se presenta a continuación.

código general.r

Instrucciones para correr la aplicación.

```
######################
## Librerias
######################
library("tidyverse")
library("magrittr")

setwd("C:/Users/h_air/Desktop/CIMAT MCE/Semestre_2/Ciencia de Datos/Tareas/Tarea 2-2
getwd()

########################################
## Nota, el clasificador es del 0 al 9
########################################



######################
## Importando datos
######################
Xtrain <- read.table("mnistXtrain.dat", header = T, sep = "")
Xtrain %>% dim
Xtest <- read.table("mnistXtest.dat", header = T, sep = "")
Xtest %>% dim
Ytrain <- read.table("mnistYtrain.dat", header = T, sep = "")
Ytrain %>% dim
```

```
Ytest <- read.table("mnistYtest.dat", header = T, sep = "")
Ytest %>% dim
######################
## Guardando PCA
######################
datos <- prcomp(Xtrain) # componentes principales
saveRDS(datos, file = "my_data.rds")

######################
## Clasificador
######################

# Matriz clasificadora
ceroMatrix <- matrix(0L, dim(Xtrain)[1],10)
for (i in 1:dim(Xtrain)[1]){
  ceroMatrix[i,Ytrain[i,1]+1] <-1
} # end for

# rexportar pca
pc <- readRDS("my_data.rds")

# Seleccionar número de componentes
j <- 15
# scores
entrenarScores <- pc$x
# regresión
entrenamientoReg <-lm(ceroMatrix ~ entrenarScores[,1:j]-1, data=as.data.frame(entren
# coeficientes
entrenamientoCof <- entrenamientoReg$coefficients
# loadings
load <- pc$rotation[,1:j]
```

```
## incluye la ruta donde esta la carpeta con archivos server y ui
# corra aplicación
runApp(appDir="C:/Users/h_air/Desktop/CIMAT MCE/Semestre_2/Ciencia de Datos/Tareas/T
```

ui.r

Código para mostrar la entrada de datos y salida

```
################################
### minimal example - ui.R ###
################################
library("shiny") # load shiny at beginning at both scripts
library("pixels")

shinyUI(fluidPage(
    titlePanel("Pixels"),

    fluidRow(
        column(5,
               titlePanel("Captura de digitos (simplificada)"),
               shiny_pixels_output(outputId="pixels"),
               actionButton("captureDigit", "Capturar")
               ),

        column(5,
               tableOutput(outputId="table1")

               ##plotOutput(outputId="plot1")
               )
    )
```

```
))
```

server.r

Función que realiza la predicción

```
library("shiny")
library("pixels")
library("OpenImageR")
##library(ripa)

shinyServer(function(input, output) {
output$pixels <- shiny_render_pixels(
 show_pixels(grid=c(28,28),
    brush = matrix(c(0, 0.5, 0.8, 0.5, 0,
      0.5, 1, 1, 1, 0.5,
      0.8, 1, 1, 1, 0.8,
      0.5, 1, 1, 1, 0.5,
      0, 0.5, 0.8, 0.5, 0), 5, 5)) )
  output$prompt <- renderText("Dibuja un numero")

  observeEvent(input$captureDigit, {
    dig <<- NormalizeObject(as.numeric(input$pixels))
    #dig <<- NormalizeObject(as.numeric(Xtest[1,]))
    ## realiza ciertas rotaciones al digito escaneado,
    ## porque note que al graficarlo, esta en otra
    ## orientacion
    data.digit <- matrix(dig,ncol=28,nrow=28,byrow=T)
    temp.dig <- flipImage(rotateFixed(data.digit,180))
    tr.dig <- as.numeric(temp.dig)
```

```
    ##plot.img(tr.dig)

    ## limpiar el grid
    output$pixels <- shiny_render_pixels(
      show_pixels(grid=c(28,28),
 brush = matrix(c(0, 0.5, 0.8, 0.5, 0,
 0.5, 1, 1, 1, 0.5,
 0.8, 1, 1, 1, 0.8,
 0.5, 1, 1, 1, 0.5,
 0, 0.5, 0.8, 0.5, 0), 5, 5))
    )

    ## reescala el digito escrito
    tt <- scale.default(t(tr.dig),pc$center,pc$scale)
    ## obtiene scores (proyecciones en los componentes prncipales)
    proj <- tt%*%load
    #
    pruebaY <- proj%*% entrenamientoCof # coeficientesde entrenamiento
    pruebaYmayor <- apply(pruebaY, 1, which.max)

    ## graficar
output$table1 <-
renderTable(data.frame(Prediccion = isolate(pruebaYmayor)),bordered = T,
 striped = T, hover = T,
spacing =  "l",
 rownames = F, caption = paste0("Con ",j," componentes"))
  })
})
```

