

Razonamiento bajo incertidumbre

Tarea 5. Agrupamiento de colores en imágenes

Ángel García Báez

2024-10-15

Índice

1 Instrucciones:	2
2 Definiciones necesarias antes de comenzar	3
2.1 Vector de medias:	3
2.2 Matriz de Varianzas y Covarianzas	3
2.3 Distribución de probabilidad normal multivariante	3
3 Idea del K-medias	4
4 Idea del modelo de mezcla de Gaussianas	5
5 Imágenes a utilizar y forma de abordar el problema	6
6 Resultados	7
6.1 Asignación del k óptimo para la imagen 1.	7
6.2 Asignación del k óptimo para la imagen 2.	8
6.3 Asignación del k óptimo para la imagen 3.	9
6.4 Resultados de K-medias y GMM para la imagen 1 (Denisse Guerrero)	10
6.5 Resultados de K-medias y GMM para la imagen 2 (Ana Sofía)	11
6.6 Resultados de K-medias y GMM para la imagen 3 (Selena Quintanilla)	12
7 Conclusiones	13
8 Bibliografía	14
9 Anexos	14
9.1 Código en Julia del GMM	14
9.2 Código en R de todo el proceso:	15

1 Instrucciones:

Dada una imagen digital que va a ser denotada por \vec{x} y a partir de un valor k proporcionado por el usuario:

- 1.- Realizar la segmentación de la imagen en k grupos usando mezclas gaussianas.
- 2.- Realizar la segmentación de grupos usando k-medias.
- 3.- Para ambos cosas se debe probar con almenos 3 imágenes y 3 valores distintos de K .
- 4.- Proponer una forma en que se pueda proporcionar el numero de k -grupos de manera automática mediante: el gráfico de codos de Janbu, la estimación por máxima verosimilitud, usando el criterio AIC o alguna otra propuesta.

2 Definiciones necesarias antes de comenzar

2.1 Vector de medias:

El vector de medias para una matriz sera definido como un vector fila de tamaño $1 \times P$ donde P es la cantidad de columnas que tenga la matriz de la que se quiere obtener.

$$\bar{x} = [\bar{x}_1 + \bar{x}_2 + \cdots + \bar{x}_p]$$

2.2 Matriz de Varianzas y Covarianzas

La forma de calcular la matriz de varianzas y covarianzas de la matriz de datos, puede resumirse en la siguiente expresión:

$$\Sigma = \frac{1}{n-1} (X^T X - n\bar{x}^T \bar{x})$$

Donde:

Σ = Matriz de varianzas y covarianzas

X = Matriz de datos

X^T = Matriz de datos transpuesta

n = Filas o casos de la matriz

\bar{x} = Vector fila de las medias

\bar{x}^T = Vector fila de las medias transpuesto

2.3 Distribución de probabilidad normal multivariante

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{P/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \bar{\mathbf{x}})^T \Sigma^{-1} (\mathbf{x} - \bar{\mathbf{x}})\right)$$

Donde:

P = Cantidad de variables.

X = Vector de datos de tamaño $1 \times P$.

\bar{x} = Vector de medias de tamapo $1 \times P$.

Σ = Matriz de varianzas y covarianzas.

Σ^{-1} = Inversa de la matriz de varianzas y covarianzas.

$|\Sigma|$ = Determinante de la matriz de varianzas y covarianzas.

3 Idea del K-medias

De acuerdo con lo que explica Bishop (2006), el K-medias puede entenderse como un algoritmo de aprendizaje no supervisado el cual tiene como propósito generar variables latentes categóricas o etiquetar al conjunto de datos dada su similitud en K posibles grupos formados alrededor de K posibles centroides.

Para poder inicializar el algoritmo, es necesario proporcionarle justamente un valor de K para formar esa posible cantidad de grupos con los datos de los que se dispone pero esto representa un problema en si mismo dado que no hay una manera de saber cuantos grupos estan presentes en los datos. Para poder abordar esto existen varios enfoques, como por ejemplo, probar y ver con cuantos K se logra minimizar la función de costo o la distancia entre grupos, probar a ver cual da un mayor valor de AIC, BIC o la mayor verosimilitud posible, entre otros más criterios que se han propuesto a lo largo del tiempo. Para el enfoque de este trabajo se decidió quedarse con con el criterio de la distancia entre grupos para determinar el numero de K óptimo de grupos.

Ahora sí, partiendo de un conjunto de datos con N filas y P variables **Continuas** se desea etiquetar a cada fila dado un valor K propuesto, para ello se siguen los siguientes pasos del algoritmo:

Paso 1: Se toman K sujetos aleatorios del conjunto de datos que van a ser los centroides μ_k iniciales.

Paso 2: Se define una medida de distancia que es la norma al cuadrado de la diferencia entre el punto n -ésimo y el centroide k -ésimo como sigue:

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|x_n - \mu_k\|^2$$

donde r_{nk} es una variable para identificar la asignación del punto n -ésimo al grupo k -ésimo si la distancia a dicho grupo es la mínima con respecto a todos los demás posibles grupos.

