# 《数据挖掘导论》实验 4: k-means实验

# 一、实验目的

（1）了解k-means算法思想

（2）掌握k-means算法的应用

# 二、实验环境

（1）Anaconda2 开发环境

（2）IDE是ipython notebook

（3）使用的库有numpy，pandas，matplotlib

# 三、实验内容

## 1. k-means

(1) k-means基本思想

k-means算法的基本思想是初始随机给定K个簇中心，按照最邻近原则把待分类样本点分到各个簇。然后按平均法重新计算各个簇的质心，从而确定新的簇心。一直迭代，直到簇心的移动距离小于某个给定的值。

(2) k-means算法伪代码

|  |
| --- |
| 创建k个点作为初始聚类中心（经常是随机选择）  当任意一个点的簇分配结果发生改变时  对数据集中的每个数据点  对每个聚类中心  计算聚类中心与数据点之间的距离  将数据点分配到距其最近的簇  对每一个簇，计算簇中所有点的均值并将均值作为质心。 |

（3） 聚类步骤图示：

下图展示了对n个样本点进行K-means聚类的效果，这里k取2：

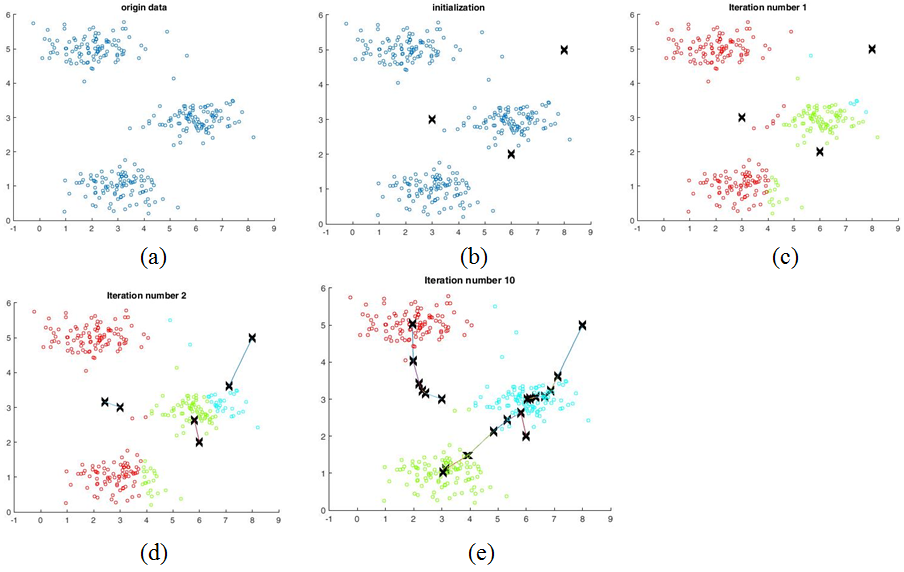
(a)未聚类的初始点集。

(b)随机选取两个点作为聚类中心。

(c)计算每个点到聚类中心的距离，并聚类到离该点最近的聚类中去。

(d)计算每个聚类中所有点的坐标平均值，并将这个平均值作为新的聚类中心。执行（c）将每个点聚类到离该点最近的聚类中去。

(e)重复(d)直至聚类中心不再进行大范围移动或者聚类次数达到要求为止。



## 2. 二分k-means

(1) 二分k-means基本思想

二分k-means算法的基本思想是将所有样本点作为一个簇，然后将该簇一分为二。之后选择其中一个簇继续进行划分，选择哪一个簇进行划分取决于对其划分是否可以最大程度降低SSE（Sum of Squared Error, 误差平方和）的值。SSE的计算如公式（1）所示，其中k是聚类中心个数，x是样本点，表示属于中心表示的类别的样本，表示样本到中心的欧几里得距离的平方。

