

kmeans算法实验报告

**机器学习实验报告**

课程名称 机器学习

专业班级 计算机少61

姓 名 熊兴宇

学 号 2140506094

目录

[一、 实验要求](#_Toc31763_WPSOffice_Level1) [3](#_Toc31763_WPSOffice_Level1)

[1. 实现K-means方法](#_Toc9398_WPSOffice_Level2) [3](#_Toc9398_WPSOffice_Level2)

[2. 在数据集上尝试K-means方法](#_Toc10446_WPSOffice_Level2) [3](#_Toc10446_WPSOffice_Level2)

[二、 实验原理](#_Toc9398_WPSOffice_Level1) [3](#_Toc9398_WPSOffice_Level1)

[1. 实验步骤](#_Toc12075_WPSOffice_Level2) [3](#_Toc12075_WPSOffice_Level2)

[2. 关于K-means收敛的说明](#_Toc4314_WPSOffice_Level2) [3](#_Toc4314_WPSOffice_Level2)

[三、 实验过程](#_Toc10446_WPSOffice_Level1) [4](#_Toc10446_WPSOffice_Level1)

[1. 实验环境](#_Toc18282_WPSOffice_Level2) [4](#_Toc18282_WPSOffice_Level2)

[2. 读取数据集 4](#_Toc18282_WPSOffice_Level2)

[3. Test类读取本地数据集文件并生成kmeans类实例 4](#_Toc18282_WPSOffice_Level2)

[4. K-means类初始化随机生成初始质心](#_Toc8976_WPSOffice_Level2) [4](#_Toc8976_WPSOffice_Level2)

[① 全局变量](#_Toc15617_WPSOffice_Level3) [6](#_Toc15617_WPSOffice_Level3)

[② 类初始化](#_Toc28227_WPSOffice_Level3) [6](#_Toc28227_WPSOffice_Level3)

[③ 聚类质心初始化](#_Toc27726_WPSOffice_Level3) [6](#_Toc27726_WPSOffice_Level3)

[5. K-means类循环](#_Toc7058_WPSOffice_Level2) [6](#_Toc7058_WPSOffice_Level2)

[6. K-means类输出收敛前后聚类结果](#_Toc13147_WPSOffice_Level2) [7](#_Toc13147_WPSOffice_Level2)

[四、 实验分析](#_Toc12075_WPSOffice_Level1) [8](#_Toc12075_WPSOffice_Level1)

[五、 实验心得](#_Toc4314_WPSOffice_Level1) [9](#_Toc4314_WPSOffice_Level1)

1. 实验要求
2. 实现K-means方法
3. 在数据集上尝试K-means方法
4. 实验原理
5. 实验步骤

K均值是发现给定数据集的k个簇的算法。簇个数k是用户给定的，每一个簇通过其质心来描述。K均值工作流程是这样的:首先，随机选择K个初始质心，其中K是用户指定的参数，即所期望的簇的个数。然后将数据集中每个点指派到最近的质心，而指派到一个质心的点即为一个簇。然后，根据指派到簇的点，将每个簇的质心更新为该簇所有点的平均值。重复指派和更新步骤，直到簇不发生变化，或等价地直到质心不发生变化。

K-means伪代码

创建 k 个点作为起始质心 (随机选择):

当任意一个点的簇分配结果发生改变的时候:

对数据集中的每个数据点:

对每个质心:

计算质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:

求出均值并将其更新为质心

1. 关于K-means收敛的说明
2. means面对的第一个问题是如何保证收敛，前面的算法中强调结束条件就是收敛，可以证明的是K-means完全可以保证收敛性。下面我们定性的描述一下收敛性，我们定义畸变函数如下：



畸变函数表示每个样本点到其质心的距离平方和。K-means是要将J调整到最小。假设当前J没有达到最小值，那么首先可以固定每个类的质心，调整每个样例的所属的类别来让J函数减少，同样，固定，调整每个类的质心也可以使J减小。这两个过程就是内循环中使J单调递减的过程。当J递减到最小时，模型收敛。