Paso 3: Con los datos etiquetados en el paso 2 se toman para actualizar el valor de los centroides del siguiente modo:

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum r_{nk}}$$

Paso 4: Se verifica la convergencia con la medida J , si $J^{(i)} = J^{(i-1)}$ entonces se detiene el proceso, si no es el caso, se repiten los pasos 2 y 3 hasta lograr la convergencia.

De esta sencilla forma, se puede aplicar el método de las k – medias para la segmentación de datos.

4 Idea del modelo de mezcla de Gaussianas

El modelo de mezclas Gaussianas comienza por pensar a los datos como una mezcla de K posibles distribuciones gaussianas que pueden ser descompuestas para poder hallar el grupo al cual pertenece cada dato del conjunto.

Dicha expresión de K posibles distribuciones gaussianas se puede expresar como sigue:

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

en donde se deben respetar las siguientes restricciones:

$$0 \leq \pi_k \leq 1, \quad \sum_{k=1}^K \pi_k = 1$$

El objetivo principal de dicho modelo de mezclas, es lograr captar la variabilidad de los grupos de datos, ademas de tratar de describirlos únicamente mediante sus medias.

A continuación se describen los pasos para llevar a cabo dicho proceso:

Paso 1: Establecer los parámetros iniciales de μ_k, Σ_k, π_k y k para inicializar los valores del algoritmo, el libro de Bishop (2006) recomienda que estos valores se obtengan después de realizar un $K - \text{medias}$ y se evalua la log-verosimilitud definida por:

$$\ln(p(X|\mu, \Sigma, \pi)) = \sum_{n=1}^N \ln \left(\sum_{k=1}^K (\pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)) \right)$$

Paso 2: En este paso se busca calcular el valor **esperado** dados los actuales parámetros para las responsabilidades de pertenecer a una u otra distribución Gaussiana de acuerdo a la siguiente formula:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

Paso 3: Dado que se busca **máximizar** la verosimilitud, se re-estiman los parámetros usando las responsabilidades o ponderaciones obtenidas en el **paso 2** como sigue:

$$\begin{aligned} \boldsymbol{\mu}_k^{\text{nuevo}} &= \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \\ \boldsymbol{\Sigma}_k^{\text{nuevo}} &= \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{nuevo}}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{nuevo}})^{\top} \\ \pi_k^{\text{nuevo}} &= \frac{N_k}{N} \end{aligned}$$

Donde:

$$N_k = \sum_{n=1}^N \gamma(z_{nk}).$$

Paso 4: Evaluar nuevamente la log-verosimilitud, si $\log(\mathcal{L}(\theta))^{(i)} = \log(\mathcal{L}(\theta))^{(i-1)}$ entonces para el algoritmo, si no es el caso, el algoritmo vuelve a repetirse desde el paso 2.

5 Imagenes a utilizar y forma de abordar el problema

A continuación se muestra una selección de 3 imágenes donde aparecen 3 personas de distintas. En cada imagen se puede apreciar un contraste entre los tonos del fondo, de la persona y de la ropa de la persona.



Table 1: Imagenes para el desarrollo del ejercicio.

Las 3 imágenes tienen características distintas en cuanto a composición de colores y la idea es poder segmentarlas mediante el método de $k - medias$ y el modelo de mezcla de Gaussianas.

Para poder llevar a cabo el desarrollo de este ejercicio se utilizo el lenguaje R para trabajar las imágenes y el $k - means$ y el lenguaje Julia para el algoritmo de mezcla de Gaussianas debido a su complejidad y poder de computo que exigia para lograr encontrarlo.

El plan para abordar el problema fue el siguiente:

Paso 1: Cargar una imagen en lenguaje R y descomponerla en 3 vectores (1 por cada canal de color R,G y B) que posteriormente serán unidos en un solo dataframe, al cual ademas se le hará el etiquetado de las coordenadas de cada pixel (fila del dataframe) para su posterior reconstrucción y manipulación.

Paso 2: Se determino el número de grupos óptimos para cada imagen mediante el criterio de la distancia entre grupos para cada imagen haciendo uso de su respectivo gráfico de codos de Janbu.

Paso 3: Una vez determinado el K óptimo para cada imagen (en todas fue 3), se decidió sugerir como posibles valores de K el 5 y el 10 para observar los cambios en la segmentación por imagen.

