Probabilidad
Tarea 8
Ajuste de Modelo GMM Gaussian Mixture Model

Dado un conjunto de I=30 puntos generados muestreando aleatoriamente 2 gaussianas unidimensionales $\mu1=4$ $\sigma1=3$, $\mu2=12$ $\sigma2=3.5$. Muestrear 15 de la Gaussiana 1 y 15 puntos de la Gaussiana 2. Iterar el algoritmo para encontrar los parámetros óptimos para el conjunto de entrenamiento. Se proporciona k=2.

Para realizar este trabajo, se hace uso de las siguientes formulas:

$$Pr(\mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \lambda_k \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k]$$

Calculo de la probabilidad para la multigaussiana

$$\begin{aligned} &= \frac{\lambda_k \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k]}{\sum_{j=1}^{K} \lambda_j \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j]} \\ &= r_{ik} \end{aligned}$$

$$\lambda_k^{[t+1]} = \frac{\sum_{i=1}^{I} r_{ik}}{\sum_{j=1}^{K} \sum_{i=1}^{I} r_{ij}}$$

$$\boldsymbol{\mu}_k^{[t+1]} = \frac{\sum_{i=1}^{I} r_{ik} \mathbf{x}_i}{\sum_{i=1}^{I} r_{ik}}$$

$$\boldsymbol{\Sigma}_k^{[t+1]} = \frac{\sum_{i=1}^{I} r_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k^{[t+1]})(\mathbf{x}_i - \boldsymbol{\mu}_k^{[t+1]})^T}{\sum_{i=1}^{I} r_{ik}}$$

Calculo de parámetros para la siguiente iteración

En la primera iteración los valores de μ, σ y λ para las dos k empiezan de manera aleatoria y con cada iteración se van entrenando de acuerdo a los 30 puntos que se generaron de las dos gaussianas dadas.

El código se programa en python:

```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import math

mu1=4
sigma1=3
mu2=12
sigma2=3.5
k=2
puntos=15
rango=2*puntos
nums=np.array([])
numsy=np.array([])
lambdac1=0.50+(random.randint(0,9)/100)
muc1=random.randint(0,12)
sigmac1=random.randint(0,10)
lambdac2=1-lambdac1
muc2=random.randint(0,12)
sigmac2=random.randint(0,10)
gmuc1=np.array([muc1])
gmuc2=np.array([muc2])
glambdac1=np.array([lambdac1])
glambdac2=np.array([lambdac2])
gsigmac1=np.array([sigmac1])
gsigmac2=np.array([sigmac2])
xiteracion=np.array([0])

#FUNCION PARA CALCULO DE LA NORMAL
def normal (x,mu,sigma):
	return np.exp(-1*((x-mu)**2)/(2*(sigma**2)))/(math.sqrt(2*np.pi) * sigma)

#CREACION DE PUNTOS ALEATORIOS BASADO EN DATOS DE GAUSSIANA
for i in range(puntos):
	x=random.gauss(mu1, sigma1)
```

```
        nums=np.append(nums,x)
        numsy=np.append(numsy,0)
        x=random.gauss(mu2, sigma2)
        nums=np.append(nums,x)
        numsy=np.append(numsy,0)
        numsord=sorted(nums)

#CREACION DE FIGURAS CON MU Y SIGMA DADOS
plt.subplot(2,2,1)
plt.title("Gaussianas de valores dados")
x1=np.linspace(mu1-6*sigma1,mu1+6*sigma1,100)
y1=normal(x1,mu1,sigma1)
x2=np.linspace(mu2-6*sigma2,mu2+6*sigma2,100)
y2=normal(x2,mu2,sigma2)
plt.plot(x1,y1,'b')
plt.plot(x2,y2,'g')
plt.scatter(nums,numsy,color='k')
plt.xlim(0,20)
plt.ylim(0,0.5)

#CREACION DE FIGURA CON MU Y SIGMA ALEATORIOS
plt.subplot(2,2,2)
plt.title("Gaussianas de valores aleatorios")
x1=np.linspace(muc1-6*sigmac1,muc1+6*sigmac1,100)
y1=normal(x1,muc1,sigmac1)
x2=np.linspace(muc2-6*sigmac2,muc2+6*sigmac2,100)
y2=normal(x2,muc2,sigmac2)
plt.plot(x1,y1,'b')
plt.plot(x2,y2,'g')
plt.scatter(nums,numsy,color='k')
plt.xlim(0,20)
plt.ylim(0,0.5)

#INICIO DE ITERACIONES
for j in range(1,5):

        vrik1=np.zeros((rango))
        vrik2=np.zeros((rango))
        sumrik1=0
        sumrik2=0
        sumriks=0
        sumrik1x=0
        sumrik2x=0
```