由于畸变函数J是非凸函数，意味着我们不能保证取得的最小值是全局最小值，也就是说k-means对质心初始位置的

选取比较敏感，但一般情况下k-means达到的局部最优已经满足需求。

1. 实验过程
2. 实验环境

Windows 10操作系统python3.7，sublime，git bash

使用的库有Numpy，random，matplotlib，os，sys，用于矩阵计算，随机初始化，画图，读取工作路径

tests.py

1. work\_path=sys.path[0]
2. 读取数据集

tests.py

1. **def** loadDataSet(fileName):
2. numFeat = len(open(fileName).readline().split(','))    # 计算有多少列
3. dataMat = []
4. labelMat = []
5. fr = open(fileName)
6. **for** line **in** fr.readlines():        #  遍历原始数据集每一行
7. **if**(line=='\n'):
8. **continue**
9. lineArr =[]
10. curLine = line.split(',')      # 是一列表类型,这里先采用iris数据集，以逗号分开
11. **for** i **in** range(numFeat-1):     # numFeat - 1的原因：因为原始数据的最后一列是类别，不是属性数据
12. lineArr.append(float(curLine[i]))  # 一个一个传进lineArr列表向量
13. dataMat.append(lineArr)     # 再传进dataMat列表向量
14. labelMat.append(str(curLine[-1]).strip('\n'))  # 写进标签列表，去除空格
15. **return** dataMat, labelMat

注意iris数据集使用逗号分隔数据，可能别的数据集不同需要稍作修改。

1. Test类读取本地数据集文件并生成kmeans类实例

tests.py

1. **class** test():
2. dataMat,labelMat=loadDataSet(work\_path+r'/iris/iris.data')
4. **def** test\_kmeans(self):
5. k=3
6. kmeans\_exam=kmeans(k,self.dataMat)
7. kmeans\_exam.init\_center()
8. kmeans\_exam.iterate()

k是初始聚类中心的个数，K-means算法不需要带标签数据集来训练，初始化中心找到循环收敛结果即可。

1. K-means类初始化随机生成初始质心

kmeans.py

1. **from** random **import** uniform
2. **import** matplotlib.pyplot as plt
3. **import** numpy as np
4. DIM\_X=0
5. DIM\_Y=1
6. MAX\_ITER\_DEPTH=3
7. ITER\_LIMIT=False
8. DRAW=False
10. **class** kmeans():
11. **def** \_\_init\_\_(self,knum,args):
12. '''''
13. expect args as a list with multiple dimension data, each element is a dictionary, use the feature name (like x,y) as its key, the value of features as its values
14. '''
16. self.knum=knum
17. self.data=np.array(args)#二维数组，数据矩阵
18. self.d\_len=len(self.data[0])#数据特征长度
19. self.max\_data=args[0][:]#必须要加.copy或者[:](仅限于list)否则max\_data min\_data共享内存地址,因为此处不用改变args了所以[:]复制也可以
20. self.min\_data=args[0][:]
21. self.center\_list=[]#kmeans递归中生成的聚类中心
23. self.data\_class=[0 **for** i **in** range(len(self.data))]#该数组用于指示当前循环中某个数据点属于某类，下标顺序与self.data相同
24. #self.sigma=para\_variance
25. #self.N=gauss\_distrib\_num
26. #self.Miu=[]
27. self.max\_data=self.data.max(axis=0)
28. self.min\_data=self.data.min(axis=0)
30. self.colors=np.array([[i//4%2,i//2%2,i%2] **for** i **in** range(1,self.knum+1)])#rgb三色组
31. self.iter\_depth=0#当前函数执行次数（递归深度）
32. self.fig, self.axs = plt.subplots(MAX\_ITER\_DEPTH+1,2, figsize=(10, 10))
34. **def** init\_center(self):
35. '''''
36. use the max and the min of single data dimension as the margin of data space, randomly choose k centers from the space
37. '''
38. **for** i **in** range(0,self.knum):
39. center=uniform(self.min\_data,self.max\_data)
40. self.center\_list.append(center)
41. self.center\_list=np.array(self.center\_list)
42. 全局变量

