实验 KMeans

姓名：李昀哲　学号：20123101

1 题目和数据：

题目：K-means

通过对给定数据进行聚类分析来了解 K-means 算法。对 K-means 算法原理，K-means 算法流程，K-means 算法应用有更深的了解

数据描述:

本数据是随机生成的符合高斯分布的二维样本点。通过查看数据的 shape，观察到其为 80 行 2 列，即 80 组样本，每组样本有两个特征。数据点的分布如图 1 所示

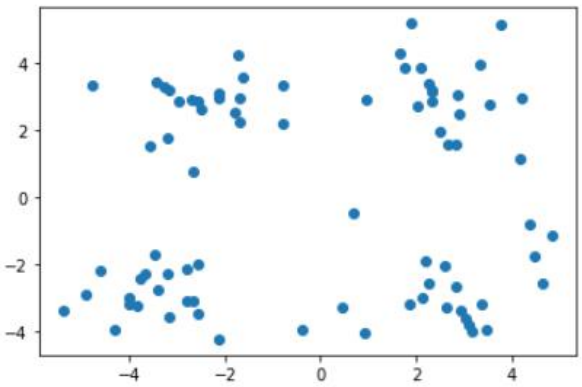

图 1 K-means 实验数据

2 算法：

KMeans 聚类是最基础常用的聚类算法，聚类属于非监督学习。它的基本思想是，通过迭代寻找 K 个簇（Cluster）的一种划分方案，使得聚类结果对应的损失函数最小。其中，损失函数可以定义为各个样本距离所属簇中心点的误差平方和 J.

与分类、序列标注等任务不同，聚类是在事先并不知道任何样本标签的情况下，通过数据之间的内在关系把样本划分为若干类别，使得同类别样本之间的相似度高，不同类别之间的样本相似度低（即增大类内聚，减少类间距）。

KMeans 最核心的部分就是先固定中心点，调整每个样本所属的类别来减少 J；再固定每个样本的类别，调整中心点继续减小 J。两个过程交替循环，J 单调递减直到最（极）小值，中心点和样本划分的类别同时收敛。

其**优点**是算法思想简单，便于实现；但**缺点**也较为明显：K 值难以确定，复杂程度与样本呈线性关系，很难发现任意形状的 cluster，如图 2 所示。因此往往也会通过 DBSCAN 算法处理。

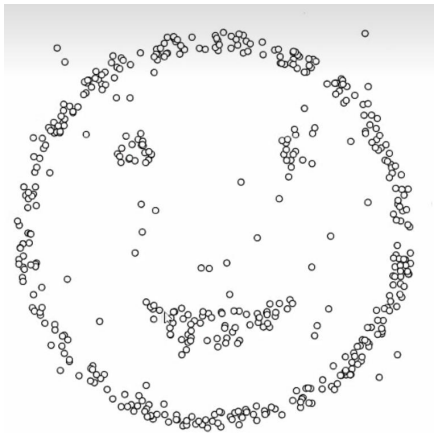

图 2 K-means 难以处理的形状

3 代码及结果

首先加载数据集，得到 DataMat 并进行展示；

```
import pandas as pd
def loadDataSet(fileName):
    """
    函数说明:加载数据集
    parameters:
        fileName - 文件名
    return:
        DataMat - 数据集
    """
    data = pd.read_table("./KmeansData.txt", sep="\t", header=None)
    data = np.array(data)
    return data
DataMat = loadDataSet("./KmeansData.txt")
plt.scatter(DataMat[:, 0], DataMat[:, 1])
```

再根据算法流程，找到初始的随机簇中心；

```
import numpy as np
def randCent(DataMat, k):
    """
    函数说明:从当前样本点中随机选取k个初始簇中心
    parameters:
         DataMat - 数据集
         k - 聚类后簇的数量
    return:
        centroids - 簇中心列表
    """
    num_examples = DataMat.shape[0]
    random_index = np.random.permutation(num_examples)
    centroids = DataMat[random_index[:k], :]
    return centroids
init_centroid = randCent(DataMat, 3)
print(init_centroid)
```

得到初始簇中心后需要计算每个点到每个中心的距离，以进行迭代，因此需要用到计算两个点之间的欧几里得距离；

```
def distEclud(vecA, vecB):
    """
    函数说明:利用欧式距离来计算每个样本点之间的距离
    parameters:
        vecA - A样本的特征向量（本数据中指它的二维坐标值）
        vecB - B样本的特征向量（本数据中指它的二维坐标值）
    return:
        Dist - 样本点间的欧式距离
    """
    dist = np.sqrt(np.sum(vecA - vecB)**2)
    return  dist
```

所有算法工具函数准备齐全后，进行整体的迭代，其中需要注意的是，在得到最近簇中心的过程中，要记录的并非到簇中心的对短距离，而是记录这个簇中心的索引值，此处在测试是发生错误，多次调试后才发现。

