构建带高程的空间权重矩阵，距离阈值为反距离权

步骤：

1、数据准备：

数据集需要包含经纬度、高程以及相关属性。每个观测点应有三维坐标（经度、纬度、高程）。

2、计算三维欧氏距离：

利用三维坐标计算每对点之间的三维欧氏距离：

图示

描述已自动生成

其中*x*i,*y*i是为投影坐标系的经纬值，而*zi*代表高程。

1. **定义反距离权重函数**

**文本

中度可信度描述已自动生成**

1. 应用阈值

设置一个距离阈值 ，对于超过这个阈值的两点间的距离，赋予其权重为0，以限制仅考虑一定范围内的空间依赖性。

1. 标准化权重

对于某些分析，可能需要行标准化权重矩阵，确保每一行权重之和为1，以保持加权邻域内总影响力不变。

1. 创建空间权重矩阵

使用上述计算出的权重填充到一个矩阵中，该矩阵的行索引和列索引对应于数据集中点的编号，形成一个空间权重矩阵。

代码实现：

import numpy as np

from scipy.spatial.distance import pdist, squareform

*# 假设df是一个GeoDataFrame或DataFrame，其中含有三列分别代表经度、纬度和高程的笛卡尔坐标*

df = ... *# 你的数据集，例如：['x', 'y', 'z']三列是笛卡尔坐标*

*# 计算三维欧氏距离*

xyz = df[['x', 'y', 'z']].values

distances\_3d = pdist(xyz, metric='euclidean')

*# 应用反距离权重函数，这里选择p=2*

p = 2

weights = 1 / distances\_3d\*\*p

*# 应用距离阈值，这里假设threshold为一个预设的最大距离*

threshold = ... *# 根据实际情况设定阈值*

weights[distances\_3d > threshold] = 0

*# 将一维向量转换为对称矩阵*

weights\_matrix = squareform(weights)

*# 对权重矩阵进行行标准化（可选）*

row\_sums = weights\_matrix.sum(axis=1)

normalized\_weights\_matrix = weights\_matrix / row\_sums[:, np.newaxis]

*# 此时，normalized\_weights\_matrix即为所求的空间权重矩阵*