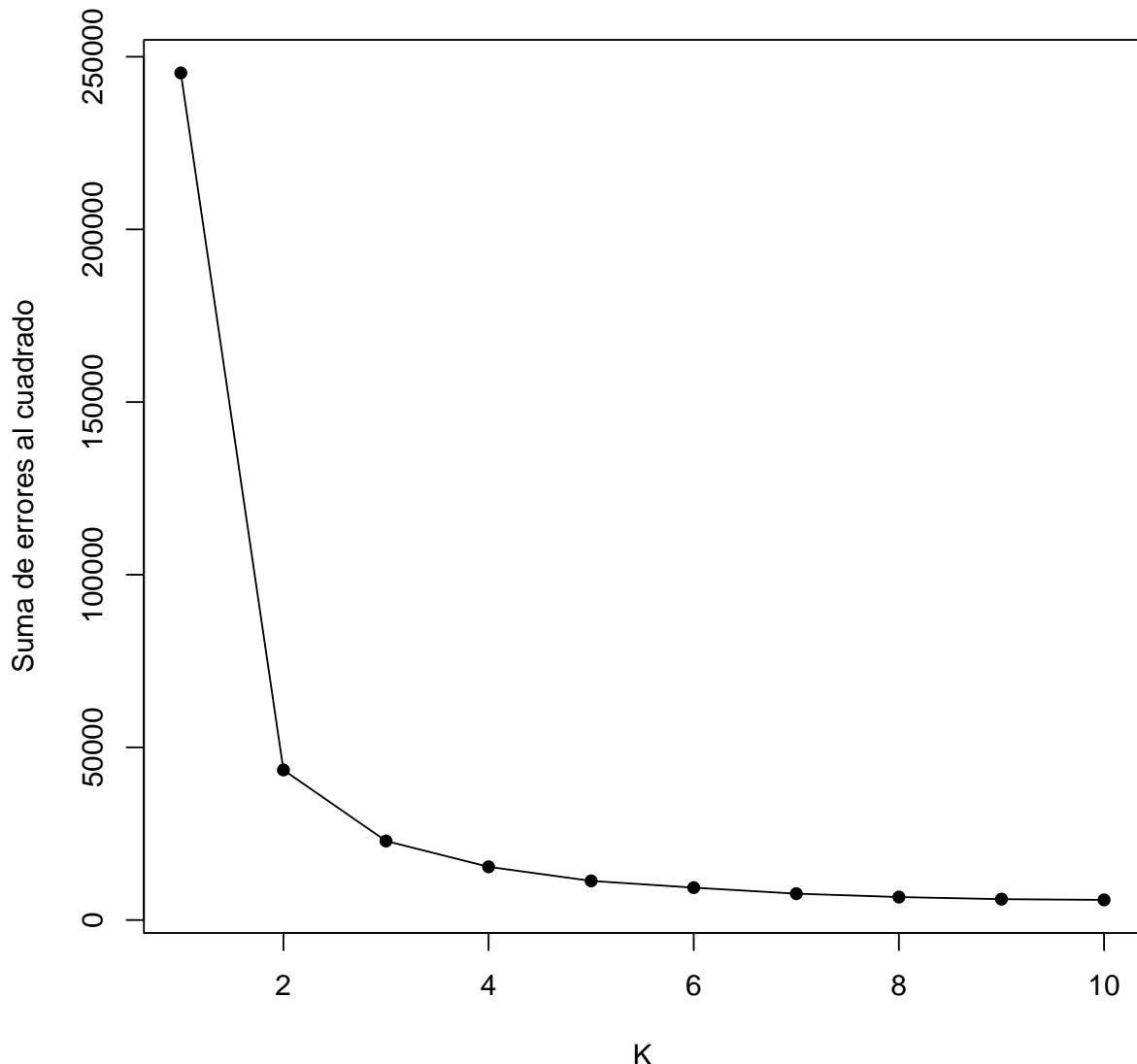
Paso 4: En el paso 4 se utilizaron las funciones que ya venían por defecto en R para realizar el calculo del algoritmo $k - medias$ para los 3 posibles valores de k especificados en el **paso 3**. Se fijo una semilla (2024), se corrió el algoritmo para cada valor de K y para cada imagen para posteriormente, proyectar los resultados de la segmentación en cada caso pintando los pixeles del mismo grupo acorde con el centroide del grupo.

Paso 5: Para calcular la mezcla de Gaussianas, se tomaron los mismos K propuestos en el **paso 3** para poder correr dicho algoritmo pero en lenguaje Julia debido a su costo computacional, se fijo una semilla (2024), se obtuvieron las variables latentes (la asignación de los grupos) y se proyectaron los resultados de la segmentación, pintando los pixeles del mismo grupo acorde con el vector de medias del grupo.

