4

ALGORITMOS DE AGRUPAMIENTO (CLUSTERING)

El agrupamiento de objetos es tan antiguo como la necesidad humana de describir las características destacadas de los objetos e identificarlos en alguna clase. Además, abarca diversas disciplinas: desde las matemáticas y la estadística hasta la genética y la biología, cada una de ellas hace uso de distintos términos para describir las topologías usando este análisis. Desde los síndromes médicos y genotipos genéticos, hasta las taxonomías biológicas o los grupos de tecnología; el problema es el mismo: encontrar categorías de entidades y asignar a los individuos a los grupos apropiados en ellas.

Los algoritmos de agrupamiento o de *clustering* por su nombre en inglés, son una herramienta extensivamente utilizada en el aprendizaje no supervisado para organizar, caracterizar, clasificar y modelar información y datos. Estos algoritmos dividen un conjunto de datos en distintos grupos de manera en que las diferencias entre estos grupos son menores que la diferencia que hay con el resto de ellos. En este capítulo se abordarán distintos algoritmos de agrupamiento, como los algoritmos secuenciales, basados en centroide, basados en densidad y los algoritmos de agrupamiento jerárquico. Al avanzar en el capítulo cada una de estas técnicas será analizada de manera individual partiendo desde un enfoque teórico para posteriormente mostrar su implementación en lenguaje Python para facilitar la comprensión de cada uno de ellos.

4.1 INTRODUCCIÓN

Los algoritmos de agrupamiento y clasificación son una tarea fundamental en la inteligencia computacional. Mientras que los algoritmos de clasificación son

mayoritariamente utilizados como métodos de aprendizaje supervisado en los que se entrena partiendo de un conjunto de datos etiquetado en el cual se conocen las salidas correspondientes al vector de información de entrada. En el aprendizaje no supervisado no se cuenta con esta información, es en estos casos cuando se cuenta con los algoritmos de agrupamiento, cuyo objetivo es descubrir un nuevo conjunto de grupos que sean similares entre si dependiendo de sus características.

Los algoritmos de agrupamiento dividen el conjunto de datos en subconjuntos de tal manera que datos con instancias similares son agrupados juntos, mientras que aquellos cuyas instancias sean diferentes pertenecerán a grupos distintos. Tales instancias, deberán por lo tanto ser organizadas de una manera que el conjunto de datos muestreado sea caracterizado eficientemente.

4.2 DEFINICIÓN DE CLÚSTER

Formalmente, la estructura de los algoritmos de agrupamiento está representada partiendo de un conjunto de datos $X=\{x_1,x_2,...,x_n\}$ y dividiéndolo en un conjunto de subconjuntos (clústeres) $C=\{C_1,C_2,...,C_k\}$ de S tal que:

$$\bigcup_{i=1}^{k} C_i = X, \qquad i = 1,2,\ldots,k \tag{4.1}$$

$$C_i \neq \varnothing, \quad i = 1,2,\ldots,k \tag{4.2}$$

$$C_i \cap C_j = \varnothing \ , i \neq j, \ i, j = 1,2,\ldots,k \tag{4.3}$$

en consecuencia, cualquier instancia en S puede pertenecer exactamente a solo un subconjunto C_i.

Debido a que los algoritmos de agrupamiento dividen el conjunto de datos de acuerdo con la similitud de sus características, es necesario tener una métrica que determine qué tan similares o diferentes son dos objetos.

4.3 MÉTRICAS DE PROXIMIDAD

Debido a que los algoritmos de agrupamiento dividen el conjunto de datos de acuerdo con la similitud de sus características, es necesario tener una métrica que determine qué tan similares o diferentes son dos objetos. Existen distintas métricas de proximidad para estimar esta relación, medidas de similitud y medidas de disimilitud.

Para comenzar, es posible definir una métrica de disimilitud d en X como una función:

$$d : X \times X \to \mathbb{R} \tag{4.4}$$

Siendo $\mathbb{R}$ el conjunto de los números reales de tal manera que:

$$\begin{gathered} \exists d_0 \in \mathbb{R} : -\infty < d_0 \leq d\left(x, y\right) < +\infty, \ \forall x, y \in X \\ d\left(x, x\right) = d_0 \ , \forall x \in X \end{gathered} \tag{4.5}$$

Y:

$$d\left(x, y\right) = d\left(y, x\right) \forall \, x, y \in X \tag{4.6}$$

Adicionalmente:

$$d\left(x, y\right) = d_0 \, sii \, x = y \tag{4.7}$$

Además

$$d\left(x, z\right) \leq d\left(x, y\right) + d\left(y, z\right), \forall x, y, z \in X \tag{4.8}$$

De esta manera d puede ser llamada una métrica *DM* de desigualdad. Finalmente, la igualdad presentada en la ecuación 4.7 indica que el menor valor de desigualdad d_0 posible entre dos vectores del conjunto X se alcanza solo si estos vectores son idénticos. Es frecuente mencionar las métricas de disimilitud como una distancia aunque este término no sea utilizado en el sentido matemático más estricto.

Por otro lado, una métrica de similitud (*SM*) s en X esta definida como una función:

$$s : X \times X \to \mathbb{R} \tag{4.9}$$

De tal manera que:

$$\begin{gathered} \exists s_0 \in \mathbb{R} : -\infty < s_0 \leq s\left(x, y\right) < +\infty, \ \forall x, y \in X \\ s\left(x, x\right) = s_0 \ , \forall x \in X \end{gathered} \tag{4.10}$$

Y:

$$s\left(x, y\right) = s\left(y, x\right) \forall \, x, y \in X \tag{4.11}$$

Adicionalmente:

$$s\left(x, y\right) = s_0 \, sii \, x = y \tag{4.12}$$

Además

$$s(x,z) \leq s(x,y) + s(y,z), \forall x,y,z \in X \tag{4.13}$$

4.3.1 Métricas de disimilitud

Entre las métricas de disimilitud una de las más ampliamente usadas es la conocida como la distancia euclidiana que está definida como:

$$d_2(x,y) = \sqrt{\sum_{i=1}^{l} (x_i - y_i)^2} \tag{4.14}$$

Donde $x,y \in X$ y x_i, y_i son las i-esimas coordenadas de x e y respectivamente. En esta métrica de disimilitud d_0 tiene un valor igual a cero, siendo esta la mínima distancia disponible entre dos vectores de X. Cumpliendo también que la distancia entre un vector y si mismo es igual a d_0. Además, es fácil observar que $d(x,y)=d(y,x)$ cumpliendo así todos los requisitos para ser considerada una métrica de disimilitud.

