Centro de Enseñanza Técnica Industrial

Plantel Colomos

Ingeniería en Desarrollo de Software

Inteligencia Artificial

REPORTE DE PRÁCTICA

IDENTIFICACIÓN DE LA PRÁCTICA

Práctica	**5**	**Nombre de la práctica**	**k-means multidimensional**
Fecha	**07/04/2025**	**Nombre del profesor**	**Alma Nayeli Rodríguez Vázquez**
Nombre del estudiante	**Jesús Alberto Aréchiga Carrillo**		

OBJETIVO

El objetivo de esta práctica consiste en implementar el algoritmo k-means multidimensional para clustering.

PROCEDIMIENTO

Realiza la implementación siguiendo estas instrucciones.

Implementa el algoritmo k-means en Python. Para ello, considera los siguientes requerimientos:

1. Utiliza el set de datos “dataSet_kmeans2D_iris.txt”
2. Implementa el algortimo de clustering k-means utilizando la función `KMeans()` de `sklearn`.
3. Utiliza los siguientes valores para el parámetro `n_clusters`:
   `n_clusters = 2`, `n_clusters = 3` y `n_clusters = 4`
4. Dibuja los clusters y centroides con diferentes colores.
5. Reporta el valor de los centroides. Además, reporta el número de datos que tiene cada cluster en cada prueba.

IMPLEMENTACIÓN

Agrega el código de tu implementación aquí.

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

data = pd.read_csv('dataSet_kmeans2D_iris.csv')
data.head()

x = data.values
plt.plot(x[:,0], x[:,1], 'yo')

media = x.mean(axis=0)
sigma = x.std(axis=0, ddof=1)
x = (x-media)/sigma
```

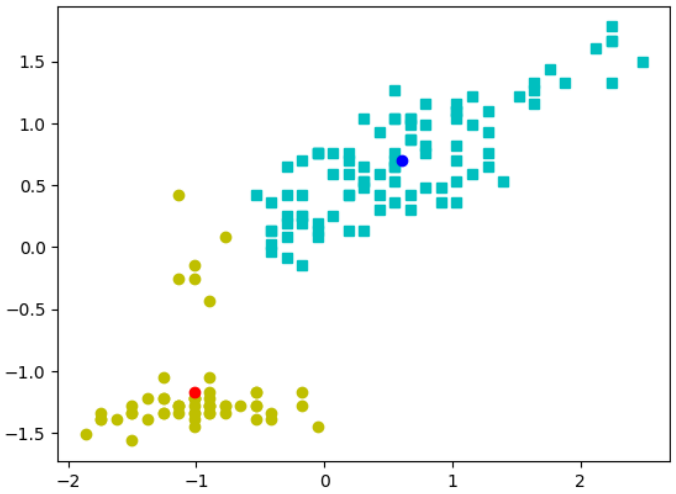
```
model = KMeans(n_clusters=2)
model.fit(x)

c1 = model.cluster_centers_[0,:]
c2 = model.cluster_centers_[1,:]
c1, c2

cluster1 = np.argwhere = (model.labels_ == 0)
cluster2 = np.argwhere = (model.labels_ == 1)

plt.plot(x[cluster1,0], x[cluster1,1], 'yo')
plt.plot(x[cluster2,0], x[cluster2,1], 'cs')
plt.plot(c1[0], c1[1], 'ro')
plt.plot(c2[0], c2[1], 'bo')
plt.show()

model = KMeans(n_clusters=3)
model.fit(x)

c1 = model.cluster_centers_[0,:]
c2 = model.cluster_centers_[1,:]
c3 = model.cluster_centers_[2,:]
c1, c2, c3

cluster1 = np.argwhere = (model.labels_ == 0)
cluster2 = np.argwhere = (model.labels_ == 1)
cluster3 = np.argwhere = (model.labels_ == 2)

plt.plot(x[cluster1,0], x[cluster1,1], 'yo')
plt.plot(x[cluster2,0], x[cluster2,1], 'cs')
plt.plot(x[cluster3,0], x[cluster3,1], 'gd')
plt.plot(c1[0], c1[1], 'ro')
plt.plot(c2[0], c2[1], 'bo')
plt.plot(c3[0], c3[1], 'mo')
plt.show()

model = KMeans(n_clusters=4)
model.fit(x)

c1 = model.cluster_centers_[0,:]
c2 = model.cluster_centers_[1,:]
c3 = model.cluster_centers_[2,:]
c4 = model.cluster_centers_[3,:]
c1, c2, c3, c4

cluster1 = np.argwhere = (model.labels_ == 0)
cluster2 = np.argwhere = (model.labels_ == 1)
cluster3 = np.argwhere = (model.labels_ == 2)
cluster4 = np.argwhere = (model.labels_ == 3)

plt.plot(x[cluster1,0], x[cluster1,1], 'yo')
plt.plot(x[cluster2,0], x[cluster2,1], 'cs')
```

