Sistemas Inteligentes

Proyecto Final:

“K-Means Clustering”

Kennia Jackeline Sánchez Castillo A00517129

Profe. Luis Ricardo Peña Llamas

Monterrey, Nuevo León

7 de Junio, 2024

Objetivo

Aplicar tus conocimientos en Clustering:

- ➔ Programar K-Means

El objetivo de este proyecto fue aplicar conocimientos en clustering mediante la programación del algoritmo K-means. Este método de agrupamiento no supervisado se utiliza para particionar un conjunto de datos en *k* clusters, minimizando la suma de las distancias cuadradas de los puntos a sus centroides respectivos.

Introducción

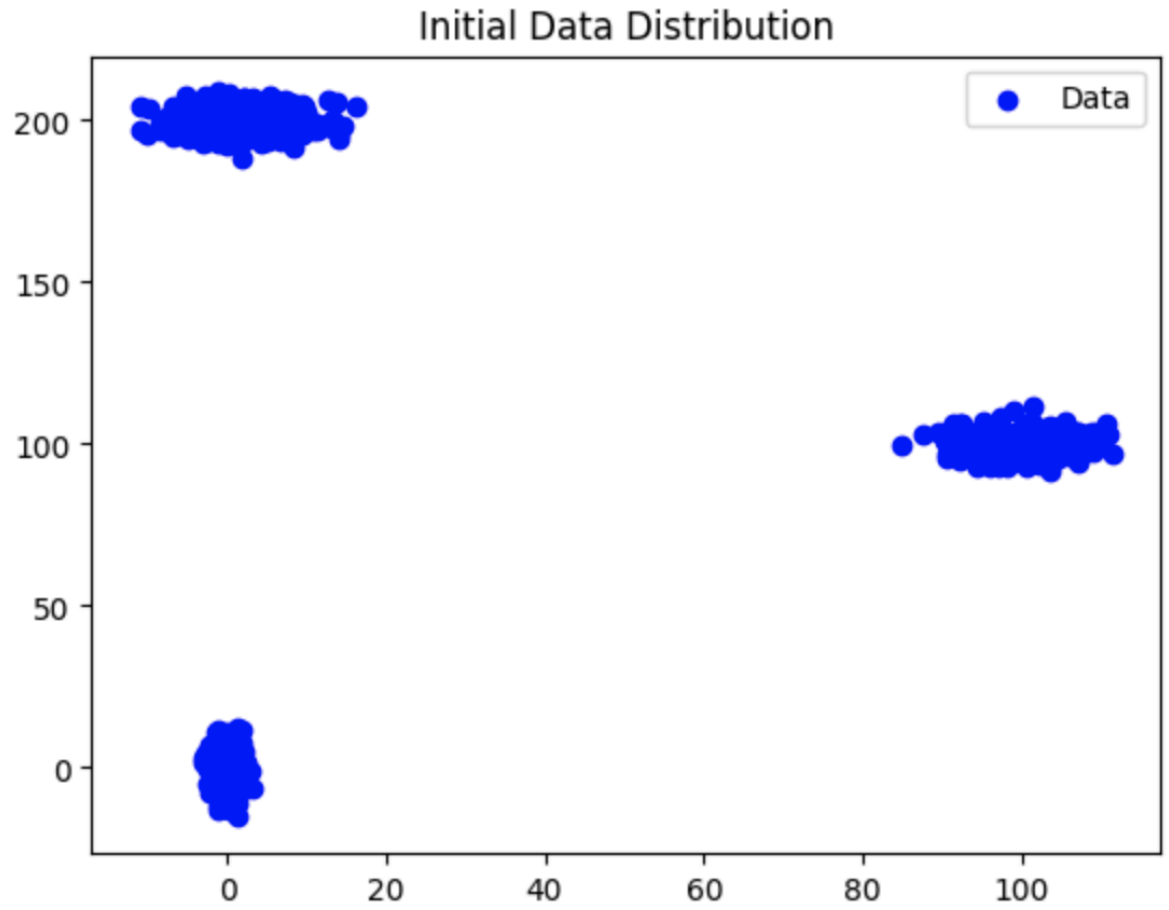
El algoritmo K-means es un método de agrupamiento no supervisado que se utiliza para particionar un conjunto de datos en *k* clusters (o grupos). Cada cluster está representado por el centroide, que es la media de los puntos en ese cluster. El objetivo es minimizar la suma de las distancias cuadradas de los puntos a sus centroides respectivos.

Los pasos para lograr el algoritmo K-means son los siguientes:

- ➔ Inicialización: Seleccionar *k* centroides iniciales, que pueden elegirse de forma aleatoria o mediante algún método heurístico.
- ➔ Asignación (E-Step): Asignar cada punto de datos al centroide más cercano. Esto crea *k* clusters basados en la proximidad.
- ➔ Actualización (M-Step): Calcular los nuevos centroides de los clusters formados. El nuevo centroide de cada cluster es la media de todos los puntos de datos en ese cluster.
- ➔ Convergencia: Repetir los pasos 2 y 3 hasta que los centroides no cambien significativamente, es decir, hasta que se alcance la convergencia.

