內容

[老師給的要求 1](#_Toc162812235)

[程式前的需求分析 1](#_Toc162812236)

[整體隨機點趨勢以及不同 K 值的差異 2](#_Toc162812237)

[Initial random point Set k=10/30/50 3](#_Toc162812238)

[Final random point Set k=10/30/50 3](#_Toc162812239)

[當k=5，原始隨機點圖 3](#_Toc162812240)

[當k=5，迭代後隨機點圖 4](#_Toc162812241)

[第10/20/30/40/50/60/70/80/90/100次迭代 5](#_Toc162812242)

[迭代過程GIF圖 9](#_Toc162812243)

[我的程式碼 10](#_Toc162812244)

老師給的要求

Please generate 100 particles (2D points) randomly on a 2D plane. The locations of all particles are to be updated iteratively. In each iteration, every particle, say , is updated according to the following rules.

1. Find the K nearest neighbors for particle , say, .
2. Compute the centroid of the K nearest neighbors of particle by

.

1. Move particle towards the centroid of its neighbors slightly by

=.

where is a small positive number.

Please write a program (e.g. Java/python/JS/Typescript/C++) to simulate the movements of the particles, and use EXCEL or screen shot to display the results for a few iterations (e.g. iteration = 10, 20, …, 100). Additionally, please discuss the overall trend of the movements and the differences for different values of K.

程式前的需求分析

First of all let’s Clarity the procedure and the calculation of the code.

1.生成隨機的點：使用 numpy 的 random.rand 函數生成包含 100 個隨機點的二維座標。並給定每個點座標序號，將每個點原始的座標位置(x,y)記錄下來。

2.使用 sklearn 中的 NearestNeighbors 來找到每個點的最近鄰居。(預設為k=5)

3.定義更新函數 update：此函數在每一動畫中被使用，用以計算每個點的最近(K個)鄰居的平均位置，並將點的座標位置進行相應的更新。

4.建立動畫：使用 matplotlib 中的 FuncAnimation 創建動畫，將更新函數應用到動畫中，更新圖形，需在動畫中央顯示Iteration的次數，並且隨機點請用不同顏色標示。

5.顯示每個點移動的過程並至製成動畫播放。

6.設定的迴圈要在iteration到100後就結束break，並且列印出每十個迭代中每個點的 序號 / 原始座標 / 移動距離 / 最後座標

**#我設定要將原始圖存起來(作為對比)動畫才會繼續跑迭代**

整體隨機點趨勢以及不同 K 值的差異

通過迭代，將這些隨機產生100個的二維點座標朝向其對應最近的 K 個鄰居點的質心移動。在每個迭代中，隨機點會根據其鄰居的質心向其移動一小步。

.

=.

這邊的  **可以是極小的正值，我們這邊預設為0.01**。

最後迭代完成後會產生隨機點集的結果，整體趨勢會慢慢收斂到一個中心點，若是不設定停止收斂，而是一直讓迭代無限的跑下去，所有隨機點最後都會聚集在一起。

而Ｋ值表示每個隨機點的要找的鄰居數量，在原始的程式中我們將ｋ設定為5，**若Ｋ值越大**，會發現隨機產生的二維座標點會受到更多鄰居點的影響，可能會導致**隨機點更平滑的移動或集中，最終產生的集群數量會較少**。