DIM\_X，DIM\_Y是为了iris四维数据集抽取其中两个特征作图，使聚类结果可视化地显现。

MAX\_ITER\_DEPTH是最大循环深度，为了更方便debug和追踪循环结果，查看聚类收敛的趋势

ITER\_LIMIT是是否启用最大深度

DRAW是是否启用绘图

1. 类初始化

接收两个参数，一个是分类个数knum，一个是数据集

1. 聚类质心初始化

在四个特征的最大最小值区间之间随机生成四维点作为初始聚类质心

1. K-means类循环
2. **def** iterate(self):
3. self.kmeans\_iterate()#本来想用do while的，python好像没有哈哈
4. **while**(self.recal\_center()):
5. self.kmeans\_iterate()
6. self.**print**(self.axs,MAX\_ITER\_DEPTH)#输出聚类收敛的结果
7. plt.show()

10. **def** two\_norm(self,data1,data2):
11. norm=0
12. **for** i **in** range(self.d\_len):
13. norm+=(data1[i]-data2[i])\*\*2
14. **return** norm\*\*0.5

17. **def** recal\_center(self):
18. old\_center\_list=self.center\_list.copy()#不能用[:]而用浅复制,这两者有区别
19. **for** i **in** range(self.knum):
20. '''''
21. 以下相当于计算某类中所有点的平均几何中心，本来想用np中的数组sum运算再除一下，但self.data\_class这个东西我试了下用化为np矩阵算有一些问题，
22. np矩阵不太好进行大规模增改，同时指定元素获取下标也不是太简单，还是用普通的一维数组一步一步写吧
23. 本来的思路是np矩阵的每一个列向量代表某类点的集合，这样axis=0统计和非常方便，但是要在一个矩阵中查找指定元素把它挪到另一个列向量里就很麻烦了
24. '''
25. sum=np.zeros(self.d\_len)
26. num=0
27. **for** j **in** range(len(self.data)):
28. **if** i==self.data\_class[j]:
29. num+=1
30. sum+=self.data[j]
31. **if** num==0:
32. **continue**
33. self.center\_list[i]=sum/num
34. **if** (old\_center\_list==self.center\_list).all():#检验结果是否已经收敛
35. **print**('stop iterate')
36. **return** 0
37. self.iter\_depth+=1
38. **return** 1

41. **def** kmeans\_iterate(self):
42. **if** self.iter\_depth>=MAX\_ITER\_DEPTH **and** ITER\_LIMIT:#用于提前终止递归或者避免永远递归
43. **return** 0
44. **for** i **in** range(len(self.data)):
45. data=self.data[i]
46. dis=np.array([self.two\_norm(data,self.center\_list[i]) **for** i **in** range(self.knum)])
47. class\_num=np.argmin(dis)#获取dis数组中最小值的下标，即找到离该数据点最近的聚类中心（二范数就是欧式距离）
48. self.data\_class[i]=class\_num
49. **if** self.iter\_depth<MAX\_ITER\_DEPTH:#查看前几次递归的结果
50. self.**print**(self.axs,self.iter\_depth)
51. **return** 0