```
plt.plot(x[cluster3,0], x[cluster3,1], 'gd')
plt.plot(x[cluster4,0], x[cluster4,1], 'k*')
plt.plot(c1[0], c1[1], 'ro')
plt.plot(c2[0], c2[1], 'rs')
plt.plot(c3[0], c3[1], 'rd')
plt.plot(c4[0], c4[1], 'r*')
plt.show()
```

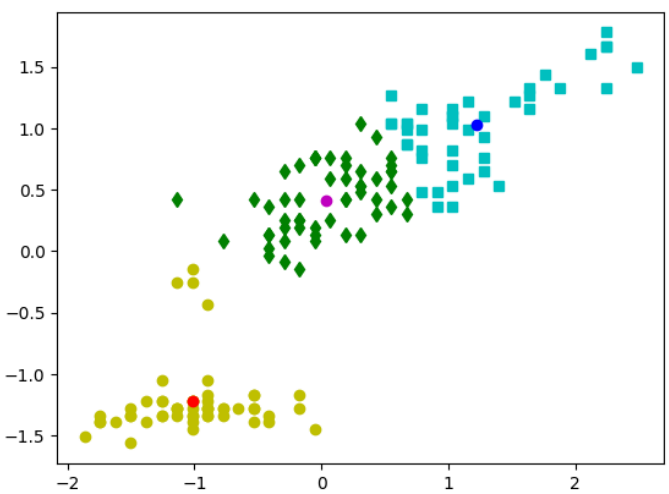
RESULTADOS

Agrega la(s) imagen(es) con los resultados obtenidos en los espacios indicados.

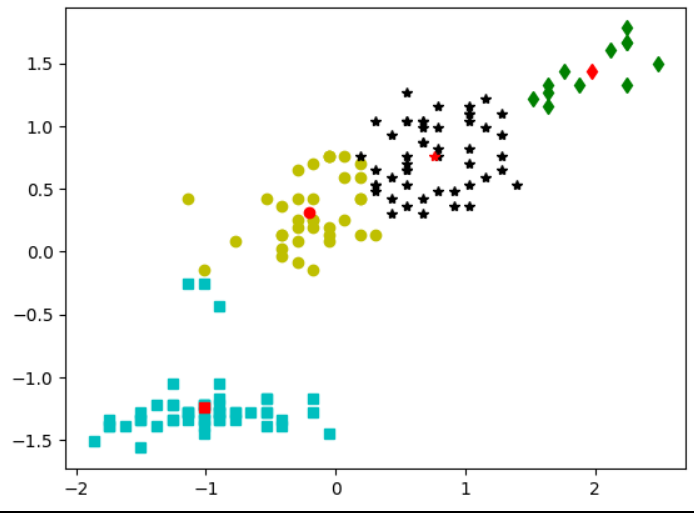
Gráfica de los datos agrupados para `n_clusters = 2`	Valores de los centroides y cantidad de datos en cada cluter para `n_clusters = 2`

	`c1 = model.cluster_centers_[0,:]`
	`c2 = model.cluster_centers_[1,:]`
	`c1, c2`
	
	`(array([-1.00981123, -1.17182686]), array([0.60158967, 0.69810962]))`

Gráfica de los datos agrupados para `n_clusters = 3`	Valores de los centroides y cantidad de datos en cada cluter para `n_clusters = 3`

	`c1 = model.cluster_centers_[0,:]`
	`c2 = model.cluster_centers_[1,:]`
	`c3 = model.cluster_centers_[2,:]`
	`c1, c2, c3`
	
	`(array([-1.01172811, -1.2244919 ]),`
	`array([1.22551348, 1.02513844]),`
	`array([0.0365328 , 0.41587943]))`

Gráfica de los datos agrupados para `n_clusters = 4`	Valores de los centroides y cantidad de datos en cada cluter para `n_clusters = 4`

	`c1 = model.cluster_centers_[0,:]`
	`c2 = model.cluster_centers_[1,:]`
	`c3 = model.cluster_centers_[2,:]`
	`c4 = model.cluster_centers_[3,:]`
	`c1, c2, c3, c4`
	
	`(array([-0.19993036, 0.3131025 ]),`
	`array([-1.01160152, -1.24482946]),`
	`array([1.97045455, 1.44032188]),`
	`array([0.75850855, 0.76368181]))`

CONCLUSIONES

Escribe tus observaciones y conclusiones.
El algoritmo de kmeans nos sirve para agrupar un grupo muy grande de datos en diferentes grupos (o clusters). Una limitación del método kmeans() de sklearn es que, para ciertos grupos de datos, no se separan de la manera más correcta, para eso hay que implementar el algoritmo manualmente.