相反，若**設定較小的Ｋ值**，每次移動會限制鄰居點的範圍，導致產生的集群數量更多、**結果會較發散**。

下面就是不同Ｋ值產生的結果，同樣隨機產生100個點，將Ｋ值設定為10、30、50的結果圖如下，我們可以看到**當K值設定越大時，最後產生的集群就越集中**。

一張含有 圖表, 文字, 行, 螢幕擷取畫面 的圖片

自動產生的描述Initial random point Set k=10/30/50

一張含有 圖表, 螢幕擷取畫面, 行, 繪圖 的圖片

自動產生的描述一張含有 圖表, 螢幕擷取畫面, 文字, 行 的圖片

自動產生的描述

一張含有 文字, 圖表, 螢幕擷取畫面, 行 的圖片

自動產生的描述Final random point Set k=10/30/50

一張含有 文字, 圖表, 螢幕擷取畫面, 繪圖 的圖片

自動產生的描述一張含有 圖表, 行, 螢幕擷取畫面, 繪圖 的圖片

自動產生的描述

當k=5，原始隨機點圖

一張含有 文字, 圖表, 螢幕擷取畫面, 行 的圖片

自動產生的描述

當k=5，迭代後隨機點圖

一張含有 文字, 圖表, 螢幕擷取畫面, 行 的圖片

自動產生的描述

我設定將每10次迭代的原始座標和最後座標列印出來，所以會有10、20…90、100次迭代的圖

第10/20/30/40/50/60/70/80/90/100次迭代

一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述

迭代過程GIF圖

因為是GIF所以會不停的重播。

這是符合老師說的k=5時的每100個隨機點的迭代變化。

一張含有 文字, 圖表, 行, 螢幕擷取畫面 的圖片

自動產生的描述

我的程式碼

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from sklearn.neighbors import NearestNeighbors

# 隨機生成100個二維點

num\_points = 100

random\_points = np.random.rand(num\_points, 2) \* 1000

random\_points\_index = np.arange(num\_points)

original\_coordinates = random\_points.copy()

# 顯示迭代前的隨機圖，方便後續做比較

plt.figure(figsize=(8, 6))

plt.scatter(random\_points[:, 0], random\_points[:, 1], c=random\_points\_index, cmap='viridis')

plt.title('Initial Random Points M11209202 saving the initial random points image before the iteration starts:')

plt.xlabel('X')

plt.ylabel('Y')

plt.grid(True)

plt.show()

# 初始化 NearestNeighbors 物件

k = 5

nn = NearestNeighbors(n\_neighbors=k)

nn.fit(random\_points)

# 初始化動畫

fig, ax = plt.subplots()

scatter = ax.scatter(random\_points[:, 0], random\_points[:, 1], c=random\_points\_index, cmap='viridis')

ax.set\_title('Updated Random Points M11209202')

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

text = ax.text(0.5, 0.5, '', horizontalalignment='center', verticalalignment='center', transform=ax.transAxes)

ax.grid(True)

# 更新函數

mean\_movement\_vectors = np.zeros\_like(random\_points)

movement\_history = []

num\_iterations = 100

def update(frame):

    global random\_points

    global movement\_history

    if frame >= num\_iterations:

        print("Iteration reached 100. Stopping animation.")

        text.set\_text("Iteration =100 ,Done! ")

        ani.event\_source.stop()  # 停止動畫

        return

    distances, indices = nn.kneighbors(random\_points)

    for i in range(len(random\_points)):

        neighbor\_indices = indices[i]

        neighbors = random\_points[neighbor\_indices]

        # 計算鄰居的質心

        mean\_movement\_vector = np.mean(neighbors, axis=0)

        mean\_movement\_vectors[i] = mean\_movement\_vector

    movement\_step\_size = 0.01

    # 將粒子向鄰居質心移動

    # x' = x + ϵ(μ - x)

    random\_points += movement\_step\_size \* (mean\_movement\_vectors - random\_points)

    movement\_history.append(random\_points.copy())

    scatter.set\_offsets(random\_points)

    text.set\_text(f"Iteration: {frame+1}")

    if (frame + 1) % 10 == 0:  # 每10個迭代列印一次

        print(f"Iteration: {frame+1}")

        print("序號\t原始座標\t\t移動距離\t\t最後座標")

        for i in range(num\_points):

            distance = np.linalg.norm(original\_coordinates[i] - random\_points[i])

            print(f"{i+1}\t{original\_coordinates[i]}\t{distance}\t{random\_points[i]}")

        print()

# 創建動畫

ani = FuncAnimation(fig, update, frames=num\_iterations+10, interval=200)

plt.show()