Si bien, la distancia euclidiana es una de las métricas de disimilitud más conocidas, en el resto de la sección se abordarán otras métricas que también pueden ser del interés del lector.

Entre las métricas de disimilitud más comunes usadas en la práctica podemos tomar en cuenta las siguientes:

La métrica pesada l_p dada por la siguiente ecuación:

$$d_p(x,y) = \left(\sum_{i=1}^{l} w_i |x_i - y_i|^p \right)^{1/p} \tag{4.15}$$

Siendo de x_i, y_i las i-ésimas coordenadas de los vectores x,y y w_i un valor mayor a cero el i-ésimo coeficiente de ponderación de dichos vectores y coordenadas. De esta norma deriva la distancia euclidiana cuando se ajusta el valor $p=2$.

De este caso también deriva la distancia *Manhattan dada por:*

$$d_p(x,y) = \sum_{i=1}^{l} w_i |x_i - y_i| \tag{4.16}$$

4.3.2 Métricas de similitud

Del otro lado, las métricas de similitud más conocidas son:

Métrica del producto interno: esta métrica está dada por:

$$s_{int}\left(x,y\right)=x^{T}y=\sum_{i=1}^{l}x_{i}y_{i} \tag{4.17}$$

esta métrica suele ser utilizada cuando los vectores *x,y* están normalizados de manera que tengan el mismo tamaño.

Otra métrica importante relacionada con la métrica del producto interno es la *métrica del coseno* dada por:

$$s_{cos}\left(x,y\right)=\frac{x^{T}y}{\|x\|\,\|y\|} \tag{4.18}$$

Siendo:

$$\|x\|=\sqrt{\sum_{i=1}^{l}x_{i}^{2}} \tag{4.19}$$

Y

$$\|y\|=\sqrt{\sum_{i=1}^{l}y_{i}^{2}} \tag{4.20}$$

4.4 PASOS BÁSICOS PARA HACER AGRUPAMIENTO

Una vez repasados los conceptos básicos y definiciones de los algoritmos de agrupamiento y medidas de proximidad es sencillo desempeñar una tarea de agrupamiento dividiéndola en los siguientes pasos:

1. *Selección de características:* estas deben ser seleccionadas de manera en que contengan la mayor cantidad de información posible respecto a la tarea de interés, al mismo tiempo, es necesario evitar tener redundancias en los vectores de características.

2. *Selección de métrica de proximidad:* esta métrica, como ya fue mencionado en la sección anterior, cuantifica qué tan similares o diferentes son los vectores de características. Es importante tomar en

cuenta que las características seleccionadas afecten de igual manera a la métrica de proximidad.

3. *Selección del criterio de agrupamiento:* este criterio depende de la interpretación que el experto de en términos de sensibilidad, basándose en los tipos de grupos que se espere encontrar en el conjunto de datos.

4. *Selección del algoritmo de agrupamiento:* una vez definidos el criterio de agrupamiento y la métrica de proximidad, el siguiente paso es escoger el algoritmo de agrupamiento adecuado con respecto al conjunto de datos con el que se va a trabajar.

5. *Validación de los resultados:* una vez que el algoritmo de agrupamiento ha terminado su tarea, es necesario verificar la fiabilidad de los resultados arrojados por este último. Esto último se lleva a cabo aplicando las pruebas adecuadas.

6. *Interpretación de los resultados:* finalmente los resultados arrojados por el algoritmo deben ser interpretados por un experto en el campo de aplicación en conjunto con otra evidencia experimental para llegar a una conclusión adecuada.

4.5 ALGORITMOS DE AGRUPAMIENTO CLÁSICOS

En secciones anteriores, se discutieron las definiciones básicas de grupos (clusters), métricas de proximidad, y los pasos básicos a desempeñar para llevar a cabo una tarea de agrupamiento. En esta sección se discutirán de manera detallada algunos de los algoritmos clásicos de agrupamiento y la manera adecuada de obtener el número de clústeres que suelen pedir varios de los algoritmos en cuestión.

4.5.1 Cálculo del posible número de grupos a obtener

En la actualidad existen varios algoritmos que piden al usuario la cantidad máxima de grupos para dividir el conjunto de datos. En general, si se tienen el tiempo y los recursos necesarios, la mejor manera de asignar los vectores de características de x_i ,i=1,...,N de un conjunto de datos X a grupos distintos, sería encontrar todas las posibles particiones de grupos posibles y seleccionar la más adecuada de acuerdo con el criterio de agrupamiento seleccionado. Sin embargo, es virtualmente imposible llevar acabo esta tarea incluso para un valor moderado de vectores de características N debido a lo extenuante de esta tarea. Por lo que en realidad podemos marcar $S(N,m)$ como el número de agrupamientos posibles de N vectores de características

divididos en *m* grupos. Recordando que por definición no pueden existir grupos vacíos, se pueden fijar las siguientes condiciones:

$$S(N,1)=1 \tag{4.21}$$

$$S(N,N)=1 \tag{4.22}$$

$$S(N,m)=0 \ ,sii\, m>N \tag{4.23}$$

Definiendo $L^{K}_{N\text{-}1}$ como la lista que contiene todas las posibles agrupaciones de *N*-1 vectores en *k* grupos para *k* = *m*,*m*-1. El *N*-esimo vector puede ya ser añadido a cualquiera de los grupos de $L^{m}_{N\text{-}1}$, o formar un nuevo grupo con cualquier miembro de $L^{m\text{-}1}_{N\text{-}1}$, por lo tanto, es posible escribir que:

