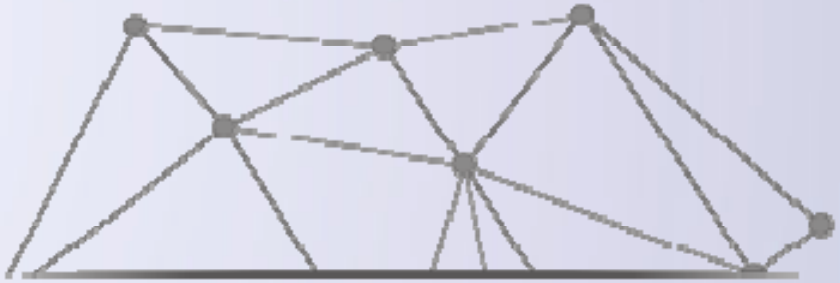
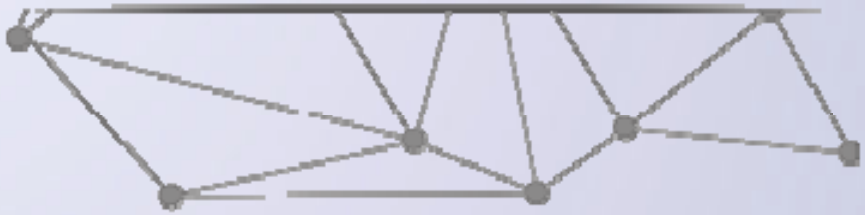


kmeans应用实例

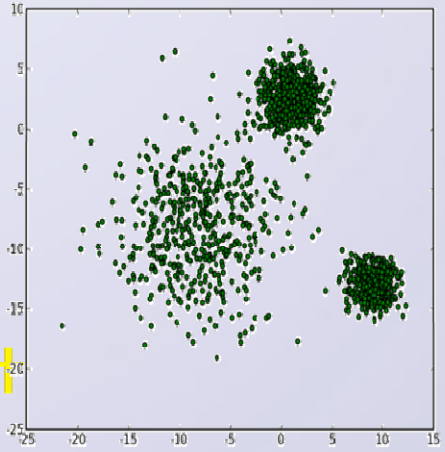


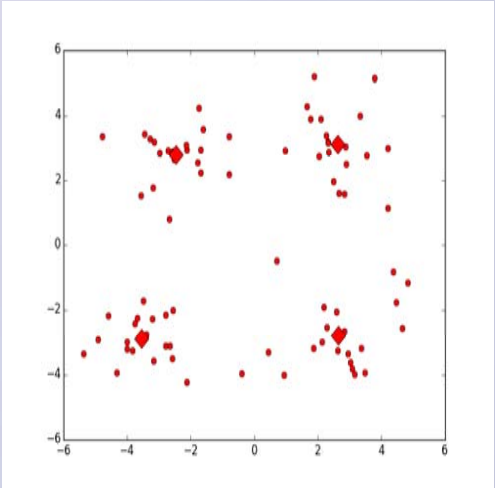
礼 欣

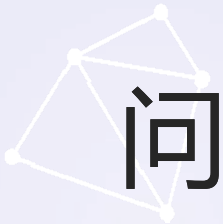
北京理工大学

问题定义

- 聚类问题是数据挖掘的基本问题，它的本质是将n个数据对象划分为 k个聚类，以便使得所获得的聚类满足以下条件：同一聚类中的数据对象相似度较高；不同聚类中的对象相似度较小。
- 相似度可以根据问题的性质进行数学定义。

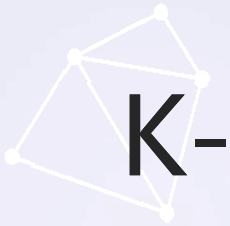






问题定义

- K-means算法就是解决这类问题的经典聚类算法
- 它的基本思想是以空间中k个点为中心，进行聚类，对最靠近他们的对象归类。通过迭代的方法，逐次更新各聚类中心的值，直至得到最好的聚类结果。其IPO描述如下：
  - 输入：N个数据
  - 操作：聚类算法
  - 输出：图形化显示聚类结果