（1）

(2) 二分k-means算法伪代码

|  |
| --- |
| 将所有点看成一个簇  当簇数据小于k时  对于每个一个簇  计算总的误差  在给定的簇上面进行k-means聚类（k=2）  计算该簇一分为二之后的总误差  选择使得误差最小的那个簇进行划分操作 |

## 3.函数说明

**(1)lodaDataSet函数**

将数据文件解析成一个二维列表。

loadDataSet(fileName)

* 参数
* fileName：文件路径。数据以“\t”分隔。
* 返回值
* dataMat：list类型，一个浮点类型的二维的列表

**(2)distEclud函数**

计算两个点之间的欧几里得距离。

distEclud(vecA, vecB)

* 参数
* vecA：matrix类型或者array类型
* vecB：matrix类型或者array类型
* 返回值
* 一个浮点数，欧几里得距离

**(3)randCent函数**

随机初始化k个聚类中心。

randCent(dataSet, k)

* 参数
* dataSet：输入数据，二维的列表。
* k：初始化聚类中心个数。
* 返回值
* dataMat：matrix类型，一个二维矩阵，每一行表示一个中心点。

**(4)kMeans函数**

k-means算法聚类函数

kMeans(dataSet, k, distMeas=distEclud, createCent=randCent)

* 参数
* dataSet：输入数据，二维的列表。
* k：聚类中心个数。
* distMeas：设置计算距离的函数，默认为distEclud函数，即使用欧几里得距离。
* createCent：聚类中心初始化方式，默认为randCent函数，即随机初始化聚类中心。
* 返回值
* centroids：matrix类型，每行是一个聚类中心
* clusterAssment：matrix类型，每行第一个元素是相应样本点的类标，第二个元素是该样本点到最近的聚类中心的距离。

**(5)biKmeans函数**

二分k-means算法聚类函数

biKmeans(dataSet, k, distMeas=distEclud)

* 参数
* dataSet：输入数据，二维的列表。
* k：聚类中心个数。
* distMeas：设置计算距离的函数，默认为distEclud函数，即使用欧几里得距离。
* 返回值
* centroids：matrix类型，每行是一个聚类中心
* clusterAssment：matrix类型，每行第一个元素是相应样本点的类标，第二个元素是该样本点到最近的聚类中心的距离。

**(6)distSLC函数**

计算地球表面两点之间的距离。

distSLC(vecA, vecB)

* 参数
* vecA：matrix类型或者array类型
* vecB：matrix类型或者array类型
* 返回值
* 一个浮点数，球面两点之间的距离

**(7)clusterClubs函数**

对地理坐标进行聚类和可视化的一个例子。

clusterClubs(numClust=5)

* 参数
* numClust：聚类中心的个数
* 返回值
* None

## 4.案例一，简单的例子

### (1)读取实验数据

testSet.txt存储的数据以‘\t’分隔，每一行数据有个数，共同表示样本的坐标。

使用如下函数读取数据集

|  |
| --- |
| import kMeans  datMat = np.mat(kMeans.loadDataSet('./testSet.txt')) |

使用datMat[:3,:]查看前3条数据如下表所示：

|  |  |
| --- | --- |
| 1.65898 | 4.285136 |
| -3.45368 | 3.424321 |
| 4.838138 | 1.151539 |

### (2)聚类

使用kMeans对测试数据进行聚类

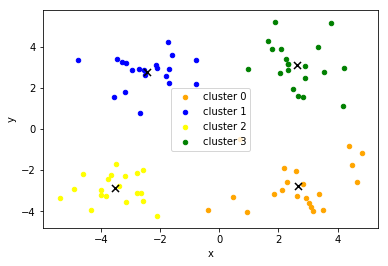
|  |
| --- |
| myCentroids, clustAssing = kMeans.kMeans(datMat,4) |