$$S(N,m)=mS(N-1,m)+S(N-1,m-1) \tag{4.24}$$

Guiando a que la solución de la ecuación 4.24 son los también llamados números de Stirling de segundo tipo dados por:

$$S(N,m)=\frac{1}{m!}\sum_{i=o}^{m}(-1)^{m-i}\binom{m}{i}i^{N} \tag{4.25}$$

4.5.2 Algoritmo Básico de Agrupamiento Secuencial

En esta sección se explicará el algoritmo básico de agrupamiento secuencial (BSAS), el cual generaliza lo discutido previamente. Para este algoritmo es necesario considerar que todos los vectores de características son pasados por el algoritmo una única vez. Otra cuestión que considerar es, que para realizar el algoritmo el número de grupos no es conocido a priori ya que el algoritmo los va creando con el paso de las iteraciones.

Tomando la función $d(x,C)$ como una métrica de disimilitud o distancia entre el vector de características y el grupo *C* siendo este último definido a través de una medida de representación *rep*. En este caso se utilizará la media de todos los puntos que conforman el grupo. Es importante recalcar que durante previo al funcionamiento del algoritmo el usuario debe definir ciertos parámetros necesarios para el correcto funcionamiento de éste, estos parámetros son: el número máximo de grupos permitidos *q* y el umbral máximo de disimilitud permitido *th*.

La idea básica del algoritmo plantea lo siguiente:

- Cada vector de características es presentado una única ocasión al algoritmo.

- Cuando un vector es analizado por el algoritmo, dependiendo del valor de la medida de disimilitud, este puede ser asignado a un grupo previamente existente o bien, formar un grupo nuevo.
- Este proceso se repite hasta analizar todos los vectores de características.

El flujo del algoritmo se puede seguir en la Figura 4.1

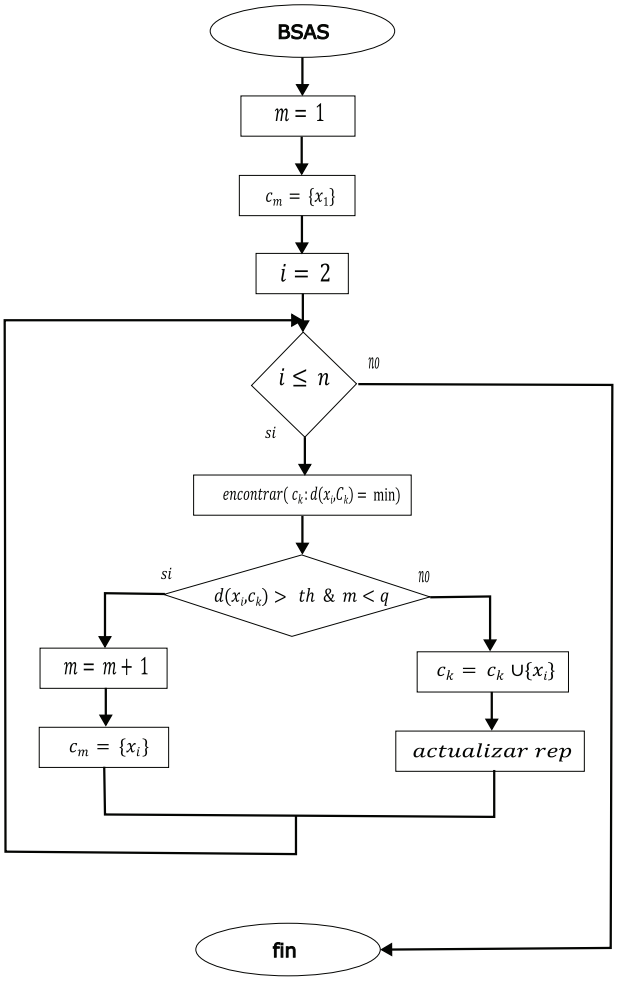

Figura 4.1 Diagrama de flujo algoritmo de agrupamiento BSAS

A continuación, se muestra un ejemplo de implementación del algoritmo BSAS en Python:

Código 4.1 Algoritmo BSAS en Python

```
import numpy as np
import matplotlib.pyplot as plt

def BSAS(X, th, q, orden):
    # Reordenar datos si se proporciona un orden específico
    l, N = X.shape
    if len(order) == N:
        X = X[:,orden]
# Inicialización
    n_clust = 1
    bel = np.zeros(N,dtype=int)
    bel[0] = n_clust

# matriz de centroides (cada columna es un clúster)
    m = X[:,[0]]
    for i in range(1, N):
        m2 = m.shape[1]
# Calcular distancias a los centroides existentes
        distances = np.sqrt(np.sum((m - X[:,i].reshape(-1,1))**2,
axis=0))
        s1 = np.min(distances)
        s2 = np.argmin(distances)
        if s1 > th and n_clust < q:
            n_clust += 1
            bel[i] = n_clust
            m = np.hstack((m, X[:,[i]]))
        else:
            bel[i] = s2 + 1
            count = np.sum(bel[:i+1] == bel[i])
            m[:, s2] = ((count - 1) * m[:,s2] + X[:,i]) / count
    return bel, m
```

Ejemplo 4.1:

Utilizar el algoritmo BSAS para hacer un agrupamiento sobre un conjunto de datos aleatorio, el umbral de distancia máxima para entrar al grupo *th* será de 4 y el número máximo de grupos permitido *q* será de 5.

