**哈尔滨工业大学**

**数据挖掘理论与算法实验报告**

**(2015年度秋季学期)**

**课程编码** S1300019C

**授课教师 邹兆年**

**学生姓名 白辰甲**

**学 号 15S003002**

**学 院 计算机科学与技术**

**一、实验内容**

**1.问题：**使用K\_means聚类方法完成对二维欧式空间中的点的聚类。

**2.目标：**使用测试数据集，每个样本是空间中的点的二维坐标，使用这些样本运用算法进行聚类，最终生成K个簇，每个点被划归到某个簇中。其中K的个数为用户输入。

**3.算法：**

|  |  |
| --- | --- |
|  | K\_means聚类算法 |
| 1 | 选择K个点作为初始质心 |
| 2 | Repeat： |
| 3 | 将每个点指派到最近的质心 |
| 4 | 重新计算每个簇的质心 |
| 5 | Until 质心不发生变化 |

**4.期望结果：**

期望能够在有限计算步骤内使得K均值聚类算法能够收敛，即质心的位置不再发生变化，从而达到最小的SSE。

**二、实验设计**

**1.程序流程图**

见右图所示

**2.数据结构**

使用类来存储每个数据点，代码如下。其中：

x和y分别代表数据点的x坐标和y坐标。

dis[]是一个数组，存储该点到K个簇质心的距离。

cluster存储一个整数，代表该点属于哪个簇。

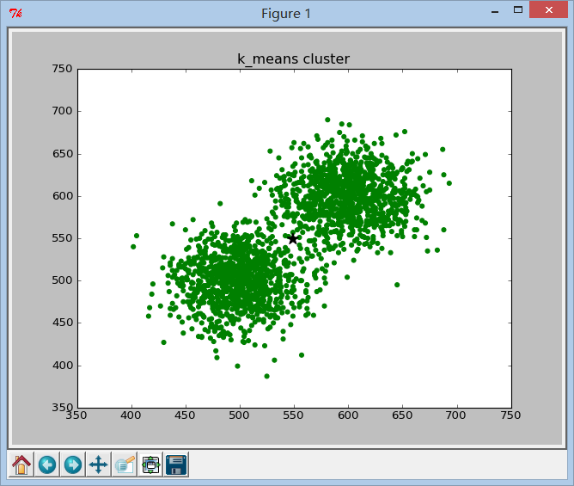
其中cluster的获取需要通过比较dis数组中距离最小的簇来获取。

|  |
| --- |
| class Data:  def \_\_init\_\_(self, x, y):  self.x = x  self.y = y  self.dis = []  self.cluster = 0 |

**3.关键函数及说明**

|  |  |  |
| --- | --- | --- |
| 函数 | **说明** | **关键代码** |
| Readdata | 读取实验数据集中的数据，为每个点建立一个Data类的对象。 | trainlist = []  for line in fp.readlines():  line = line.strip()  X = re.split(r'\s+', line)  newdata = Data(int(X[0]), int(X[1]))  trainlist.append(newdata) |
| Cal\_dis | 计算两点的欧氏距离 | temp1 = pow((x - x\_center), 2)  temp2 = pow((y - y\_center), 2)  res = temp1 + temp2  res1 = sqrt(res) |
| kmeans\_cluster | 对于给定的簇质心，将每个点划归到最近的簇中 | # 将每个点划分到某个簇内  for point in traindata:  min\_dis = min(point.dis)  cho\_cluster = point.dis.index(min\_dis)  point.cluster = cho\_cluster+1 |
| kmeans\_main | 给定迭代次数，调用kmeans\_cluster函数，在有限步内使得质心位置不再变化 | # 迭代调用函数计算簇质心  iter\_num = 1000  while iter\_num:  new\_center = kmeans\_cluster(center)  if new\_center[0][0] == center[0][0]:  break  center = new\_center  iter\_num -= 1  print "最终质心位置：", center[:] |
| 主函数 | 接收用户输入的K值 | global traindata  traindata = readdata()[:]  k1 = input("please input the cluster number: ")  kmeans\_main(k1) |

**三、测试数据**

**1.数据集描述**

数据由二维正态分布产生，在欧式空间中呈现分离较为明显的两个簇，如右图所示。

**2.数据样例**

前10个样本如下表：

|  |
| --- |
| 500 537 |
| 522 507 |
| 438 567 |
| 515 500 |
| 509 475 |
| 562 495 |
| 499 549 |
| 512 518 |

**3.预处理过程**

数据质量良好，只需要去掉数据前后的无关字符，再将一行数据分割为两个数字即可。

line = line.strip()

X = re.split(r'\s+', line)