### (3)可视化

使用如下代码对聚类结果进行可视化

|  |
| --- |
| # 将聚类中心数据转化为DataFrame类型，字段名分别取为x和y  center = pd.DataFrame(myCentroids, columns=['x', 'y'])  # 将原始的坐标数据转化为DataFrame类型，字段名分别为x和y  frame = pd.DataFrame(datMat, columns=['x', 'y'])  # 将聚类的结果添加到fame变量中  frame['label'] = clustAssing.A[:, 0].astype(pd.np.int)  # 根据原始数据的坐标绘制散点图，给不同类标的数据绘制不同的颜色  ax = None  color = ['orange','blue', 'yellow', 'green', 'red']  for i, c in enumerate(sorted(frame['label'].unique())):  ax = frame[frame['label'] == c].plot(x='x', y='y', kind='scatter', label='cluster {0}'.format(c), c=color[i%len(color)], ax=ax)  # 绘制聚类中心，用黑色的x表示  center.plot(x='x', y='y', kind='scatter', marker='x', s=50, c='black', ax=ax) |

结果如下图所示：



## 5.案例二，对地图上的点进行聚类

### (1)读取俱乐部纬度信息

places.txt存储的数据以‘\t’分隔，没有存储字段名称，各个字段的含义如下表所示。

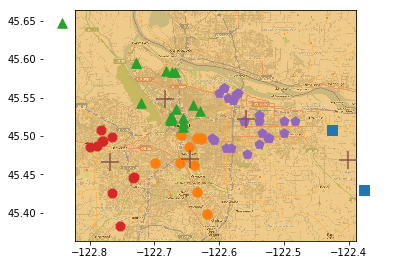
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| club name | street adress | city | latitude | longitude |
| Dolphin II | 10860 SW Beaverton-Hillsdale Hwy | Beaverton, OR | 45.486502 | -122.788346 |

### (2)聚类及可视化

clusterClubs方法封装了读取places.txt文件存储的俱乐部信息，使用biKmeans进行聚类并对结果绘图的函数。

|  |
| --- |
| import kMeans  kMeans.clusterClubs(numClust=5) |

结果如下图所示：



### (3)clusterClubs函数代码分析

对地理坐标进行聚类和可视化的一个例子。

clusterClubs(numClust=5)

* 参数
* numClust：聚类中心的个数
* 返回值

None

clusterClubs函数源代码如下表所示，其中第3行至第5行是逐行读取places.txt文件内容，然后以‘\t’为分隔符解析数据，最后读取经纬度数值字符串转换为浮点数，并存放至datList中，每行表示一个经纬度坐标。第7行是对经纬度坐标进行使用二分k-means算法进行聚类。第8行至第15行读取一个图片并显示到axes对象中。第16行新建一个axes对象，frameon=False设置后只显示当前绘制的内容，这样就能将散点绘制到一个背景图上了。第17到20行以不同颜色和形状绘制不同的簇。第21行绘制簇中心点。第22行显示图形。

|  |
| --- |
| 1. def clusterClubs(numClust=5):  2. datList = []  3. for line in open('places.txt').readlines():  4. lineArr = line.split('\t')  5. datList.append([float(lineArr[4]), float(lineArr[3])])  6. datMat = mat(datList)  7. myCentroids, clustAssing = biKmeans(datMat, numClust, distMeas=distSLC)  8. fig = plt.figure()  9. rect=[0.1,0.1,0.8,0.8]  10. scatterMarkers=['s', 'o', '^', '8', 'p', \  11. 'd', 'v', 'h', '>', '<']  12. axprops = dict(xticks=[], yticks=[])  13. ax0=fig.add\_axes(rect, label='ax0', \*\*axprops)  14. imgP = plt.imread('Portland.png')  15. ax0.imshow(imgP)  16. ax1=fig.add\_axes(rect, label='ax1', frameon=False)  17. for i in range(numClust):  18. ptsInCurrCluster = datMat[nonzero(clustAssing[:,0].A==i)[0],:]  19. markerStyle = scatterMarkers[i % len(scatterMarkers)]  20. ax1.scatter(ptsInCurrCluster[:,0].flatten().A[0], ptsInCurrCluster[:,1].flatten().A[0], marker=markerStyle, s=90)  21. ax1.scatter(myCentroids[:,0].flatten().A[0], myCentroids[:,1].flatten().A[0], marker='+', s=300)  22. plt.show() |