Desarrollo

Para este proyecto lo que primero empezamos haciendo es cargar el archivo de datos y después poner un código para poder visualizar más fácilmente cómo están distribuidos los datos.



El siguiente paso es la elaboración e implementación del algoritmo K-means con los pasos vistos en la introducción.

- ➔ Inicializamos los 3 centroides de manera aleatoria y repetimos para que no haya cambios.

E-Step

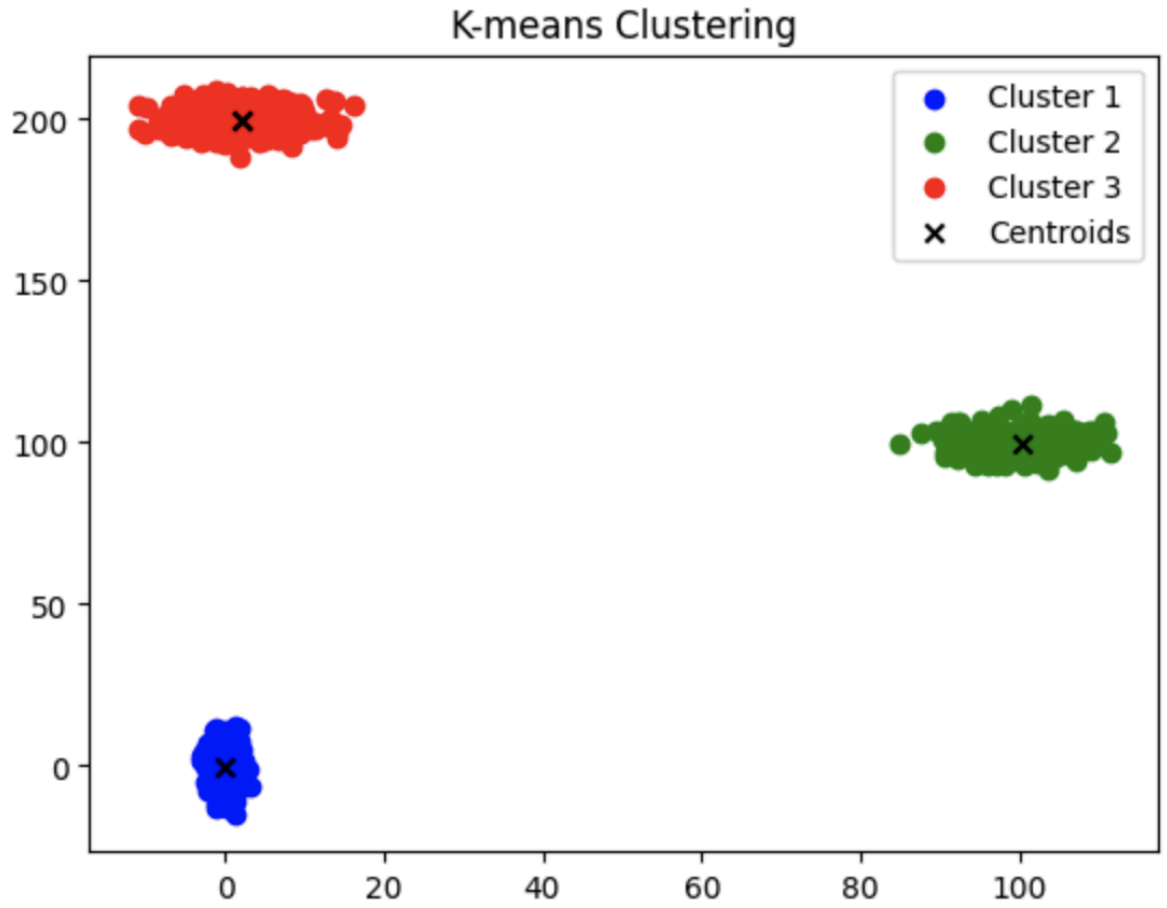
- ➔ Medir la distancia euclidiana con respecto a cada centroide y asignar el dato al centroide más cercano.

M-Step

- ➔ Calcular la media de todos los datos que pertenecen a ese cluster y el nuevo centroide será la media de todos los datos del cluster.

Repetimos los pasos del E-Step y M-Step hasta que los centroides no cambien mucho.

Por último mostramos los datos obtenidos en un diagrama.