6 Resultados

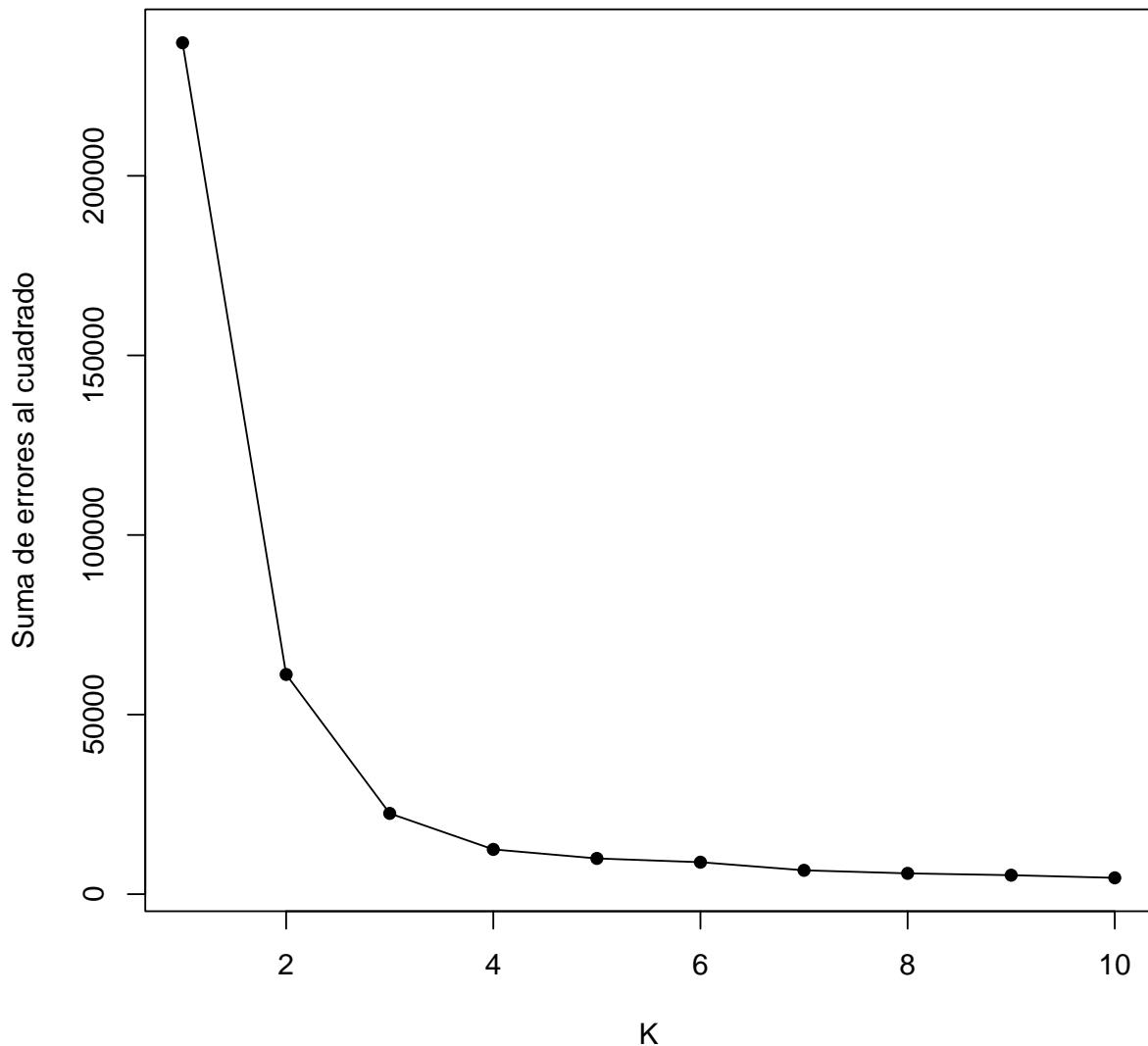
6.1 Asignación del k óptimo para la imagen 1.

A continuación se muestra el gráfico de codos de Janbu para la imagen 1.