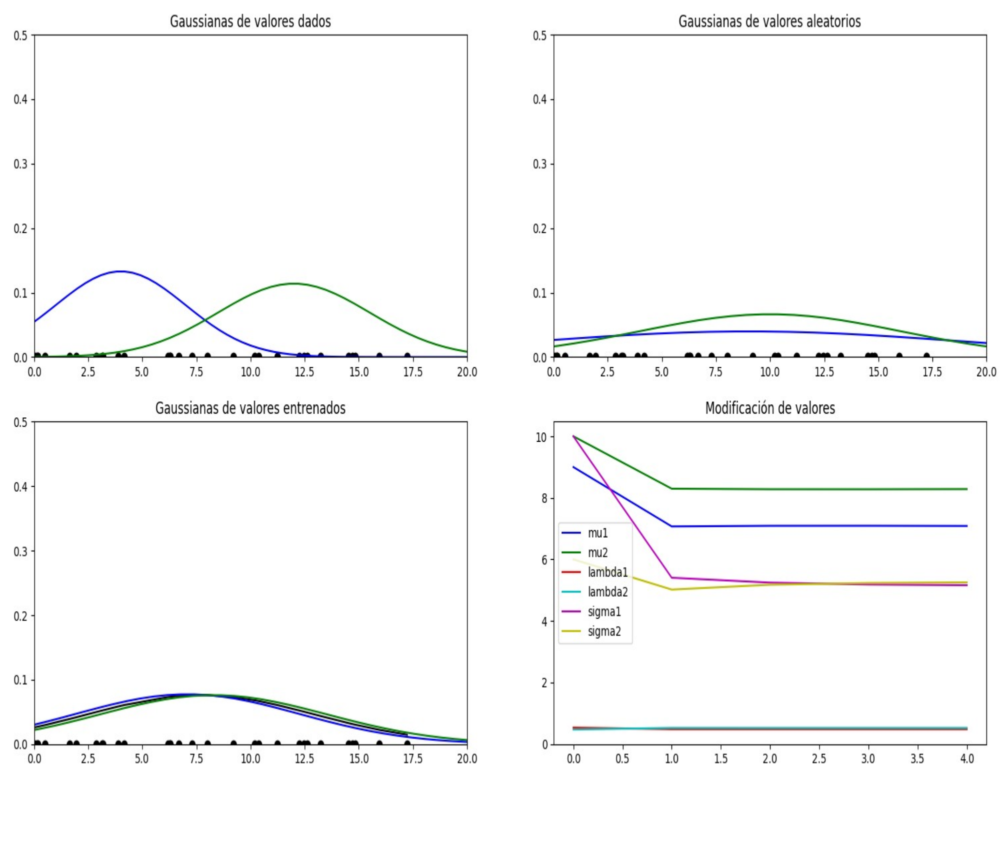
```
    sumrik1xmu=0
    sumrik2xmu=0

#CALCULO DE rik DE CADA PUNTO EN CADA K
    for i in range(rango):
        n1=lambdac1*normal(nums[i],muc1,sigmac1)
        n2=lambdac2*normal(nums[i],muc2,sigmac2)
        rik1=n1/(n1+n2)
        rik2=n2/(n1+n2)
        vrik1[i]=rik1
        vrik2[i]=rik2
        sumrik1=sumrik1+rik1
        sumrik2=sumrik2+rik2
        sumriks=sumriks+rik1+rik2
        sumrik1x=sumrik1x+(rik1*nums[i])
        sumrik2x=sumrik2x+(rik2*nums[i])

    #CALCULO DE NUEVOS LAMBDAS
    lambdac1old=lambdac1
    lambdac2old=lambdac2
    lambdac1=sumrik1/sumriks
    lambdac2=sumrik2/sumriks

    #CALCULO DE NUEVAS MUS
    muc1old=muc1
    muc2old=muc2
    muc1=sumrik1x/sumrik1
    muc2=sumrik2x/sumrik2

    #CALCULO DE NUEVAS SIGMAS
    for k in range(rango):
        dif1=nums[k]-muc1
        dif2=nums[k]-muc2
        rik1xmu=dif1*dif1*vrik1[k]
        rik2xmu=dif2*dif2*vrik2[k]
        sumrik1xmu=rik1xmu+sumrik1xmu
        sumrik2xmu=rik2xmu+sumrik2xmu

    sigmac1old=sigmac1
    sigmac2old=sigmac2
    sigmac1=(sumrik1xmu/sumrik1)**0.5
    sigmac2=(sumrik2xmu/sumrik2)**0.5
```

```
        gmuc1=np.append(gmuc1,muc1)
        gmuc2=np.append(gmuc2,muc2)
        glambdac1=np.append(glambdac1,lambdac1)
        glambdac2=np.append(glambdac2,lambdac2)
        gsigmac1=np.append(gsigmac1,sigmac1)
        gsigmac2=np.append(gsigmac2,sigmac2)
        xiteracion=np.append(xiteracion,j)

print(xiteracion)
print("mu1:")
print(gmuc1)
print("mu2:")
print(gmuc2)
print("lambda1:")
print(glambdac1)
print("lambda2:")
print(glambdac2)
print("sigma1:")
print(gsigmac1)
print("sigma2:")
print(gsigmac2)

pr=np.zeros((rango))
for i in range(rango):
norm1=lambdac1*normal(numsord[i],muc1,sigmac1)
norm2=lambdac2*normal(numsord[i],muc2,sigmac2)
pr[i]=norm1+norm2

#CREACION DE FIGURA CON MU Y SIGMA ENTRENADOS
plt.subplot(2,2,3)
plt.title("Gaussianas de valores entrenados")
x1=np.linspace(muc1-6*sigmac1,muc1+6*sigmac1,100)
y1=normal(x1,muc1,sigmac1)
x2=np.linspace(muc2-6*sigmac2,muc2+6*sigmac2,100)
y2=normal(x2,muc2,sigmac2)
plt.plot(numsord,pr,'k')
plt.plot(x1,y1,'b')
plt.plot(x2,y2,'g')
plt.scatter(nums,numsy,color='k')
plt.xlim(0,20)
plt.ylim(0,0.5)

#CREACION DE FIGURA DE CAMBIO DE DATOS
```

