Guía de Clustering K-means desde Cero en Python

Profesor: Abel Alvarez

2025-02-13

Intrucciones

Antes de comenzar a realizar el taller, se debe crear un repositorio en Github y compartirlo con el usuario abelalv (usuario de Github). El nombre del repositorio debe ser **Proyecto1**. Cada vez que haga un cambio al taller significativo, se debe hacer un commit y un push al repositorio. Cuando se termine el taller se debe subir en le ink de actividades de la brigespace el link del repositorio. Recuerde, que desde el momento que se comparte el repositorio, se va a estar revisando los commits y push realizados.

Punto 1

Imagina que formas parte de un equipo de análisis de datos y deseas explorar técnicas de *clustering*. Se te pide implementar el algoritmo **k-means** sin utilizar librerías especializadas para el clustering (aunque sí para el manejo de datos y visualización). Para ello, se proporcionarán datos sintéticos (dummy) con 5 clusters en espacios de 2D y 3D, en los que se conoce la etiqueta verdadera de cada punto. Esto permitirá comparar los resultados del algoritmo con las etiquetas originales.

Además, se requiere que experimentes con diferentes distancia para analizar cómo influye la elección de la métrica en la asignación de clusters.

a. Carga los datos que se encuentran en lo archivo 'data_2d.csv' (debes hacer lo mismo con el archivo'data_3d.csv'), para ello puedes usar el siguiente código:

importar libreria

import pandas as pd import numpy as np # Lectura de los datos 2D data_2d = pd.read_csv(“data_2d.csv”) print(“Datos 2D:”) print(data_2d.head())

este comando leer el archivo ‘data_2d.csv’ y muestra las primeras filas del archivo, recuerda que la lectura de los datos es un formato data frame. Realiza el mismo procedimiento para el archivo ‘data_3d.csv’.

b. Realiza un estudio estadistico de las variables, para ello puedes usar el siguiente comando:

```
# Estudio estadistico de los datos 2D
print("Estudio estadistico de los datos 2D:")
print(data_2d.describe())
```

c. Realiza una gráfica de los datos 2D y 3D, para ello puedes usar el siguiente comando:

```
# importar libreria
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
# Gráfica de los datos 2D
plt.figure(figsize=(8, 6))
plt.scatter(data_2d["x"], data_2d["y"], s=50)
plt.title("Datos 2D")

# Gráfica de los datos 3D
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(data_3d["x"], data_3d["y"], data_3d["z"], s=50)
ax.set_title("Datos 3D")
plt.show()
```

d. Implementa el algoritmo **k-means** en Python. Para ello, puedes seguir los siguientes pasos:

1. Inicializa los centroides de manera aleatoria, para que el experimento sea reproducible debes fijar la semillas, que es un número que se utiliza para inicializar el generador de números aleatorios. Para ello puedes usar el siguiente comando:

```
# Fijar la semilla
np.random.seed(42)
```

2. Asigna cada punto al centroide más cercano.

3. Actualiza los centroides como el promedio de los puntos asignados.
4. Repite los pasos 2 y 3 hasta que los centroides no cambien o se alcance un número máximo de iteraciones.

e. Evalua el rendimiento del algoritmo **k-means** en los datos 2D y 3D. Para ello, para ello debes medir la inercia que esta dada por la siguiente formula

$$inercia = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i|\,|_q^q$$

donde k es el número de clusters, C_i es el conjunto de puntos asignados al cluster i, μ_i es el centroide del cluster i, $||x - \mu_i||_q^q = \sum_{j=1}^{N} |x_j - (\mu_i)_j|^q$ representa la distancia entre el punto x y su centroide μ_i. donde N es el número de dimensiones del espacio y q es la métrica de distancia utilizada.

f. Experimenta con diferentes distancias (euclidiana, Manhattan, Chebyshev, etc.) y analiza cómo influye la elección de la métrica en la asignación de clusters. **metricas sugeridas:**

g. $d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ (euclidiana)

ii. $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$ (Manhattan)
iii. $d(x, y) = \max_i |x_i - y_i|$ (Chebyshev)
iv. $d(x, y) = (x - y)^T A(x - y)$, donde A es una matriz definida positiva (Mahalanobis)

g. Realiza un análisis de los resultados y concluye sobre la calidad del clustering obtenido. Aunque este algoritmo no es supervisado, puedes comparar los resultados con las etiquetas verdaderas para evaluar su desempeño.

h. Realiza un analisis de los resultados si cambias las semillas de los centroides.

Punto 2

Con base en la conclusion del ejercicio anterior, realiza un análisis de clustering con el algoritmo **k-means** en un conjunto de datos real. Para ello, debes seguir los siguientes pasos:

a. Carga los datos que se encuentran en el archivo 'data_real.csv'

b. Realiza un estudio estadistico de las variables.

c. Realiza una gráfica de los datos.

d. Implementa el algoritmo **k-means** para 2, 3, 4, ..., 10 clusters.

e. Utiliza la inercia para seleccionar el número óptimo de clusters. Para ello, puedes utilizar la regla del codo. El cual consiste en graficar la inercia en función del número de clusters y seleccionar el punto donde la inercia deja de disminuir rápidamente

f. Realiza un análisis de los resultados y concluye sobre la calidad del clustering obtenido.

Características del conjunto de datos:

La siguiente base de datos tiene llamada Mall_Customers.csv, la cual contiene información de clientes de un centro comercial. Las variables son las siguientes:

1. CustomerID: Identificación única
2. Genero: Género del cliente
3. Edad: Edad del cliente
4. Ingresos Anuales (k$): Ingresos anuales del cliente
5. Puntuación de Gastos (1-100): Puntuación asignada por el centro comercial basada en el comportamiento del cliente y la naturaleza del gasto

para poder estudiar k-means lo primero que se debe hacer es cargar los datos, realizar un estudio estadistico de las variables. Para este ejercicio vamos a ignorar las variables CustomerID y Genero, (variables categoricas) y vamos a realizar el clustering con las variables Edad, Ingresos Anuales (k$) y Puntuación de Gastos (1-100). Una de las mejores practicas es normalizar los datos antes de realizar el clustering. Lo cual consiste en restar la media y dividir por la desviación estandar. Eso lo puedes hacer con el siguiente comando:

```
# Leer el archivo CSV
mall_data = pd.read_csv('Mall_Customers.csv')

# Mostrar las primeras filas del DataFrame
print("Datos de Mall Customers:")
print(mall_data.head())

# Eliminar las variables categóricas
mall_data_numeric = mall_data.select_dtypes(include=[np.number])
mall_data_numeric = mall_data_numeric.drop(columns=['CustomerID'])

# Centralizar las variables restantes
mall_data_centered = mall_data_numeric - mall_data_numeric.mean()
# Dividir por la desviación estándar de cada columna
mall_data_standardized = mall_data_centered / mall_data_centered.std()

# Mostrar las primeras filas del DataFrame estandarizado
print("Datos estandarizados de Mall Customers:")
print(mall_data_standardized.head())
```