另外，对于 numpy.array 而言，用一个数去 == numpy.array，会得到一个该索引处是否存在的 bool 数组，可以用于作为另一个 numpy.array 的索引。

```
def kMeans(dataSet, k, distMeas=distEclud, createCent=randCent):
    """
    函数说明：K-均值算法
    parameters:
         dataSet -数据集
         k -簇个数
         distMeas -距离计算函数
         createCent -创建初始质心函数
    return:
         centroids -质心列表
         clusterAssment -簇分配结果矩阵
    """
    centroids = createCent(dataSet, k)
    num_samples = dataSet.shape[0]
    num_features = dataSet.shape[1]
    max_iterations = 20

    for _ in range(max_iterations):
        # 这一步很关键，要得到的是索引，也就是哪个点最近，而不想得到距离的值
        closest_dist_id = np.zeros((num_samples, 1))
        for sample_id in range(num_samples):
            distance_to_centroid = np.zeros((k, 1))
            for centroid_id in range(k):
                distance_to_centroid[centroid_id] = distEclud(dataSet[sample_id], centroids[centroid_id])
            # 注意，这里得到的是索引，所以要argmin!!!!!!不是min!!!!!!
            closest_dist_id[sample_id] = np.argmin(distance_to_centroid)

        clusterAssment = closest_dist_id
        # 更新中心点
        for centroid_id in range(k):
            closest_id = centroid_id == closest_dist_id
#             print("Before update :", centroids[centroid_id])
#             print("Updated       :", np.mean(dataSet[closest_id.flatten()], axis=0))
            centroids[centroid_id] = np.mean(dataSet[closest_id.flatten()], axis=0)
#         print("Results :\n", centroids)

    return centroids, clusterAssment
```

最后展示结果

```
def drawDataSet(DataMat,centroids,clusterAssment,k):
    """
    函数说明：将聚类结果可视化
    parameters:
         centList -质心列表
         clusterAssment -簇列表
         dataMat -数据集
         k -簇个数
    return:
         A picture
"""
    plt.scatter(centroids[:, 0], centroids[:, 1], marker='x')

    points_0 = DataMat[(clusterAssment == 0).flatten()]
    points_1 = DataMat[(clusterAssment == 1).flatten()]
    points_2 = DataMat[(clusterAssment == 2).flatten()]
    points_3 = DataMat[(clusterAssment == 3).flatten()]
    plt.scatter(points_0[:, 0], points_0[:, 1])
    plt.scatter(points_1[:, 0], points_1[:, 1])
    plt.scatter(points_2[:, 0], points_2[:, 1])
    plt.scatter(points_3[:, 0], points_3[:, 1])
    plt.show()
```

主函数如下：

```
if __name__ =='__main__':
    """
    按照Kmeans原理来调用上述函数
    """
    centroids, clusterAssment = kMeans(DataMat, 4, distEclud, randCent)
    drawDataSet(DataMat, centroids, clusterAssment, 4)
```

由初始数据点可以观察到，大概是会分为 4 簇，因此指定 k 为 4，但对于迭代得到的结果，并不如预先所想那样分配，多组实验结果都以斜向划分，猜测可能与初始点的选择有关，且原数据并非十分规整，导致从人的角度观察，也并不能完美划分各个簇，因此认为这些结果是合理的。

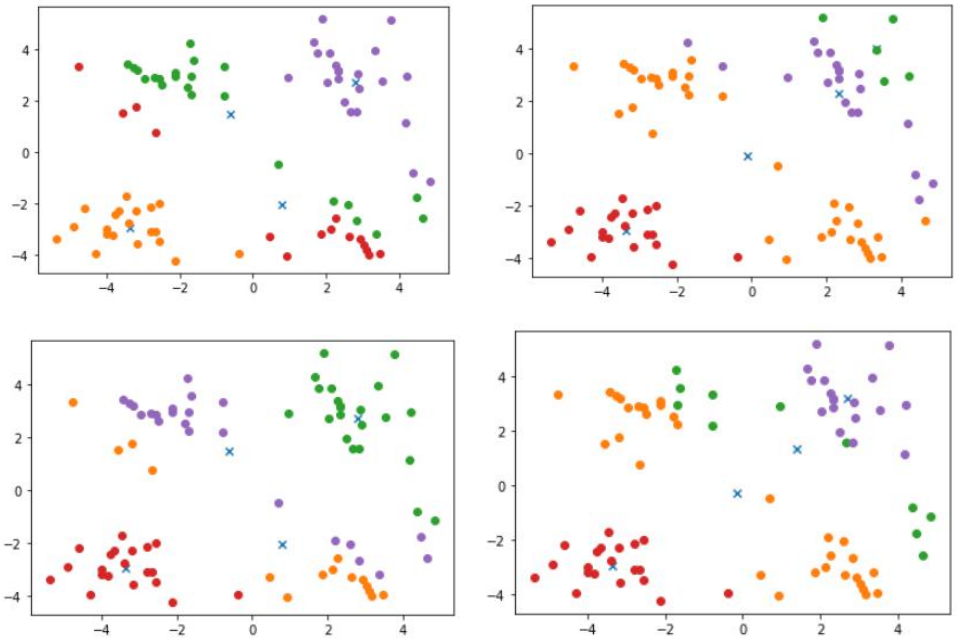

图 3 实验结果