imagenes

A continuación se presenta una serie de imágenes con las predicciones realizadas por la aplicación. En la figura 3, se realiza la predicción con 15 componentes; en la figura 4, con 20; en la figura 5, con 25; y por último, en la figura 6, con 100 componentes principales. De esta manera, como se recomendó en el ejercicio anterior, se deben utilizar los componentes que registrarán un error de predicción bajo, en este caso, un modelo entre 15 a 25 componentes.

Figure 3: Predicción con 15 componentes

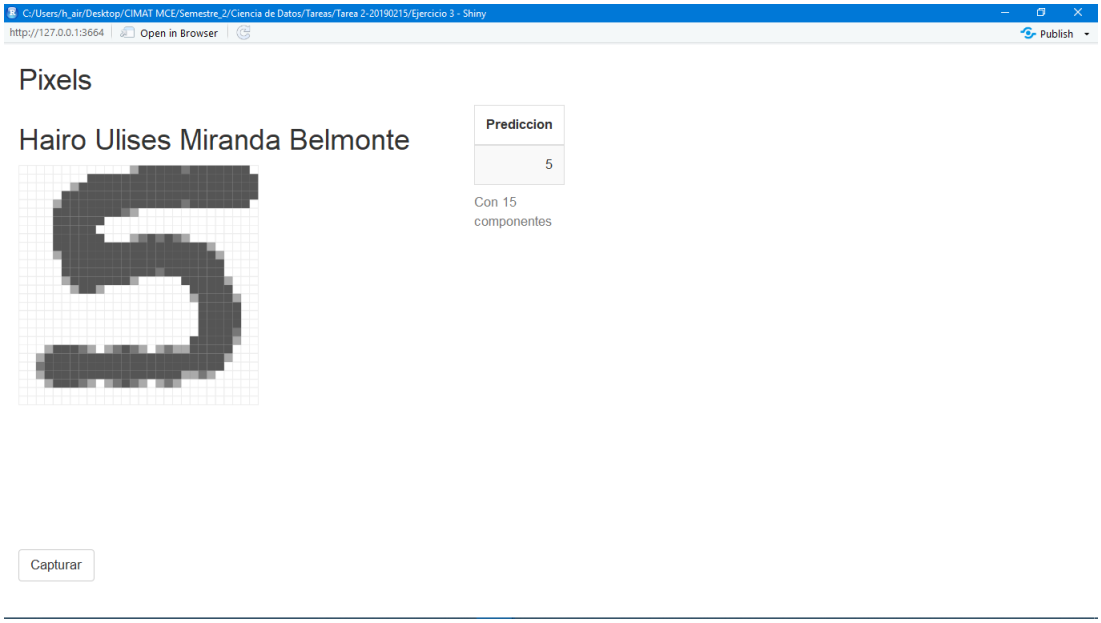


Figure 4: Predicción con 20 componentes

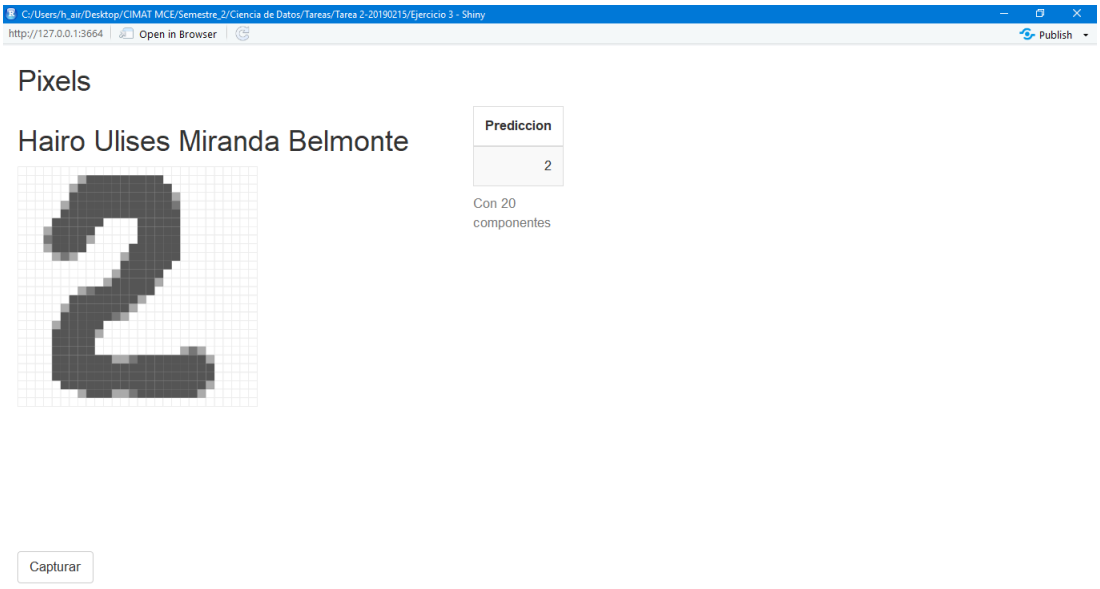


Figure 5: Predicción con 25 componentes

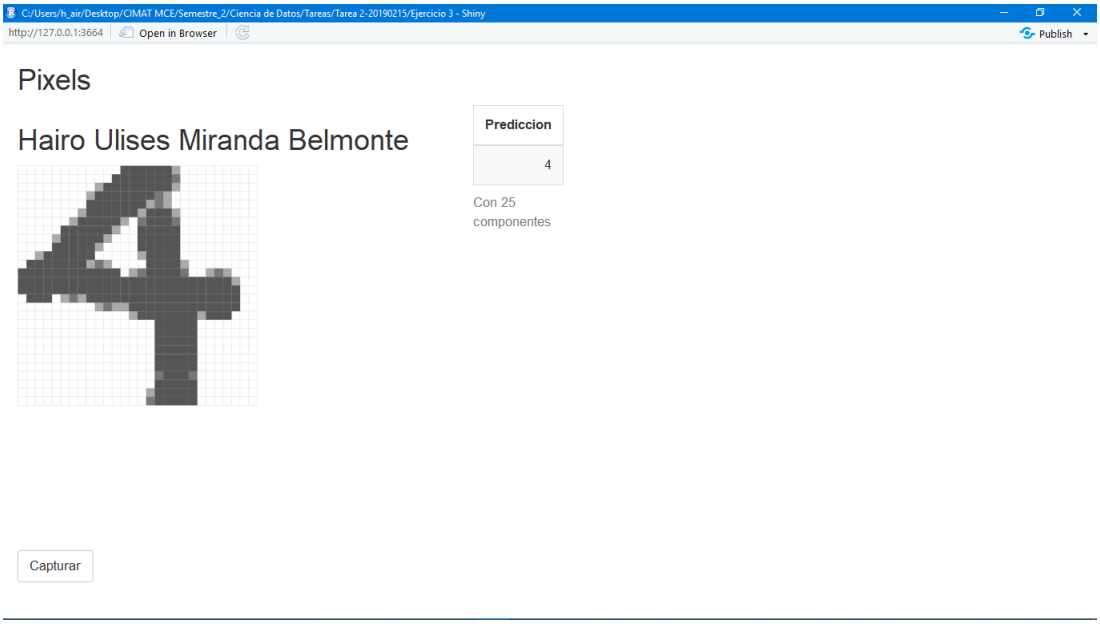


Figure 6: Predicción con 100 componentes