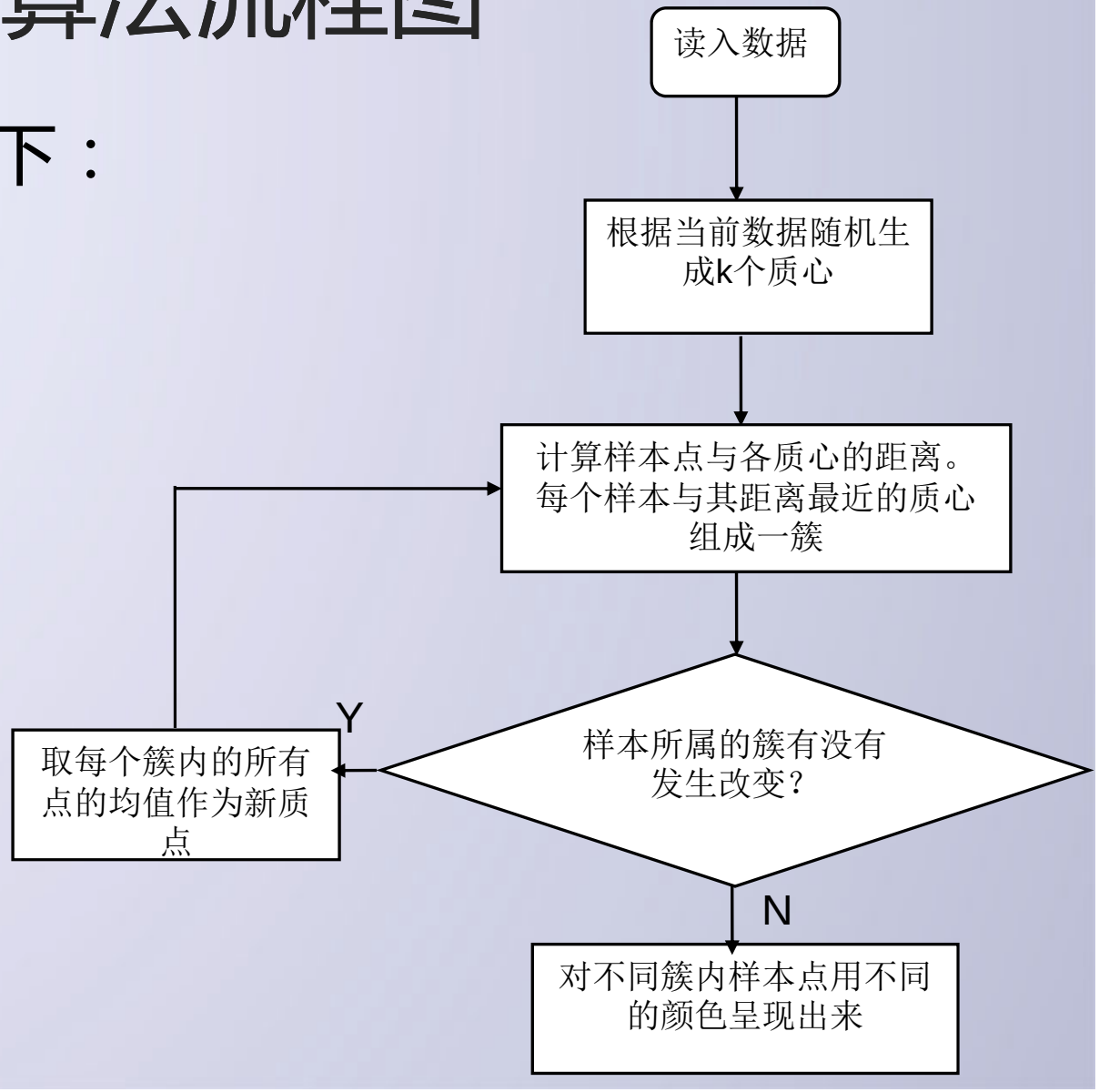


K-means算法步骤

- 算法的基本步骤为
  - 从 n个数据对象任意选择 k 个对象作为初始聚类中心；并设定最大迭代次数
  - 计算每个对象与k个中心点的距离，并根据最小距离对相应对象进行划分，即，把对象划分到与他们最近的中心所代表的类别中去；
  - 对于每一个中心点，遍历他们所包含的对象，计算这些对象所有维度的和的均值，获得新的中心点；
  - 如果聚类中心与上次迭代之前相比，有所改变，或者，算法迭代次数小于给定的最大迭代次数，则继续执行第2、3两步，否则，程序结束返回聚类结果。

