Práctica 2.3 K-Means en 3D

Abarca Romero José Ángel

Lógica Difusa

2TM9

Gráficas:

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Ilustración Gráfica de los puntos a clasificar, centroides originales y centroides finales

Código de Python:

import numpy as np

import matplotlib.pyplot as plt

import math

import random

#Generación de puntos aleatorios entre 0 y 10

xi = np.zeros((2,30))

cont = 0

for i in range(0,2,1):

    for j in range(0,30,1):

        if i == 0:

            if cont < 10:

                xi[i][j] = random.randint(0,30)

            elif cont >= 10 and cont < 20:

                xi[i][j] = random.randint(50,80)

            else:

                xi[i][j] = random.randint(100,130)

        elif i == 1:

            if cont < 10 or cont > 20 :

                xi[i][j] = random.randint(80,100)

            else:

                xi[i][j] = random.randint(0,15)

        cont += 1

    cont = 0

U = np.zeros((3,30))

Um1 = np.zeros((3,30))

for i in range(0,30,1):

    aux = random.randint(0,2)

    U[aux][i] = 1

    Um1[aux][i] = 1

print(U)

cont = 0

v1o = [0,0]

v2o = [0,0]

v3o = [0,0]

while(True):

    v11 = 0

    v12 = 0

    v21 = 0

    v22 = 0

    v31 = 0

    v32 = 0

    #Cálculo de los centroides

    #Cluster 1

    numx = 0

    denx = 0

    numy = 0

    deny = 0

    for i in range(0,30,1):

        numx += U[0][i]\*xi[0][i]

        denx += U[0][i]

        numy += U[0][i]\*xi[1][i]

        deny += U[0][i]

    v11 = numx/denx

    v12 = numy/deny

    v1 = [v11,v12]

    #Cluster 2

    numx = 0

    denx = 0

    numy = 0

    deny = 0

    for i in range(0,4,1):

        numx += U[1][i]\*xi[0][i]

        denx += U[1][i]

        numy += U[1][i]\*xi[1][i]

        deny += U[1][i]

    v21 = numx/denx

    v22 = numy/deny

    v2 = [v21,v22]

    #Cluster 3

    numx = 0

    denx = 0

    numy = 0

    deny = 0

    for i in range(0,30,1):

        numx += U[2][i]\*xi[0][i]

        denx += U[2][i]

        numy += U[2][i]\*xi[1][i]

        deny += U[2][i]

    v31 = numx/denx

    v32 = numy/deny

    v3 = [v31,v32]

    if cont == 0:

        v1o = v1

        v2o = v2

        v3o = v3

    #Distancias entre los centroides y los datos

    d1 = np.zeros(30)

    d2 = np.zeros(30)

    d3 = np.zeros(30)

    for i in range(0,30,1):

        #Cluster 1

        d1[i] = math.sqrt((xi[0][i]-v1[0])\*\*2 + (xi[1][i]-v1[1])\*\*2)

        #Cluster 2

        d2[i] = math.sqrt((xi[0][i]-v2[0])\*\*2 + (xi[1][i]-v2[1])\*\*2)

        #Cluster 3

        d3[i] = math.sqrt((xi[0][i]-v3[0])\*\*2 + (xi[1][i]-v3[1])\*\*2)

    #Actualización de U

    for i in range(0,30,1):

        aux = [d1[i],d2[i],d3[i]]

        valmin = np.min(aux)

        for j in range(0,3,1):

            if j == aux.index(valmin):

                Um1[j][i] = 1

            else:

                Um1[j][i] = 0

    cont += 1

    if np.array\_equal(U,Um1):

        break

    U = Um1

#Graficación

plt.figure(1)

plt.plot(xi[0],xi[1],"o",label = "Muestras a agrupar")

plt.plot(v1o[0],v1o[1],"o",label = "Centroide C1 original")

plt.plot(v2o[0],v2o[1],"o",label = "Centroide C2 original")

plt.plot(v3o[0],v3o[1],"o",label = "Centroide C3 original")

plt.plot(v1[0],v1[1],"o",label = "Centroide C1 final")

plt.plot(v2[0],v2[1],"o",label = "Centroide C2 final")

plt.plot(v3[0],v3[1],"o",label = "Centroide C3 final")

plt.legend()