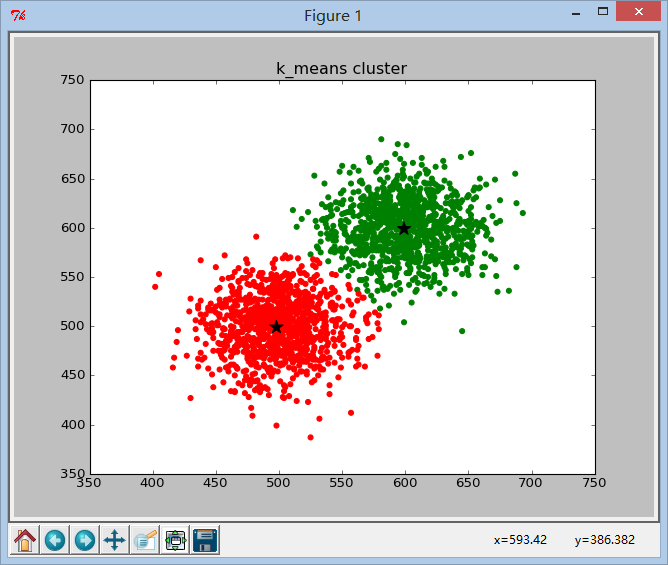
**四、实验结果**

**1. K = 2**

运行效果：

最终质心位置： [(599.0, 599.0), (498.0, 499.0)]

每个簇点的个数是 1021 1027

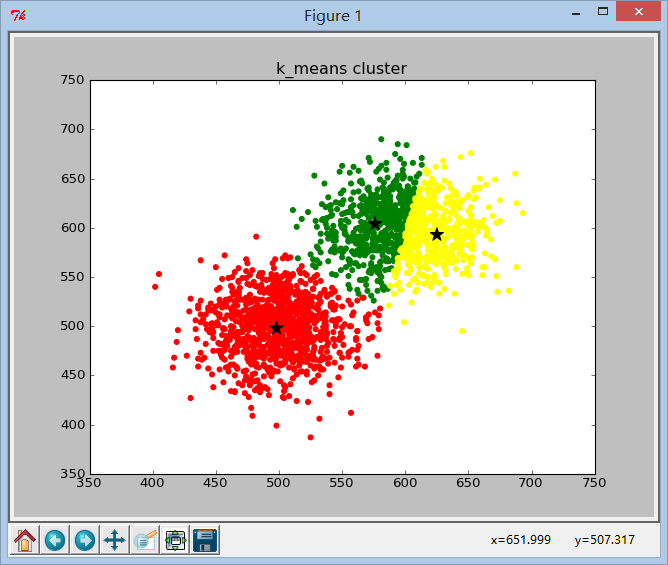


**2. K = 3**

运行效果：

最终质心位置： [(576.0, 604.0), (498.0, 498.0), (625.0, 593.0)]

每个簇点的个数是 539 1016 493

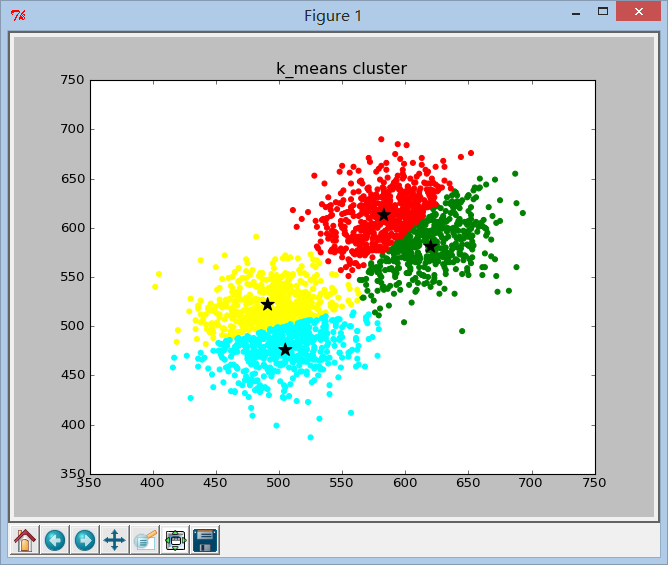


**2. K = 4**

运行效果：

最终质心位置： [(620.0, 581.0), (583.0, 613.0), (491.0, 522.0), (505.0, 476.0)]

每个簇点的个数是 465 558 523 502



**五、遇到的困难及解决方法、心得体会**

在最初考虑数据结构时，想到用二维数组存储点的坐标，用对应的数组存储距离矩阵和每个点所属的质心。这种数据结构的好处是速度快，但可读性差且容易混淆。故而后来使用了类来存储每个数据点的所有属性，更新属性只需要在每个点对象中更新，这样结构更加清晰。