Código

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Cargar los datos
data = pd.read_csv('clustering_data.csv')

# Visualizar los datos
plt.scatter(data.iloc[:, 0], data.iloc[:, 1], c='blue', label='Data')
plt.title('Initial Data Distribution')
plt.legend()
plt.show()

# Inicializar aleatoriamente 3 centroides
np.random.seed(42)  # Para reproducibilidad
k = 3
centroids = data.sample(n=k).to_numpy()
```

```
# Función para la distancia euclidiana
def euclidean_distance(a, b):
    return np.linalg.norm(a - b)

# Funición para algoritmo K-means
def kmeans(data, centroids, k):
    data_np = data.to_numpy()
    n_samples = data_np.shape[0]
    labels = np.zeros(n_samples)
    while True:

        # E-Step: Asignar cada punto al centroide más cercano
        for i in range(n_samples):
                  distances = [euclidean_distance(data_np[i], centroid) for
centroid in centroids]
            labels[i] = np.argmin(distances)

        # M-Step: Recalcular los centroides
         new_centroids = np.array([data_np[labels == j].mean(axis=0) for j in
range(k)])

        # Verificar si los centroides han cambiado
        if np.all(centroids == new_centroids):
            break
        centroids = new_centroids

    return centroids, labels

# Ejecutar K-means
final_centroids, labels = kmeans(data, centroids, k)

# Visualizar los resultados
colors = ['blue', 'green', 'red']
for i in range(k):
    points = data.to_numpy()[labels == i]
        plt.scatter(points[:, 0], points[:, 1], c=colors[i], marker='o',
label=f'Cluster {i+1}')
       plt.scatter(final_centroids[i, 0], final_centroids[i, 1], c='black',
marker='x')
plt.scatter(final_centroids[i,   0],   final_centroids[i,   1],   c='black',
marker='x', label='Centroids')
plt.title('K-means Clustering')
plt.legend()
plt.show()
```

Explicación Código

Se importan las librerías necesarias:

pandas: Para manipulación y análisis de datos.

numpy: Para operaciones matemáticas y manejo de arrays.

matplotlib.pyplot: Para visualización de datos.

Cargar y visualizar los Datos:

Los datos se cargan desde un archivo CSV llamado *clustering_data.csv* en un DataFrame de *pandas*.

Se visualizan los datos en un gráfico de dispersión, donde los puntos son de color azul y se les da la etiqueta *'Data'*.

Inicializar aleatoriamente los centroides:

Se establece una semilla aleatoria con *np.random.seed(42)* para que los resultados sean reproducibles.

Se seleccionan 3 puntos aleatorios de los datos para usarlos como los centroides iniciales. Estos puntos se convierten en un array de *numpy*.

Definir la función de distancia euclidiana:

Se define una función que calcula la distancia euclidiana entre dos puntos a y b utilizando *numpy.linalg.norm*.

Implementar el algoritmo K-means:

Convierte los datos a un array de *numpy*.

Inicializa un array *labels* para almacenar las etiquetas de los clusters para cada punto de datos.

Inicia un bucle *while True* que se repetirá hasta que los centroides no cambien.

E-Step:

Para cada punto de datos, se calculan las distancias a cada centroide.

Asigna el punto al centroide más cercano actualizando el array *labels*.

M-Step:

Para cada cluster, re-calcula el centroide tomando la media de todos los puntos asignados a ese cluster.

Verificación:

Si los centroides no han cambiado, el algoritmo ha convergido y se sale del bucle.

Si han cambiado, se actualizan los centroides y se repite el proceso.

Ejecutar K-means y visualizar resultados:

Se ejecuta la función *kmeans* con los datos y los centroides iniciales.

Se obtienen los centroides finales y las etiquetas de los clusters.

Se visualizan los resultados en un gráfico de dispersión:

Los puntos se colorean según su clúster.

Los centroides finales se marcan con una *'x'* negra.

Se añade una leyenda para identificar los clusters y los centroides.

Conclusión

El proyecto mostró cómo se puede implementar y aplicar el algoritmo K-means para agrupar datos de manera efectiva. A través de la visualización de los clusters formados y los centroides finales, se pudo apreciar cómo los puntos de datos se agruparon de acuerdo con sus proximidades en el espacio. Esta implementación básica de K-means proporciona una base sólida para entender cómo funciona el agrupamiento no supervisado y puede servir como punto de partida para aplicaciones más complejas en análisis de datos y aprendizaje automático.

El proyecto me resultó bastante interesante y desafiante. Al principio, no sabía cómo empezar, ya que solo tenía conocimientos básicos sobre las librerías y cómo mostrar la información en una gráfica. Después, investigué más a fondo sobre el algoritmo K-means y encontré varias páginas que lo explicaban de manera clara y precisa. Gracias a esta investigación, pude entender mejor el algoritmo y comenzar con el desarrollo del proyecto.

Referencias

Universidad de Oviedo. (s.f.). *Algoritmo K-means*. https://www.unioviedo.es/compnum/laboratorios_py/kmeans/kmeans.html

Keita, Z. (2023, Enero 17). How to perform KMeans clustering using python. Towards Data Science. https://towardsdatascience.com/how-to-perform-kmeans-clustering-using-python-7cc296cec092

Na. (2018, Marzo 12). K-Means en Python paso a paso. Aprende Machine Learning. https://www.aprendemachinelearning.com/k-means-en-python-paso-a-paso/