## 6.实践任务

**（1）Iris鸢尾花朵聚类**

1. **数据来源:** **https://archive.ics.uci.edu/ml/datasets/Iris**

原始数据整理后的格式如下表所示，第一行是字段名字，花萼的长，花萼的宽，花瓣的长，花瓣的宽和花的类别，其中花的类别有'setosa', 'versicolor', 'virginica'。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | class |
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |

**b)聚类评价指标**

**评价指标：** 兰德指数（Rand Index, 简称RI）

兰德指数(Rand index)需要给定实际类别信息C，假设K是聚类结果，a表示在C与K中都是同类别的元素对数，b表示在C与K中都是不同类别的元素对数，则兰德指数为：

其中m是总的样本数，RI取值范围为[0,1]，值越大意味着聚类结果与真实情况越吻合。

**c)实验要求:**

1) 分别使用k-means算法和二分k-means算法对iris数据集进行聚类，聚类个数分别取多个参数。

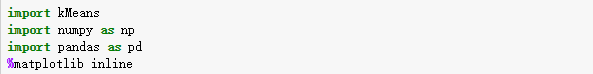
2)使用b)评价指标对聚类结果进行评估。

3)绘制如下形式的表格：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 聚类个数a | 聚类个数b | 聚类个数c |
| k-means |  |  |  |
| 二分k-means |  |  |  |

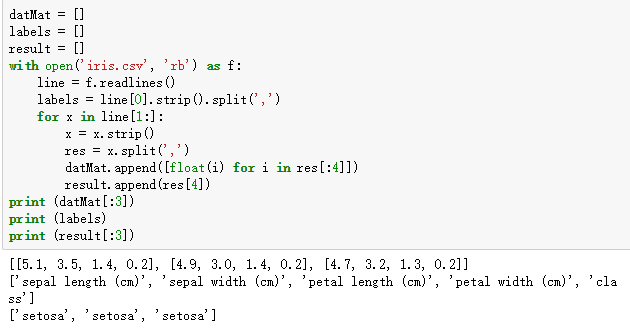
1. 利用k-means计算

① 导入需要使用的库



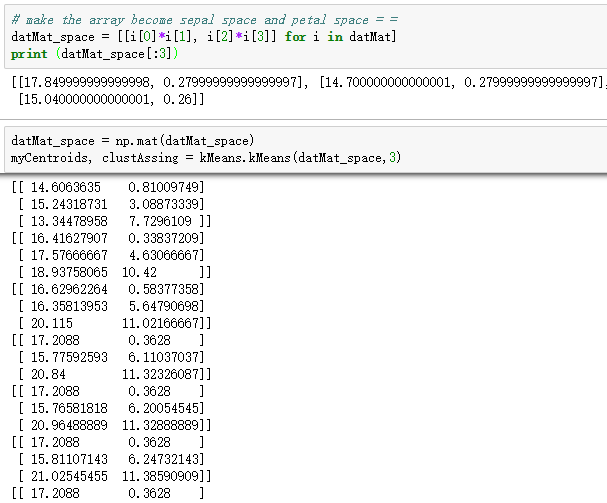
此处使用实验提供的kMeans算法，故而导入该包，并在后面会使用numpy的数组以及矩阵对象处理我们的数据，最后再引用pandas作为我们的绘图工具。

② 数据的导入以及描述



这里使用的是iris鸢尾花数据，有150个数据项，其中包括五个数据，speal length、speal width、petal length、petal width以及他们对应的类。这里我们分别将数据，类标分别存入datMat和result列表中。Labels则存储我们每一项数据对应的意义。