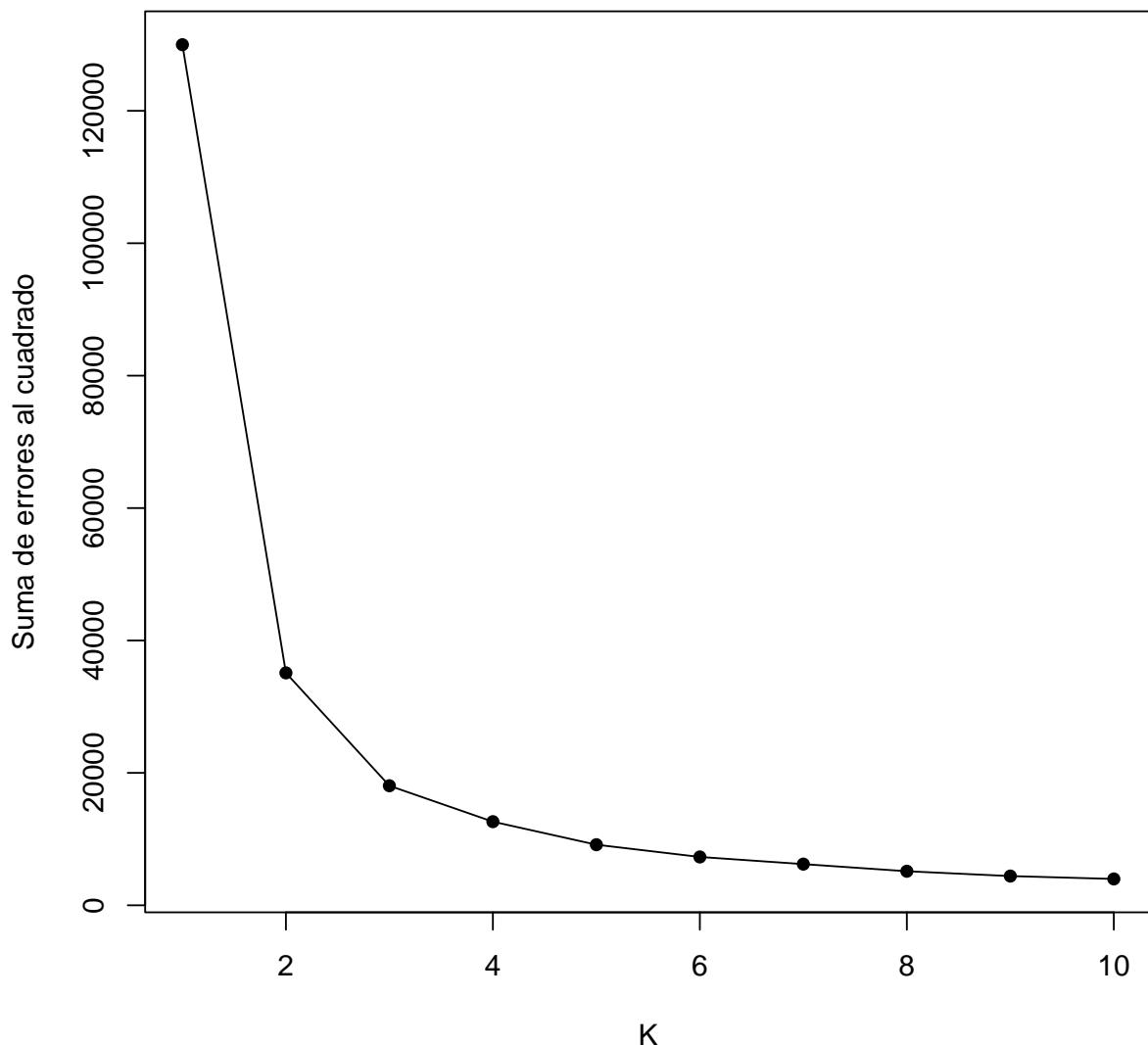
6.2 Asignación del k óptimo para la imagen 2.

A continuación se muestra el gráfico de codos de Janbu para la imagen 2.



6.3 Asignación del k óptimo para la imagen 3.

A continuación se muestra el gráfico de codos de Janbu para la imagen 3.



6.4 Resultados de K-medias y GMM para la imagen 1 (Denisse Guerrero)



Table 2: Comparación de los resultados de K-medias y GMM con $K = 3, 5$ y 10 respectivamente para la imagen 1.

6.5 Resultados de K-medias y GMM para la imagen 2 (Ana Sofía)



Table 3: Comparación de los resultados de K-medias y GMM con $K = 3, 5$ y 10 respectivamente para la imagen 2.

6.6 Resultados de K-medias y GMM para la imagen 3 (Selena Quintanilla)



Table 4: Comparación de los resultados de K-medias y GMM con $K = 3, 5$ y 10 respectivamente para la imagen 3.

7 Conclusiones

Lo primero que cabe mencionar del presente trabajo son los artificios técnicos que se tuvieron que utilizar para llevar a cabo el procesamiento de las imágenes. Por un lado, se descubrió que el algoritmo implementado en R para el cálculo del *k-medias* hace que R envíe los datos al lenguaje FORTRAN, realice el procesamiento y posteriormente devuelva las etiquetas de los grupos al terminar. Esto fue especialmente sorprendente, ya que al compararlo con la implementación en las paqueterías de Julia, esta última resultaba ser más lenta.

Por otro lado, para la implementación del GMM en R, se encontró una paquetería, pero al correrla con la primera imagen, dicha función tardó alrededor de 15 minutos en etiquetar los píxeles, por lo que se optó por hacer que dicha función se procesara en lenguaje Julia, logrando el etiquetado de los píxeles de la primera imagen en menos de 20 segundos.

Sobre los resultados de las imágenes, se observa que para las imágenes con $k = 3$ el *k – medias* presenta una segmentación relativamente buena, logra segmentar los colores más fuertes de la imagen y nos deja con un bonito retrato casi como si fuera monocromático, sin embargo, el algoritmo de GMM pareciera que suaviza demasiado la imagen y le cuesta lidiar cuando son poquitos grupos.

A medida que aumenta el K para las imágenes, se van pareciendo más entre sí los resultados de ambos métodos hasta el punto de ser prácticamente la misma imagen, sin embargo, a criterio personal tras este trabajo diría que el que mejor desempeño tiene en esta tarea es el algoritmo de *K – medias* puesto que logra hacer un buen trabajo de segmentación y con un menor coste computacional respecto al método de GMM.