Solución:

Código 4.2 Ejemplo de Uso de algoritmo BSAS en Python

```
# ----------- Generación de datos aleatorios -----------
np.random.seed(0)
cluster1 = np.random.randn(2,20) + np.array([[5],[5]])
cluster2 = np.random.randn(2, 20) + np.array([[0],[0]])
cluster3 = np.random.randn(2, 20) + np.array([[5],[-5]])
X = np.hstack((cluster1,cluster2,cluster3))
# Mezclar el orden
order = np.random.permutation(X.shape[1])
# ----------- Ejecutar BSAS -----------
theta = 4.0  # Umbral de distancia
q = 5        # Número máximo de clusters
labels,centroids = BSAS(X,theta,q,order)
# ----------- Visualización -----------
plt.figure(figsize=(8, 6))
for k in np.unique(labels):
    idx = np.where(labels == k)
    plt.scatter(X[0,idx],X[1,idx],label=f'Cluster{k}')
plt.scatter(centroids[0,:],centroids[1,:],c='black',marker='x',\
s=100,label='Centroides')
plt.title('Agrupamiento con BSAS')
plt.xlabel('X1')
plt.ylabel('X2')
plt.legend()
plt.grid(True)
plt.show()
```

Es necesario aclarar que para el funcionamiento de este código debe incluirse la definición del algoritmo BSAS mostrada en el código 4.1, además de incluir los módulos de numpy y matplot lib en el intérprete que desee utilizarse. Como resultado el algoritmo 4.2 arrojará lo siguiente:

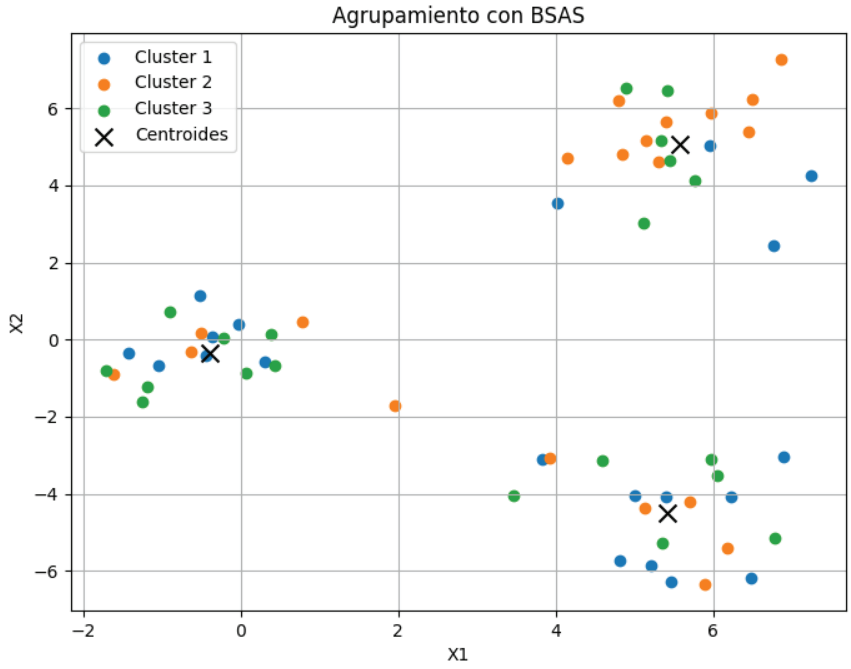

Figura 4.2 Resultado de Agrupación de algoritmo BSAS

Como se puede observar, el algoritmo detecta 3 grupos posibles marcando el centroide de cada uno de ellos con una "×".

4.5.3 Algoritmo K-medias (K-means)

En esta sección será presentado uno de los algoritmos de agrupamiento más ampliamente utilizados en una gran variedad de aplicaciones de inteligencia computacional, el algoritmo K-medias mejor conocido por su nombre en inglés como K-means.

Este algoritmo divide un conjunto de vectores $x=(x_1,x_2,...,x_n)$ en m grupos $G=(g_1,g_2,...,g_m)$. La medida de representación de los grupos en el algoritmo K-medias está definida por el centroide $c_i, i=1,2,...,m$ el cual se calcula minimizando una función de costo en la cual se asume como métrica de disimilitud la distancia euclidiana. Esta función de costo está definida de la siguiente manera:

$$J = \sum_{i=1}^{m} \sum_{k, x_k \in G_i} \|x_k - c_i\|^2 \tag{4.26}$$

Siendo J la función de costo a minimizar atribuida al grupo g_i. De acuerdo con la ecuación 4.26, el valor de J depende exclusivamente de la distribución espacial de los vectores pertenecientes a g_i y de la posición de su centroide c_i.

La separación de los vectores de características en cada grupo está definida por un factor $u_{i,j}$. Este factor, determina de forma binaria la correspondencia de un vector con un grupo g_i de la siguiente manera:

$$u_{i,j} = \begin{cases} 1, & x_i \in g_i \\ 0, & x_i \ni g_i \end{cases} \tag{4.27}$$

Siendo así que agregando este factor a la ecuación 4.26 esta se puede escribir como:

$$J = \sum_{i=1}^{m} \sum_{k\ ,x_k \in G_i} u_{i,j} * \|x_k - c_i\|^2 \tag{4.28}$$

Representando esto la suma de las distancias al cuadrado de cada vector x_i perteneciente al grupo g_i.

Teniendo todo esto en cuenta, el problema de agrupamiento de datos se puede resolver al encontrar los valores de $u_{i,j}$ y c_i que minimicen J. Este método minimiza J a través de un proceso iterativo que se divide en dos fases:

Primera Fase

En esta fase se optimiza J respecto a $u_{i,j}$ manteniendo fijas las posiciones de los centroides c_i los cuales en la primera iteración son colocados de manera aleatoria, para esto se aprovecha que el valor de J varia linealmente respecto a los valores de $u_{i,j}$. Dado que cada término $u_{i,j}$ involucra un grupo independiente y diferente g_i, es posible encontrar de manera separada cada termino $u_{i,j}$. Con esto en cuenta se elijen los valores de $u_{i,j}$=1 al centroide que tenga la mínima distancia con el vector de características que está siendo analizado, es decir, se asigna el vector de características x_i al grupo más cercano. Esto se puede formular matemáticamente de la siguiente manera:

$$u_{i,j} = \begin{cases} 1, & i = argmin\|x_j - c_i\|^2 \\ 0, & cualquier\ otro\ caso \end{cases} \tag{4.29}$$