K-means算法流程图

- 其算法流程图如下：

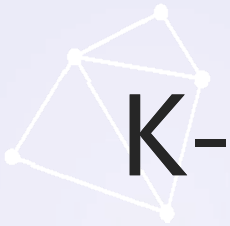


python 语言程序设计

K-means算法运行过程

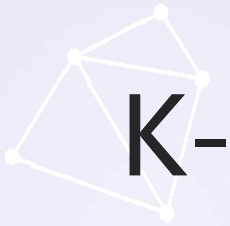
- 程序代码如下：
- 程序的控制部分：
  - 首先从文件读入数据，并将其存储在Numpy的数组对象中，
  - 指定聚类个数，与，最大迭代次数，
  - 调用kmeans聚类函数，得到聚类结果
  - 将聚类结果以图的形式展示出来。

python 语言程序设计



K-means算法运行过程

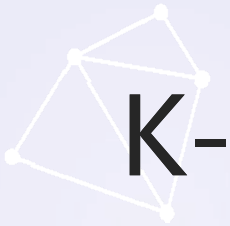
```
def main():
    ##step 1: load dataSet
    print ("step 1: loading data...")
    dataSet = []
    dataSetFile = open('./testSet.txt');
    for line in dataSetFile:
        lineArr = line.strip().split('\t')
        dataSet.append([float(lineArr[0]), float(lineArr[1])])

    #step 2: clustering...
    print ("step 2: clustering...")
    dataSet = np.mat(dataSet)

    k = 4
    centers_result, clusterAssignment_result = kmeans(dataSet,k, 100)

    #step 3: show the result
    print ("step3: showing the result...")
    showCluster(dataSet, k, centers_result, clusterAssignment_result)

main()
```



K-means算法运行过程

- 子函数定义
  - Initialize center函数通过使用numpy库的zeros函数和random.uniform函数，随机选取了k个数据做聚类中心， 并将结果存放在Numpy的Array对象centers中

```
def initCenters(dataSet, k):
    numSamples, dim = dataSet.shape
    centers = np.zeros((k, dim))
    for i in range(k):
        index = int(np.random.uniform(0, numSamples)) #random get k centers
        centers[i, :] = dataSet[index, :]
    print(centers)
    return centers
```



K-means算法运行过程

- Dist2Centers这个函数用来计算一个数据点到所有聚类中心的距离，将其存放在dis2cents中返回

```
def Dist2Centers(sample, centers):
    k = centers.shape[0]
    dis2cents = np.zeros(k)
    for i in range(k):
        dis2cents[i]=np.sqrt(np.sum(np.power(sample - centers[i,:], 2)))
    return dis2cents
```

K-means算法运行过程

- kmeans函数代码如下：
- 这部分代码完成了kmeans算法中为数据点决定所属类别 以及迭代更新类中心点的主要功能。
- 请注意numpy库的返回最小值索引的argmin函数以及计算平均值的mean函数的使用方法

python 语言程序设计

```
def kmeans(dataSet, k, iterNum):
    numSamples = dataSet.shape[0]
    iterCount = 0

    # clusterAssignment stores which cluster this sample belongs to,
    clusterAssignment = np.zeros(numSamples)
    clusterChanged = True

    ## step 1: initialize centers
    centers = initCenters(dataSet, k)
    while clusterChanged and iterCount < iterNum:
        clusterChanged = False
        iterCount = iterCount + 1
        ## for each sample
        for i in range(numSamples):

            dis2cent=Dist2Centers(dataSet[i,:], centers)
            minIndex = np.argmin(dis2cent)
            ## step 3: update its belonged cluster
            if clusterAssignment[i] != minIndex:
                clusterChanged = True
                clusterAssignment[i] = minIndex

        ## step 4: update centers
        for j in range(k):
            pointsInCluster = dataSet[np.nonzero(clusterAssignment[:] == j)[0]]
            centers[j, :] = np.mean(pointsInCluster, axis = 0)
    print ('Congratulations, Cluster Achieved!')
    return centers, clusterAssignment
```

K-means算法运行过程

- showcluster函数中，利用matplotlib库的plot函数将不同类别数据以不同颜色展现出来
- 程序代码如下：

```
def showCluster(dataSet, k, centers, clusterAssignment):
    numSamples, dim = dataSet.shape

    mark = ['or', 'ob','og','om']

    # draw all samples
    for i in range(numSamples):
        markIndex = int(clusterAssignment[i])
        plt.plot(dataSet[i, 0], dataSet[i, 1], mark[markIndex])

    mark = ['Dr', 'Db', 'Dg','Dm']
    # draw the centroids
    for i in range(k):
        plt.plot(centers[i, 0], centers[i, 1], mark[i], markersize = 17)

    plt.show()
```

实验结果

- 运行程序,
- 下面依次是，将数据，聚为两类，三类，四类的程序结果图。大家也可以通过调整迭代次数，观察生成簇的变化