8 Bibliografía

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4, No. 4, p. 738). New York: springer

9 Anexos

9.1 Código en Julia del GMM

```
# Cargar las paqueteterias para la mezcla de Gaussianas
# Ajustar el modelo GMM con 3 componentes (ya que sabemos que hay 3 especies en iris)
using DataFrames
using GaussianMixtures
using RDatasets
using Statistics
function gausiano(K,X)
    ### indicarle que K es un entero
    K = Int(K)
    ### Indicarle que X es una Matrix
    X = Matrix(X)
    ### Calcular el modelo de mezcla de Gaussiana
    gmm = GMM(K, X)
    ### Calcular los posteriores
    A = (gmmposterior(gmm,X))
    ### Extraer las etiquetas
    Z = Matrix(A[1])
    etiqueta = [argmax(row) for row in eachrow(Z)]
    return(DataFrame(Etiqueta = etiqueta))
end
```

9.2 Código en R de todo el proceso:

```

rm(list=ls())
### Cargar las librerias
# Cargar las librerias
# Cargar la imagen en R
## Cargar librerias necesarias ##
library(png) # Libreria para leer PNG's
library(flextable) # Libreria para hacer tablas bonitas
library(scatterplot3d) #gráficar en 3D
library(rsvg) # Manipular archivos SVG
library(jpeg) # Comprimir el SVG a JPEG
library(rgl) # OpenGL pero para los gráficos en 3D
library(corrplot) # Para gráficar las correlaciones
library(NbClust) # Correr el Kmedias
library(JuliaCall) # Para pasarme objetos a Julia
##### Función para leer imágenes y mapearlas #####
pixeles = function(imagen){
  # Extraer los datos
  datos = cbind.data.frame(R = as.vector(imagen[ , ,1]), # Extraer inf de R
                            G = as.vector(imagen[ , ,2]), # Extraer inf de G
                            B = as.vector(imagen[ , ,3])) # Extraer inf de B
  # Extraer las coordenadas de cada pixel para re-mapearlos
  #Filas
  nf = dim(imagen)[1]
  #Columnas
  nc = dim(imagen)[2]
  # Hacer el ordenamiento para remapearlos
  X = rep(1:nc,each = nf)
  Y = rep(1:nc,each = nf)
  datos$X = X
  datos$Y = Y
  # Añadir el vector de colores al conjunto de datos
  datos$color = rgb(datos$R,datos$G,datos$B)
  # Regresar el objeto transformado
  return(datos)
}
##### Función para calcular la intravarianza #####
INTRA = function(X){
  # Definir el objeto que va a guardar la intravarianza
  IV = c()
  # Deginir del 1 al 10
  k = 10
  for(i in 1:k){
    # Calcular el Kmedias y extraer la intra varianza
    KM = kmeans(X,i)
    IV[i] = KM$tot.withinss
  }
  # Devolver el vector de las intra varianzas
  plot(IV,type = "o",
        pch = 16, xlab = "K",ylab = "Suma de errores al cuadrado")
}
##### FUnción para sacar colores promedio
promcolor = function(X){

```

```

for(i in 1:length(unique(X[,7]))){
  # Calcular el vector media
  VM = colMeans(X[X[,7]==i ,1:3])
  # Obtener el color promedio
  X[X[,7]==i ,8] = rgb(VM[1],VM[2],VM[3])
}
return(colores = X[,8])
}

##### Sección de K medias para imagen 1#####
ruta1 = "IMAGENES/IM2.jpg"
imagen1 = readJPEG(ruta1) # Descomponer la imagen en sus canales RGB
datos1 = pixeles(imagen1)
dim(imagen1) # La imagen 1 tiene 1152 X 768 = 884,736

##### Definir el criterio de la intra varianza
set.seed(2024)
pdf(file="RESULTADOS/R1.pdf")
INTRA(datos1[,1:3])
dev.off()

##### Kmedias para la primer imagen con K = 3 #####
k = 3
A = kmeans(datos1[,1:3],k)
datos1$etiqueta = A$cluster
# Agregar el color promedio #
datos1$ecolor = promcolor(datos1)
### Proyectar la imagen ###
plot3d(x = datos1$Y,
       y = 1,
       z = datos1$X,
       col = datos1$ecolor)
rgl.snapshot("RESULTADOS/R1.2.png", fmt = "png")

##### Kmedias para la primer imagen con K = 5 #####
k = 5
A = kmeans(datos1[,1:3],k)
datos1$etiqueta = A$cluster
# Agregar el color promedio #
datos1$ecolor = promcolor(datos1)
### Proyectar la imagen ###
plot3d(x = datos1$Y,
       y = 1,
       z = datos1$X,
       col = datos1$ecolor)
rgl.snapshot("RESULTADOS/R1.3.png", fmt = "png")

##### Kmedias para la primer imagen con K = 10 #####
k = 10