Segunda Fase

En esta fase, se determina el valor de los centroides manteniendo fijos los valores de $u_{i,j}$ previamente obtenidos. Dado que los valores de la función objetivo varían de forma cuadrática respecto al valor de los centroides c_i, el valor de J se puede

minimizar igualando a 0 el valor de la derivada de J con respecto a c_i. Obteniendo de esto la siguiente expresión:

$$2 * \sum_{j=1}^{n} u_{i,j} * (x_j - c_i) = 0 \tag{4.30}$$

A partir de la cual se puede obtener el valor de los centroides de la siguiente manera:

$$c_i = \frac{\sum_{j=1}^{n} u_{i,j} * x_j}{\sum_{j=1}^{n} u_{i,j}} \tag{4.31}$$

Una vez completada la segunda etapa, el proceso se repite iterativamente hasta que no haya cambios en los valores de los centroides. En la figura 4.3 se puede seguir el paso a paso del algoritmo K-medias a manera de diagrama de flujo.

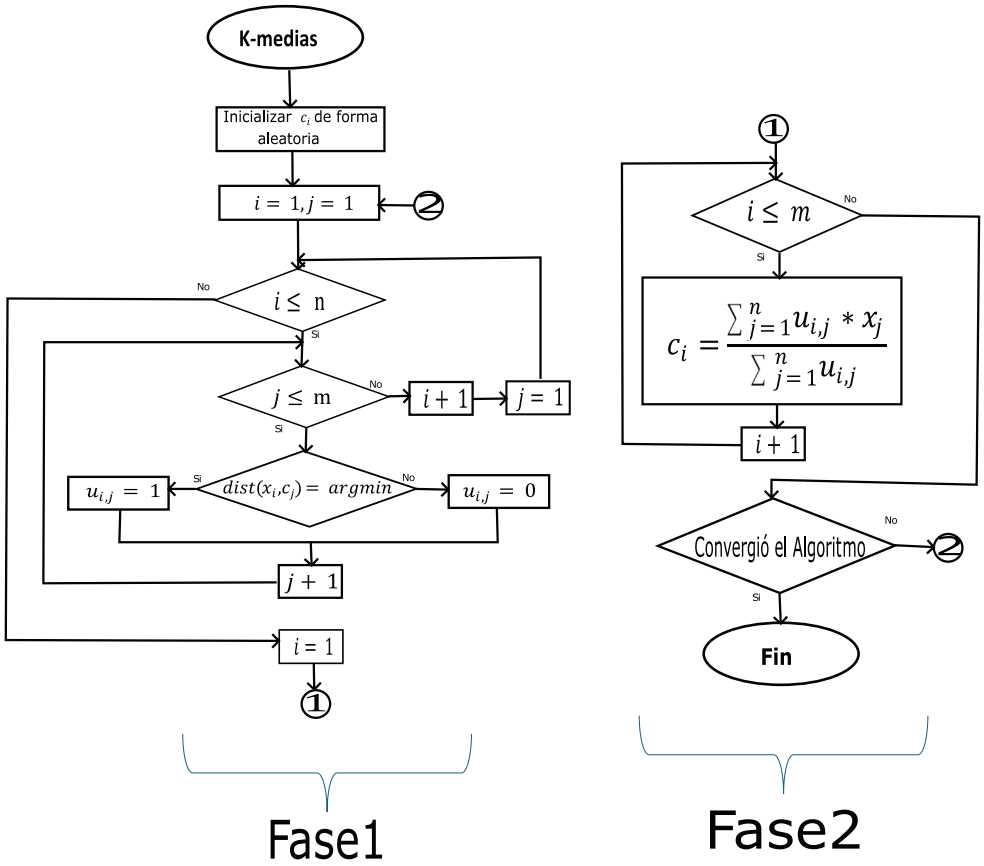

Figura 4.3 Diagrama de flujo del algoritmo K-medias, en él, la variable n es el número de vectores de características a ser analizados y m es el número de grupos definido por el usuario

Ejemplo 4.2

A continuación, se muestra un ejemplo de implementación del algoritmo K-medias en Python utilizado para separar el siguiente conjunto de datos en 4 grupos:

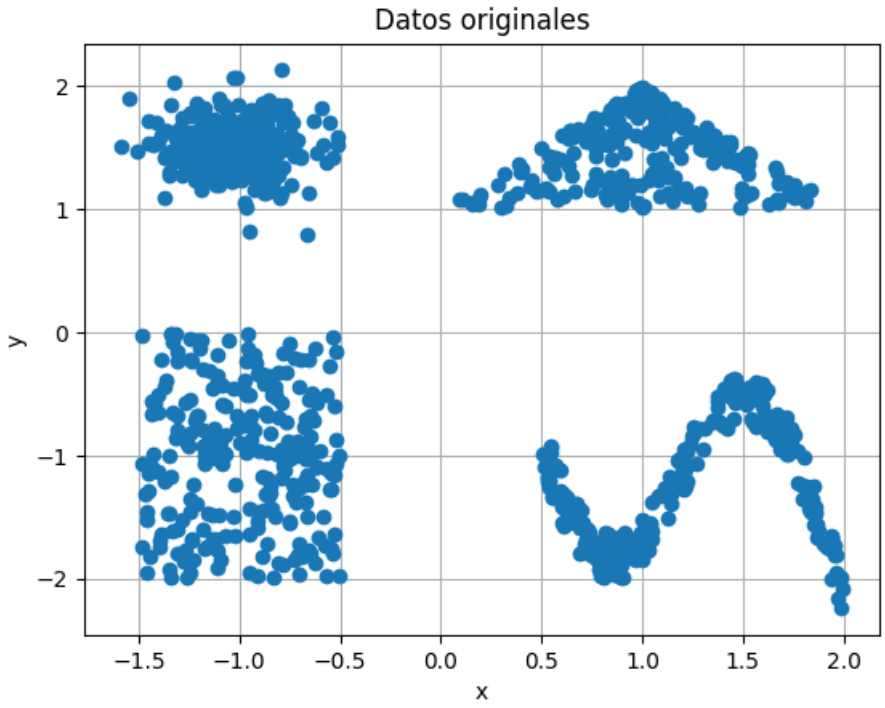

Figura 4.4 Conjunto de Datos a analizar por el algoritmo

Solución:

Código 4.3 Implementación del Algoritmo K-medias para agrupar un conjunto de datos

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.io import arff
import pandas as pd

# Leer el archivo ARFF
```

```
data, meta = arff.loadarff('shapes.arff')

# Convertir a DataFrame para manejo más cómodo
df = pd.DataFrame(data)

# Asegurarse de que los datos estén en formato numérico (por si hay
bytes)
df['x'] = pd.to_numeric(df['x'], errors='coerce')
df['y'] = pd.to_numeric(df['y'], errors='coerce')

# Extraer variables
x = df['x'].values
y = df['y'].values

# Visualizar datos originales
plt.figure()
plt.plot(x, y, 'o')
plt.title('Datos originales')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.show()

# Preparar datos para clustering
D = np.column_stack((x, y))

# Aplicar K-means con 4 clusters
kmeans = KMeans(n_clusters=4, random_state=0)
idx = kmeans.fit_predict(D)

# Visualización de resultados
plt.figure()
colors = ['g', 'r', 'k', 'y']
for i in range(4):
    cluster_points = D[idx == i]
    plt.plot(cluster_points[:, 0], cluster_points[:, 1], 'o' +
colors[i],\
 label=f'Cluster {i+1}')

# Graficar los centroides
```

```
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:,0],centroids[:,1],s=150,c='blue',marker='X',\
label='Centroides')

plt.title('Resultados de K-means')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```

Resultado:

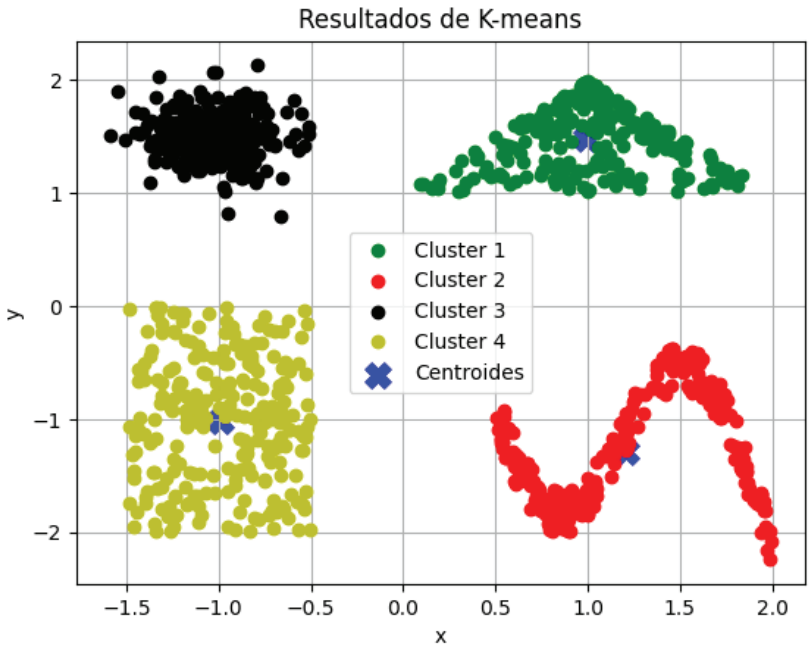

Figura 4.5 Resultado de Agrupamiento con algoritmo K-medias

Como se puede apreciar, el algoritmo separa el conjunto de datos de acuerdo con la distribución espacial de las nubes de datos asignándolos a su centroide más cercano. Es necesario aclarar que para que el código 4.3 funcione se necesita tener el archivo "shapes.arff" en la misma carpeta que se ejecute el código. Este archivo será añadido en el material complementario de la página web del libro.

4.5.4 Algoritmos de Agrupamiento Jerárquico

En esta sección serán analizados los algoritmos de agrupamiento jerárquico. Estos algoritmos realizan una serie de particiones sobre el conjunto de datos de tal manera que permite que sean visualizadas en un diagrama de árbol comúnmente conocido como dendrograma. Como se puede apreciar en la Figura 4.6, esta es una estructura configurada de tal manera en que cada elemento *D* del nivel más bajo del árbol representan un grupo individual, mientras que el elemento raíz D_r del árbol representa el grupo formado por la unión de todos los vectores de características a analizar. Tomando en cuenta estas consideraciones, en los niveles intermedios del árbol es posible encontrar diversas agrupaciones de los vectores de características dependiendo de las distancias relativas entre ellos.

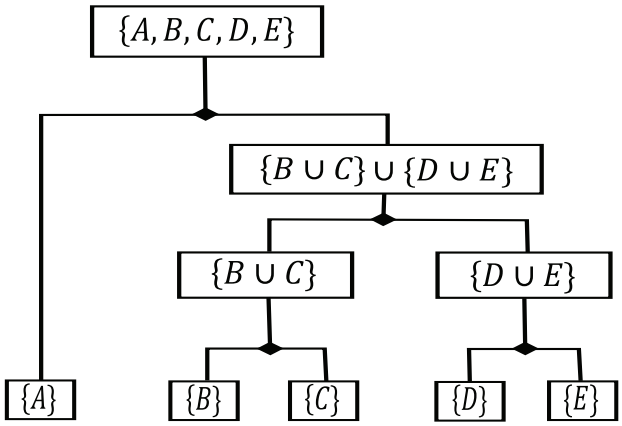

Figura 4.6 Ejemplo de Dendrograma

Este tipo de algoritmos se pueden dividir en dos enfoques principales: los métodos de aglomeración, los cuales empiezan por definir para cada uno de los *n* elementos a agrupar un grupo individual. Teniendo de esta manera *n* grupos y, con ellos, el algoritmo realiza un proceso iterativo de unión de los grupos que de acuerdo con un criterio de similitud tienden a estar más relacionados. Por la otra parte, los procesos de agrupamiento jerárquico divisivos realizan la tarea opuesta, es decir, el algoritmo inicia con un grupo principal que contiene todos los vectores de características a analizar y lo va separando en grupos que de acuerdo con un criterio de disimilitud son diferentes del grupo. Debido al carácter introductorio de este libro, en esta sección se abordará únicamente los algoritmos aglomerativos que son ampliamente más populares.

