Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Santa Fe

Analítica de datos y herramientas de inteligencia artificial I (Gpo 102)

Proyecto 4 K-means

Alumno:
Francisco Antonio López Ricardez - A01737275

Profesor:
Víctor Manuel de la Cueva Hernández

septiembre 15, 2025

Introducción

En este documento se explica el desarrollo de un proyecto que implementa el algoritmo K-means en Python, tanto para agrupar puntos en un plano 2D como para realizar compresión de imágenes en formato RGB.

El proyecto tiene como objetivo:

- Aplicar K-means en datos bidimensionales para visualizar los grupos formados.
- Implementar la compresión de imágenes reduciendo la cantidad de colores de una imagen de 24 bits a 16 colores representativos (4 bits por pixel).
- Mostrar comparaciones entre la imagen original y la comprimida.

Instalación de dependencias

Este proyecto requiere Python 3.8 o superior, también se recomienda actualizar pip:

```
python -m pip install --upgrade pip
```

Instalación de librerías necesarias

- matplotlib: para graficación de datos e imágenes.
- numpy: operaciones con vectores y matrices.
- csv: lectura de datos en archivos .csv o .txt.
- random y math: operaciones numéricas auxiliares.

Instalación:

```
pip install matplotlib numpy
```

(csv, functools y math ya vienen incluidos en Python).

Estructura del proyecto

Los archivos utilizados son:

- **kMeans.py:** archivo principal donde se encuentran todas las funciones.
- **ex7data2.txt:** archivo de prueba con puntos bidimensionales.
- **bird_small.png:** imagen utilizada para el ejercicio de compresión.
- **prueba.py**: archivo de ejemplo para el uso de las funciones de kMeans.py**.**

Descripción de funciones

`euclideanDistance(list1, list2)`

- **Función:** Calcula la distancia euclidiana entre dos vectores de cualquier dimensión.
- **Parámetros:**
  - `list1`: primer vector.
  - `list2`: segundo vector.

- **Retorna:** La distancia euclidiana entre los vectores.

`lector_csv(nombre_archivo)`

- **Función:** Lee un archivo CSV o TXT separado por espacios y devuelve una lista con los datos numéricos.
- **Parámetros:**
  - `nombre_archivo`: nombre o ruta del archivo.
- **Retorna:** Lista de listas con los datos numéricos.

`kMeansInitCentroids(X, K)`

- **Función:** Selecciona aleatoriamente `K` ejemplos del conjunto de datos `X` para inicializar los centroides.
- **Parámetros:**
  - `X`: conjunto de datos.
  - `K`: número de centroides.
- **Retorna:** Lista con los centroides iniciales.

`findClosestCentroids(X, initial_centroids)`

- **Función:** Asigna cada ejemplo del dataset `X` al centroide más cercano.
- **Parámetros:**
  - `X`: datos de entrada.
  - `initial_centroids`: centroides actuales.
- **Retorna:** Lista con los índices de centroides asignados a cada punto.

`computeCentroids(X, idx, K)`

- **Función:** Recalcula la posición de cada centroide como el promedio de los puntos asignados a él.

- **Parámetros:**
  - `X`: conjunto de datos.
  - `idx`: lista de asignaciones de cada punto.
  - `K`: número de centroides.

- **Retorna:** Nueva lista con las posiciones de los centroides.

`runkMeans(X, initial_centroids, max_iters)`

- **Función:** Ejecuta el algoritmo K-means de forma iterativa.

- **Pasos principales:**

  - Inicializar los centroides.
  - Repetir durante `max_iters`:
    - Asignar cada punto a un centroide.
    - Recalcular centroides.
  - Graficar los puntos coloreados por clúster y los centroides finales.

- **Parámetros:**

  - `X`: conjunto de datos.
  - `initial_centroids`: centroides iniciales.
  - `max_iters`: número máximo de iteraciones.
- **Retorna:** Lista con los centroides finales y las asignaciones.

Ejemplo de uso

En el archivo prueba.py se muestra un ejemplo de uso, donde de la línea 12 a la 15 se ejecuta k-means con el set de datos de “ex7data2.txt”. De la línea 17 a la 31 se usa la imagen de “bird small.png” para encontrar los K (línea 21) colores más significativos usando k-means.

```
11 if __name__ == "__main__":
12   #------------data
13   x = lector_csv("ex7data2.txt")
14   centroids = kMeansInitCentroids(x,3)
15   runkMeans(x, centroids, 10)
16   #---------imagen
17   A = imread("bird_small.png").astype(float)
18   if A.max() <= 1.0:
19       A = (A * 255).astype(np.uint8)
20   X = A.reshape(-1, 3)
21   k = 16
22   initial_centroids = kMeansInitCentroids(X, k)
23   centroids, idx = runkMeans(X, initial_centroids, 10)
24   X_compressed = np.array([centroids[i] for i in idx])
25   X_compressed = X_compressed.reshape(A.shape)
26   fig, ax = plt.subplots(1, 2, figsize=(8, 4))
27   ax[0].imshow(A)
28   ax[0].set_title(" imagen Original")
29   ax[1].imshow(X_compressed.astype('uint8'))
30   ax[1].set_title(f"imagen comprimida con {k} colores")
31   plt.show()
```