```

```

A = kmeans(datos1[,1:3],k)
datos1$etiqueta = A$cluster
# Agregar el color promedio #
datos1$ecolor = promcolor(datos1)
### Proyectar la imagen ###
plot3d(x = datos1$Y,
       y = 1,
       z = datos1$X,
       col = datos1$ecolor)
rgl.snapshot("RESULTADOS/R1.4.png", fmt = "png")

##### Sección de K medias para imagen 2#####
ruta2 = "IMAGENES/IM4.jpg"
imagen2 = readJPEG(ruta2) # Descomponer la imagen en sus canales RGB
datos2 = pixeles(imagen2)
dim(imagen2) # La imagen 1 tiene 1024 X 768 = 786,432

##### Definir el criterio de la intra varianza
##### Definir el criterio de la intra varianza
set.seed(2024)
pdf(file="RESULTADOS/R2.pdf")
INTRA(datos2[,1:3])
dev.off()

##### Kmedias para la segunda imagen con K = 3 #####
k = 3
A = kmeans(datos2[,1:3],k)
datos2$etiqueta = A$cluster
# Agregar el color promedio #
datos2$ecolor = promcolor(datos2)
### Proyectar la imagen ###
plot3d(x = datos2$Y,
       y = 1,
       z = datos2$X,
       col = datos2$ecolor)
rgl.snapshot("RESULTADOS/R2.2.png", fmt = "png")

##### Kmedias para la segunda imagen con K = 5 #####
k = 5
A = kmeans(datos2[,1:3],k)
datos2$etiqueta = A$cluster
# Agregar el color promedio #
datos2$ecolor = promcolor(datos2)
### Proyectar la imagen ###
plot3d(x = datos2$Y,
       y = 1,
       z = datos2$X,
       col = datos2$ecolor)
rgl.snapshot("RESULTADOS/R2.3.png", fmt = "png")

##### Kmedias para la segunda imagen con K = 10 #####
k = 10

```

```

A = kmeans(datos2[,1:3],k)
datos2$etiqueta = A$cluster
# Agregar el color promedio #
datos2$ecolor = promcolor(datos2)
### Proyectar la imagen ###
plot3d(x = datos2$Y,
       y = 1,
       z = datos2$X,
       col = datos2$ecolor)
rgl.snapshot("RESULTADOS/R2.4.png", fmt = "png")

##### Sección de K medias para imagen 3#####
ruta3 = "IMAGENES/IM1.jpg"
imagen3 = readJPEG(ruta3) # Descomponer la imagen en sus canales RGB
datos3 = pixeles(imagen3)
dim(imagen3) # La imagen 1 tiene 1024 X 1024 = 1,048,576

##### Definir el criterio de la intra varianza
set.seed(2024)
pdf(file="RESULTADOS/R3.pdf")
INTRA(datos3[,1:3])
dev.off()

##### Kmedias para la tercera imagen con K = 3 #####
k = 3
A = kmeans(datos3[,1:3],k)
datos3$etiqueta = A$cluster
# Agregar el color promedio #
datos3$ecolor = promcolor(datos3)
### Proyectar la imagen ###
plot3d(x = datos3$Y,
       y = 1,
       z = datos3$X,
       col = datos3$ecolor)
rgl.snapshot("RESULTADOS/R3.2.png", fmt = "png")

##### Kmedias para la tercera imagen con K = 5 #####
k = 5
A = kmeans(datos3[,1:3],k)
datos3$etiqueta = A$cluster
# Agregar el color promedio #
datos3$ecolor = promcolor(datos3)
### Proyectar la imagen ###
plot3d(x = datos3$Y,
       y = 1,
       z = datos3$X,
       col = datos3$ecolor)
rgl.snapshot("RESULTADOS/R3.3.png", fmt = "png")

##### Kmedias para la segunda imagen con K = 10 #####
k = 10
A = kmeans(datos3[,1:3],k)