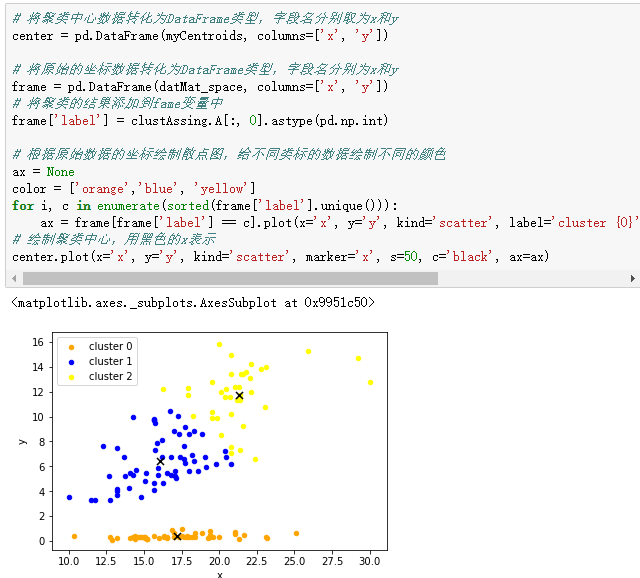
③ 数据的处理以及算法的调用



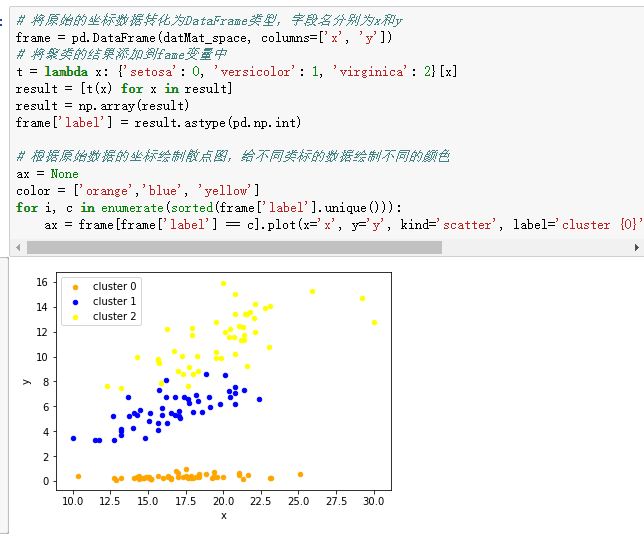
由于kmeans算法是对二维的点进行处理，故而我们需要将四个数据聚合成为两个数据，此时我们将其对应的长和宽相乘，得到我们的新数据集，前三个的结果如上图第一段代码所示。

然后我们将我们的结果转换成矩阵，再将其传入kMeans算法中进行运算，其中划分的类数为3个，其输出中心点移动的各种情况。

④ 结果的可视化演示



将我们的kMeans算法结果展示在图中就是上图的结果，该聚类被分为三个类，具体的对比在下面。



上图是原始数据的结果图示，可以见到的是，在1和2的交界处，也就i时versicolor和virginica类的交汇处，他们聚类的准度略有不足。

⑤ 结果评价



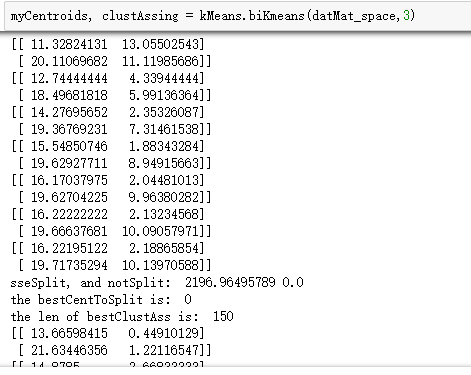
这里的三段代码分别是三种正确个数的统计、每一类数目的统计以及兰德指数的计算。其中兰德指数的计算直接使用的sklean库中的计算函数，可以得到结果为0.67左右。