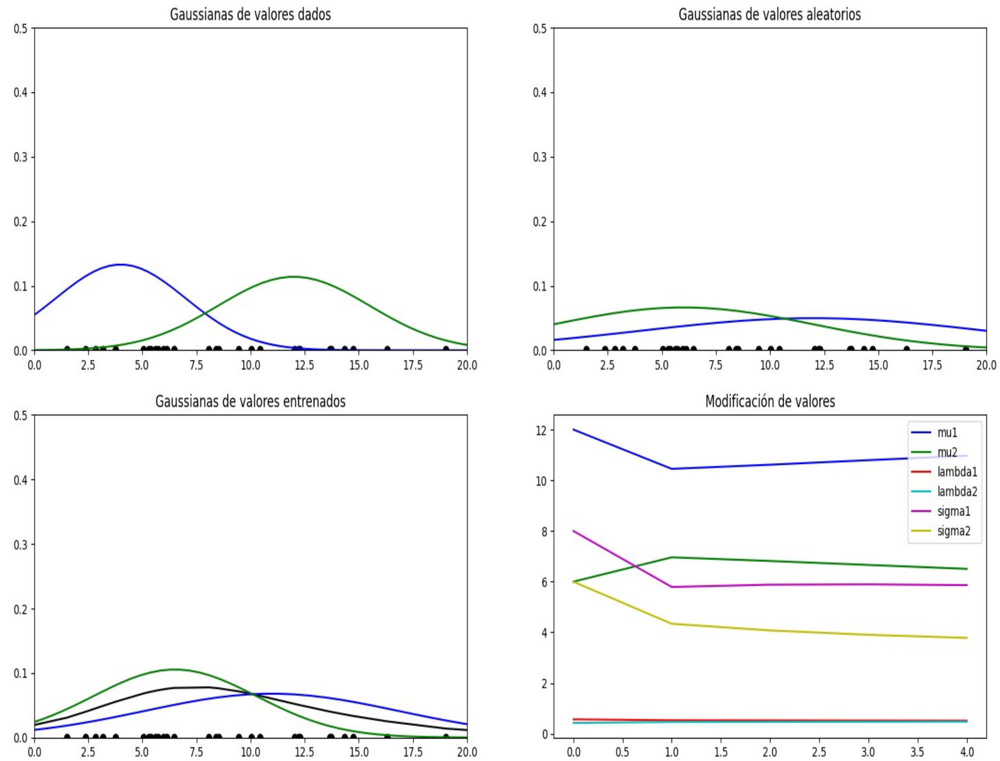
```
plt.subplot(2,2,4)
plt.title("Modificación de valores")
plt.plot(xiteracion,gmuc1,'b',label='mu1')
plt.plot(xiteracion,gmuc2,'g',label='mu2')
plt.plot(xiteracion,glambdac1,'r',label='lambda1')
plt.plot(xiteracion,glambdac2,'c',label='lambda2')
plt.plot(xiteracion,gsigmac1,'m',label='sigma1')
plt.plot(xiteracion,gsigmac2,'y',label='sigma2')
plt.legend()

plt.show()
```