```

```

datos3$etiqueta = A$cluster
# Agregar el color promedio #
datos3$ecolor = promcolor(datos3)
### Proyectar la imagen ####
plot3d(x = datos3$Y,
       y = 1,
       z = datos3$X,
       col = datos3$ecolor)
rgl.snapshot("RESULTADOS/R3.4.png", fmt = "png")

##### Mezcla de Gaussianas para la primer imagen #####
### Función para llamar el código fuente de Julia ####
julia_source("GMM en julia.jl")

### Cargar la imagen 1 ####
imagen1 = readJPEG(ruta1) # Descomponer la imagen en sus canales RGB
datos1 = pixeles(imagen1)

##### GMM con K = 3 para la primer imagen #####
k = 3
set.seed(2024)
re = proc.time()
prueba1 = julia_call("gausiano", k, datos1[, 1:3])
proc.time() - re
### Proyectar la imagen ####
datos1$etiqueta = prueba1$Etiqueta
datos1$ecolor = promcolor(datos1)
plot3d(x = datos1$Y,
       y = 1,
       z = datos1$X,
       col = datos1$ecolor)
rgl.snapshot("RESULTADOS/R1.5.png", fmt = "png")

##### GMM con K = 5 para la primer imagen #####
k = 5
set.seed(2024)
re = proc.time()
prueba1 = julia_call("gausiano", k, datos1[, 1:3])
proc.time() - re
### Proyectar la imagen ####
datos1$etiqueta = prueba1$Etiqueta
datos1$ecolor = promcolor(datos1)
plot3d(x = datos1$Y,
       y = 1,
       z = datos1$X,
       col = datos1$ecolor)
rgl.snapshot("RESULTADOS/R1.6.png", fmt = "png")

##### GMM con K = 10 para la primer imagen #####
k = 10
set.seed(2024)
re = proc.time()
prueba1 = julia_call("gausiano", k, datos1[, 1:3])

```

```

proc.time() -re
### Proyectar la imagen ####
datos1$etiqueta = prueba1$Etiqueta
datos1$ecolor = promcolor(datos1)
plot3d(x = datos1$Y,
       y = 1,
       z = datos1$X,
       col = datos1$ecolor)
rgl.snapshot("RESULTADOS/R1.7.png", fmt = "png")

##### Mezcla de Gaussianas para la segunda imagen #####
imagen2 = readJPEG(ruta2) # Descomponer la imagen en sus canales RGB
datos2 = pixeles(imagen2)

##### GMM con K = 3 para la segunda imagen #####
k = 3
set.seed(2024)
re = proc.time()
prueba2 = julia_call("gausiano", k, datos2[, 1:3])
proc.time() -re
### Proyectar la imagen ####
datos2$etiqueta = prueba2$Etiqueta
datos2$ecolor = promcolor(datos2)
plot3d(x = datos2$Y,
       y = 1,
       z = datos2$X,
       col = datos2$ecolor)
rgl.snapshot("RESULTADOS/R2.5.png", fmt = "png")

##### GMM con K = 5 para la segunda imagen #####
k = 5
set.seed(2024)
re = proc.time()
prueba2 = julia_call("gausiano", k, datos2[, 1:3])
proc.time() -re
### Proyectar la imagen ####
datos2$etiqueta = prueba2$Etiqueta
datos2$ecolor = promcolor(datos2)
plot3d(x = datos2$Y,
       y = 1,
       z = datos2$X,
       col = datos2$ecolor)
rgl.snapshot("RESULTADOS/R2.6.png", fmt = "png")

##### GMM con K = 10 para la segunda imagen #####
k = 10
set.seed(2024)
re = proc.time()
prueba2 = julia_call("gausiano", k, datos2[, 1:3])
proc.time() -re
### Proyectar la imagen ####
datos2$etiqueta = prueba2$Etiqueta
datos2$ecolor = promcolor(datos2)

```

```

plot3d(x = datos2$Y,
       y = 1,
       z = datos2$X,
       col = datos2$ecolor)
rgl.snapshot("RESULTADOS/R2.7.png", fmt = "png")

##### Mezcla de Gaussianas para la tercera imagen #####
imagen3 = readJPEG(ruta3) # Descomponer la imagen en sus canales RGB
datos3 = pixeles(imagen3)

##### GMM con K = 3 para la tercera imagen #####
k = 3
set.seed(2024)
re = proc.time()
prueba3 = julia_call("gausiano",k,datos3[,1:3])
proc.time()-re
### Proyectar la imagen ###
datos3$etiqueta = prueba3$Etiqueta
datos3$ecolor = promcolor(datos3)
plot3d(x = datos3$Y,
       y = 1,
       z = datos3$X,
       col = datos3$ecolor)
rgl.snapshot("RESULTADOS/R3.5.png", fmt = "png")

##### GMM con K = 5 para la tercera imagen #####
k = 5
set.seed(2024)
re = proc.time()
prueba3 = julia_call("gausiano",k,datos3[,1:3])
proc.time()-re
### Proyectar la imagen ###
datos3$etiqueta = prueba3$Etiqueta
datos3$ecolor = promcolor(datos3)
plot3d(x = datos3$Y,
       y = 1,
       z = datos3$X,
       col = datos3$ecolor)
rgl.snapshot("RESULTADOS/R3.6.png", fmt = "png")

##### GMM con K = 10 para la tercera imagen #####
k = 10
set.seed(2024)
re = proc.time()
prueba3 = julia_call("gausiano",k,datos3[,1:3])
proc.time()-re
### Proyectar la imagen ###
datos3$etiqueta = prueba3$Etiqueta
datos3$ecolor = promcolor(datos3)
plot3d(x = datos3$Y,
       y = 1,
       z = datos3$X,

```

```
    col = datos3$ecolor)
rgl.snapshot("RESULTADOS/R3.7.png", fmt = "png")
```