Formalmente se puede definir el agrupamiento jerárquico de la siguiente manera: dado un conjunto de datos $\boldsymbol{D}=\{x_1,x_2,\ldots,x_n\}$ para $xi\in\mathbb{R}^d$, el objetivo de

un algoritmo de agrupación jerárquica es particionar el conjunto $\boldsymbol{D}$ en m grupos $G=\{g_1,g_2,...,g_m\}$. Se dice que un conjunto de grupos $\boldsymbol{A}=\{a_1,a_2,...,a_s\}$ está anidado en otro conjunto $\boldsymbol{B}=\{b_1,b_2,...,b_t\}$ únicamente si $s>t$ de manera que cada grupo $a_i \in \boldsymbol{A}$ existe un grupo $b_j \in \boldsymbol{B}$ para el que $a_i \subseteq b_j$. Durante el agrupamiento aglomerativo, se produce una serie de particiones anidadas $p_1,p_2,...,p_n$ iniciando con $p_1=\{(x_1),(x_2),...,(x_n)\}$ en la que cada dato pertenece a un grupo individual. Después de esto, el algoritmo realizara un proceso iterativo hasta terminar con un grupo que incluya todo el conjunto de datos a analizar $p_n=\{(x_1,x_2,...,x_n)\}$. Tomando en cuenta estas definiciones es posible decir que la partición p_n se encuentra anidada en la partición p_{n+1}.

Al comienzo de la ejecución de los métodos de agrupamiento aglomerativos, la primera partición considera a cada uno de los vectores de características como un grupo individual. Después, con el avance del algoritmo, este irá uniendo los grupos de acuerdo con la similitud que presenten con base en una métrica dada. Dicho esto, suponiendo que iniciando con la partición $p_n=\{g_1,g_2,...,g_n\}$ se presentan al algoritmo dos grupos g_1 y g_2 a ser analizados, al ser estos grupos los que presentan mayor similitud, el algoritmo tomará la decisión de fusionarlos en uno solo, dando como resultado $g_{1,2}=g_1 \cup g_2$ y formando una partición nueva $p_{n+1}=\{g_1,g_2,...,g_n\}$ en la que los grupos g_1 y g_2 son sustituidos por la unión de estos $g_{1,2}$. De esta manera el proceso se ejecuta iterativamente hasta que se tiene la partición con un único grupo que incluye a todos los vectores de características. Tomando en cuenta que durante cada iteración el número de grupos se ve reducido en uno, el número de iteraciones totales del algoritmo será reducido a n pasos. Aunque es posible dejar un criterio de paro establecido para que el algoritmo se detenga en el número de grupos que de acuerdo con la separación natural de los pasos quede establecido. El proceso de un algoritmo se puede resumir de la siguiente manera:

1.	$P \leftarrow \{g_i = \{x_i\} \mid x_i \in D\}$
2.	$\delta\left(x_i, x_j\right)$
3.	Repetir
4.	$Encontrar(argmax\left(\delta\left(g_i, g_j\right)\right)$
5.	$g_{i,j}=g_i \cup g_j$
6.	Hasta $\lvert P \rvert = m$

Algoritmo 4.1 Pseudocódigo Procedimiento Algoritmo de Agrupación Jerarquico

Ejemplo 4.3

A continuación, se muestra un ejemplo de implementación del algoritmo de agrupamiento aglomerativo en Python utilizado para separar el siguiente conjunto de datos:

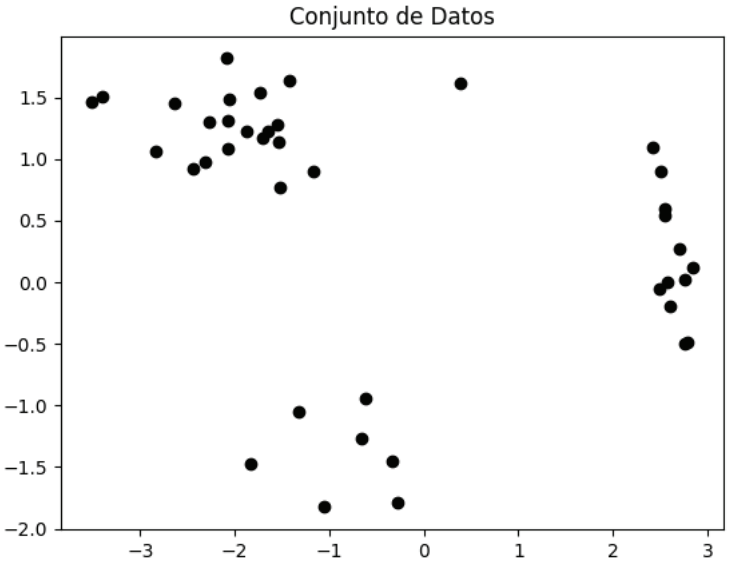

Figura 4.7 Conjunto de datos a analizar por algoritmo aglomerativo

Código 4.4 Implementación Algoritmo de Agrupamiento Aglomerativo en Python

```
Import numpy as np
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
from scipy.spatial.distance import pdist
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture



# Parámetros de las distribuciones
muD = np.array([[-2.1, 1.28], [-1.2, -1.40], [2.64, 0.19]])
sigmaD = np.array([
    [[0.59, 0], [0, 0.11]],
```

```
    [[0.49, 0], [0, 0.11]],
    [[0.05, 0], [0, 0.21]]
])
p = np.ones(3) / 3

# Generación de datos usando mezcla gaussiana
gmm = GaussianMixture(n_components=3, covariance_type='full')
gmm.weights_ = p
gmm.means_ = muD
gmm.covariances_ = sigmaD
gmm.precisions_cholesky_ = np.linalg.cholesky(np.linalg.inv(sigmaD))
X, _ = gmm.sample(40)

# Cálculo de distancias y clustering jerárquico
Y = pdist(X)
Z = linkage(Y, method='ward')  # MATLAB usa 'ward' por defecto

# Dendrograma
plt.figure()
dendrogram(Z)
plt.title("Dendrograma")
plt.show()

# Clasificación en 3 grupos
T = fcluster(Z, 3, criterion='maxclust')

# Índices de cada grupo
Indice1 = np.where(T == 1)[0]
Indice2 = np.where(T == 2)[0]
Indice3 = np.where(T == 3)[0]

# Gráfico por grupo
plt.figure()
plt.plot(X[Indice1, 0], X[Indice1, 1], 'o', label='Cluster 1')
plt.plot(X[Indice2, 0], X[Indice2, 1], 'x', label='Cluster 2')
plt.plot(X[Indice3, 0], X[Indice3, 1], 'D', label='Cluster 3')
plt.title("Grupos separados")
plt.legend()
plt.show()
```

Resultado:

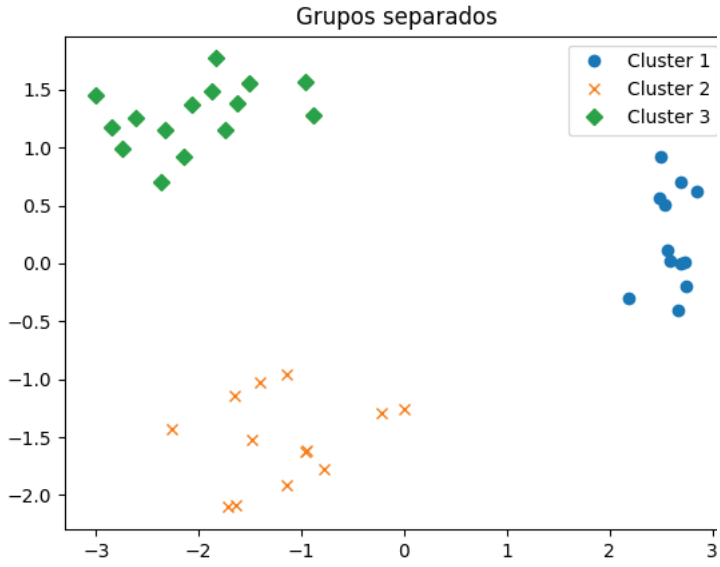

Figura 4.8 Conjunto de datos separado después de ser analizado por el algoritmo de agrupamiento aglomerativo

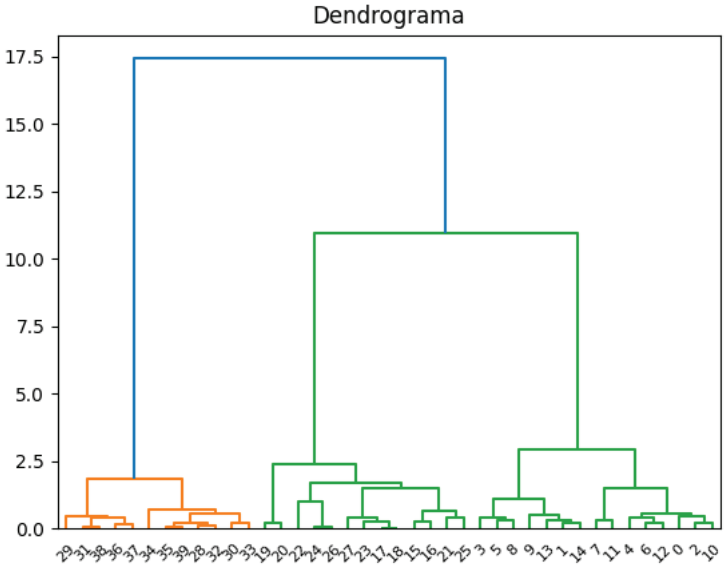

Figura 4.9 Dendrograma Resultante después de analizar el conjunto de datos con algoritmo aglomerativo

4.6 REFERENCIAS

[1] R. Xu and D. Wunsch, "A survey of clustering algorithms," IEEE Transactions on Neural Networks, vol. 16, no. 3, pp. 645–678, May 2005.

[2] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," arXiv preprint arXiv:1109.2378, Sep. 2011.

[3] N. Monath, et al., "Scalable Hierarchical Agglomerative Clustering," arXiv preprint arXiv:2010.11821, Oct. 2020.

[4] F. Murtagh and P. Legendre, "Ward's Hierarchical Clustering Method: Clustering Criterion and Agglomerative Algorithm," arXiv preprint arXiv:1111.6285, Nov. 2011.

[5] S. Dudoit and J. Fridlyand, "Bagging to improve the accuracy of a clustering procedure," Bioinformatics, vol. 19, no. 9, pp. 1090–1096, Sep. 2003.

[6] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A Comparative Study of Efficient Initialization Methods for the k-Means Clustering Algorithm," arXiv preprint arXiv:1209.1960, 2012.

[7] R. A. Fisher, "Clustering Algorithms I: Sequential Algorithms," in Pattern Recognition, Academic Press, 2007, pp. 331-344.

[8] E. A. Reyes and M. J. del Jesus, "A closer look into sequential clustering algorithms," International Journal of Intelligent Computing and Applications, vol. 6, no. 2, pp. 103–121, 2014.

[9] T. Zhang, "An overview of clustering methods with guidelines for application," Data Science Reports, vol. 45, pp. 123–138, 2023.

[10] Wikipedia contributors, "Cluster analysis," Wikipedia, The Free Encyclopedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Cluster_analysis