Resultados:

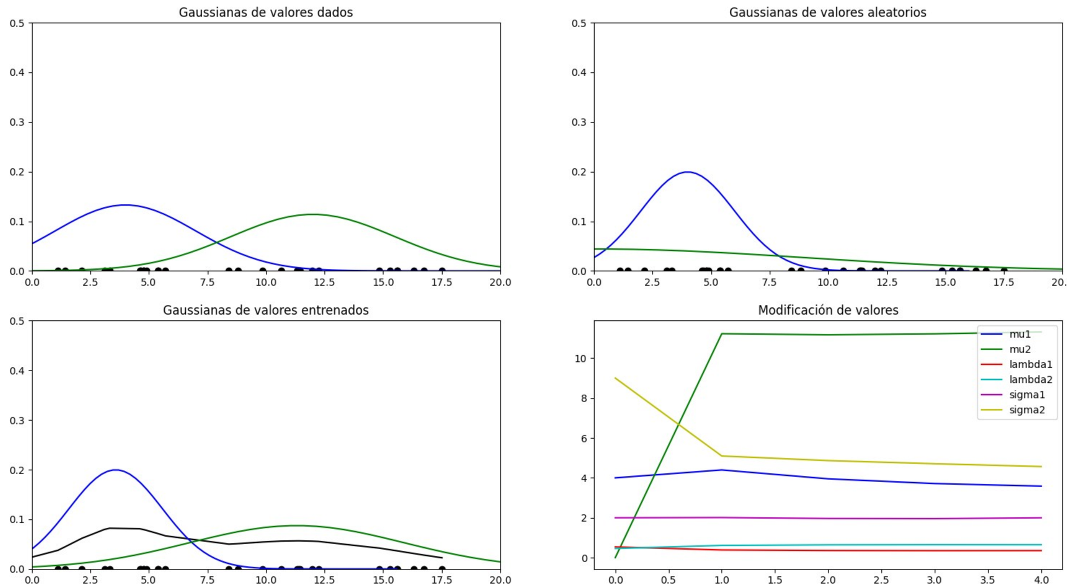
Al correr el código, debido a que los valores de las μ y σ son aleatorios, cuando las sigmas son grandes y sus valores de mu son muy cercanos es posible que ambos μ y σ terminen teniendo valores similares, y en ver de verse como dos gaussianas unidas termina pareciendo una sola:



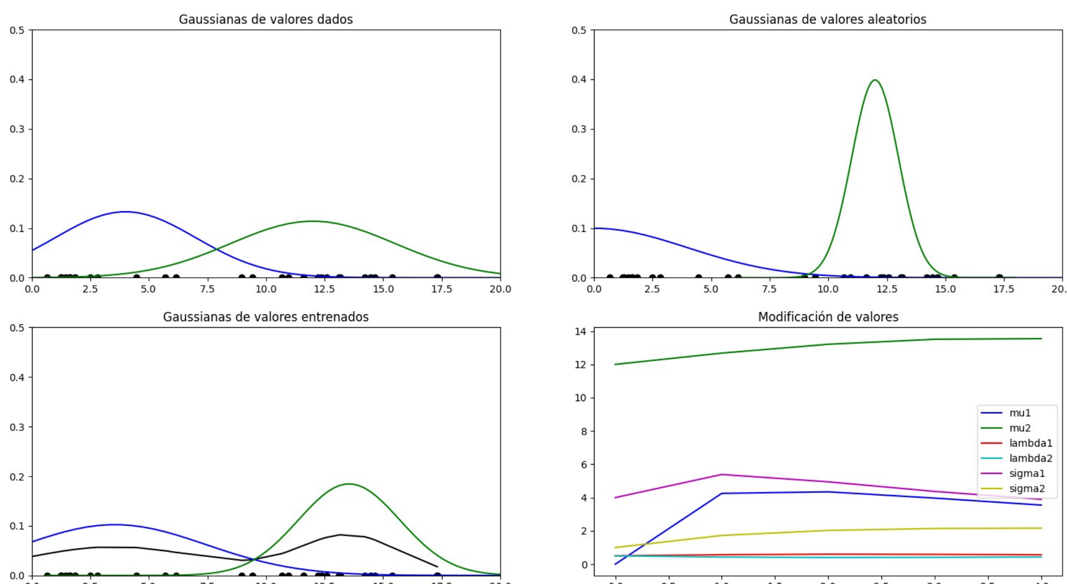
Otro caso es donde los valores si tienen cierta separación y se aprecia de mejor manera como las gaussianas se dividen y como la multigaussiana toma forma similar a las dos gaussianas juntas:



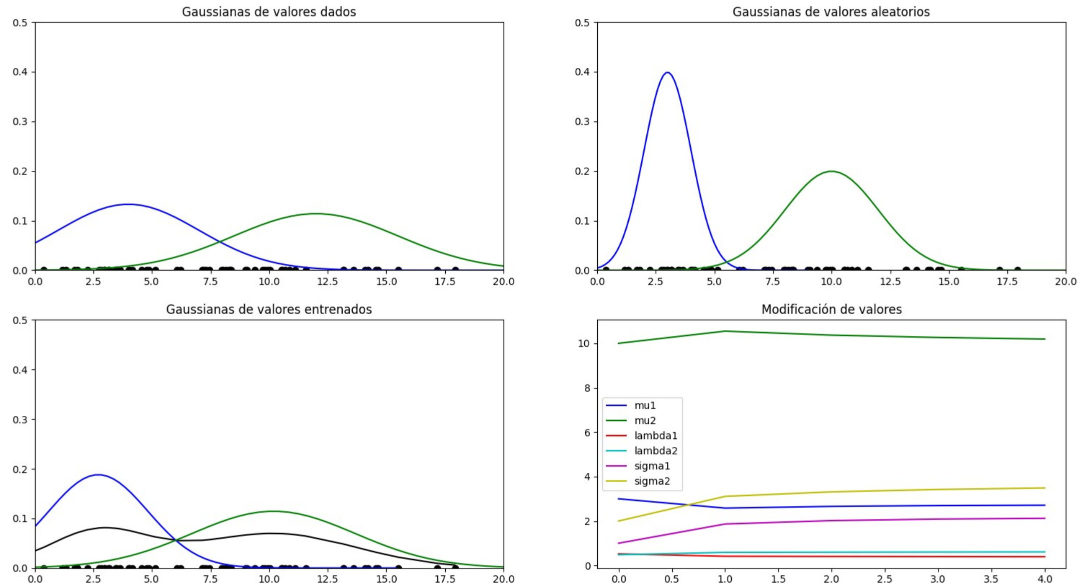
El siguiente caso se puede ver como a pesar de que μ2 empieza siendo mas chico que μ1, uno pensaria que la guassiana 1, de color azul, seria la que se moveria al lado derecho para aprender los valores de ese lado, sin embargo aquí es donde juega un factor importante los σ, ya que σ1=2 y σ2=11, por lo que al ser σ2 el valor mas grande es el que termina haciendo que la gaussiana verde se mueva a la derecha y sea la que mas probabilidad tenga para esos datos.



El siguiente caso es donde las μ empiezan separadas y tienen σ con valores menores a 5, lo cual hace que el entrenamiento, tome la forma ideal, donde es fácilmente distinguible cada gaussiana y se puede apreciar como la multigaussiana toma forma similar a las dos gaussianas.



Finalmente el ultimo caso es cuando aumentamos el numero de puntos aleatorios generados al inicio de 15 por gaussiana a 30 para cada una, dando 60 puntos en total, para obtener las siguientes graficas:



Conclusión:

En conclusión se puede ver que el modelo si es capaz de aprender de manera adecuada los puntos generados y obtener una multigaussiana para estos datos, sin embargo al ser aleatorios los μ, σ y λ es posible que aprendan de diferente manera en cada corrida, lo cual se puede mejorar limitando la aleatorización de estos valores de acuerdo al tipo de problema que se quiere resolver.