1. 利用二分k-means计算

① 相关包的导入（此处与上一个的导入重复，不再讨论）

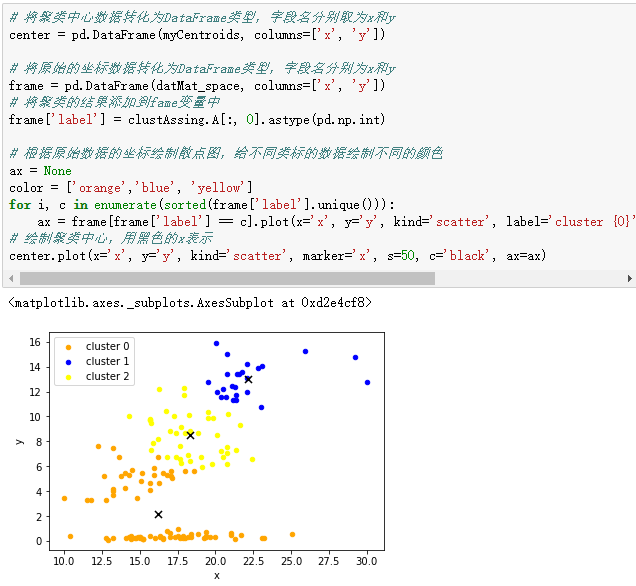
② 数据的导入以及描述（此处与上一个的导入重复，不再讨论）

③ 数据的处理以及算法的调用

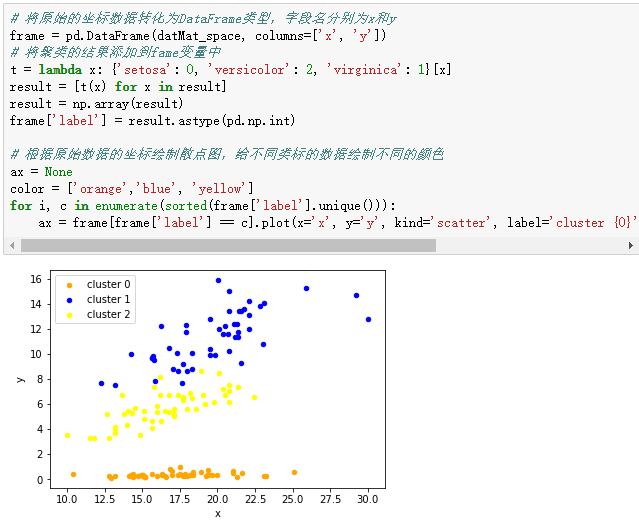


数据的处理与上题类似，不做赘述，此处直接调用了二分kMeans的算法，结果如上图所示。

④ 结果的可视化演示

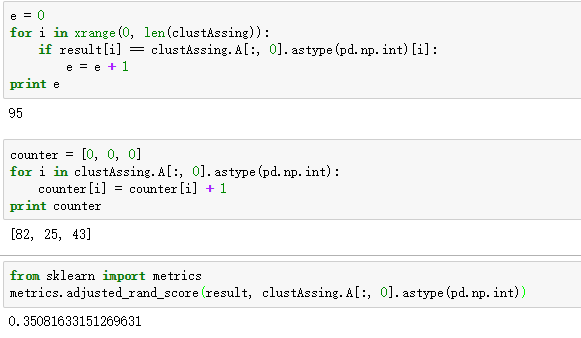


此处得出来的结果图分布与上面的kMeans算法对比，可以看到明显看到橙色部分占的面积更大了。下面展示起始数据绘制的图。



此时可以看到，黄色部分，也就是我们的versicolor不能很好的通过聚类体现出来，很多部分都被分到其他类内去了。效果看图都觉得不大好。

⑤ 结果评价



上述依旧是相同个数、每个类有的个数以及兰德指数的计算。此处可以看到，正确个数较kMeans低了很多，而兰德指数仅为0.35，小于kMeans的0.67。