iterate调用kmeans\_iterate和recal\_center，直至recal\_center传来收敛结束的信号

kmeans\_iterate用于将每个数据点分类到某个中心对应的簇上，这个中心就是欧式距离最近的中心

recal\_center用于重新计算每个簇的中心，如当前聚类中心与上次聚类中心相同，可以认为收敛

two\_norm，自己写的二范式，用于估计中心

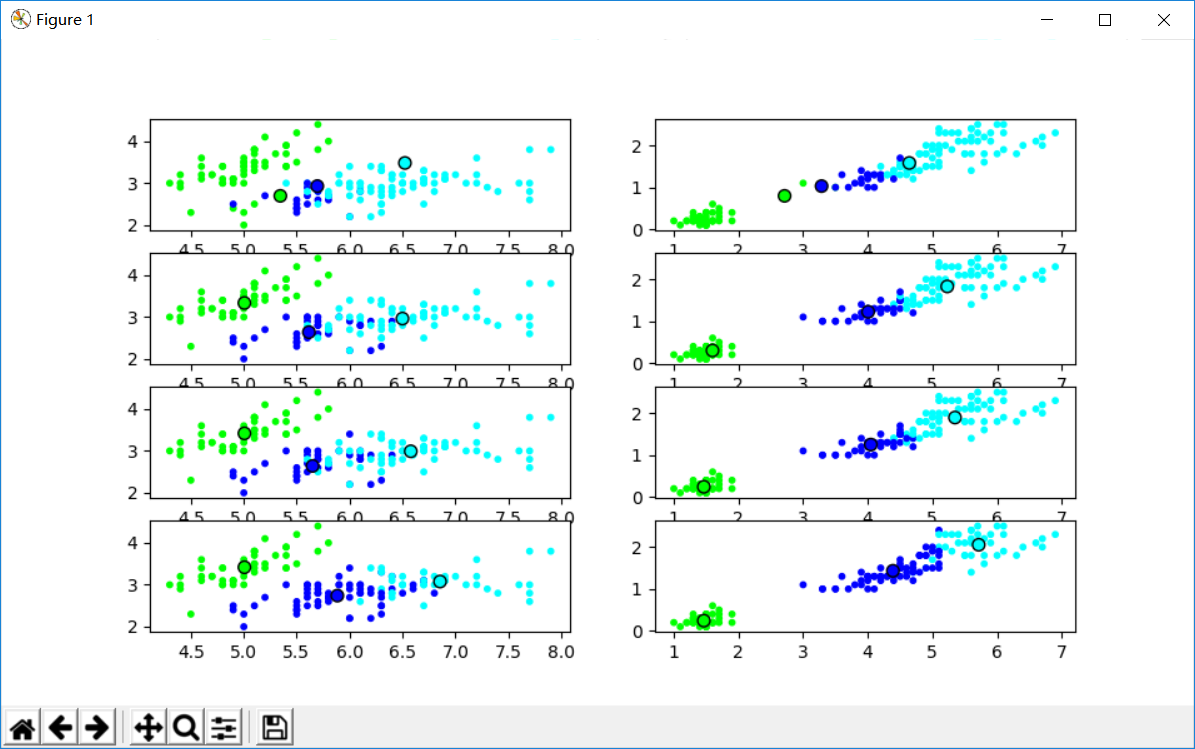
1. K-means类输出收敛前后聚类结果
2. **def** **print**(self,axs,pic\_n):
3. selectx=DIM\_X
4. selecty=DIM\_Y
5. x=self.data[:,selectx]
6. y=self.data[:,selecty]#显然，对于iris的四维数据空间，只能选取其中两项或三项来展示，这也导致聚类中心“看起来”不一定在中间
7. c=[self.colors[i] **for** i **in** self.data\_class]
8. axs[pic\_n,0].scatter(x=x,y=y,c=c,s=10)
9. axs[pic\_n,0].scatter(x=self.center\_list[:,selectx],y=self.center\_list[:,selecty],c=self.colors,marker='o',s=50,edgecolors='k')
10. #以下两项可以除去，仅适用于当前数据集（iris）
11. axs[pic\_n,1].scatter(x=self.data[:,2],y=self.data[:,3],c=c,s=10)
12. axs[pic\_n,1].scatter(x=self.center\_list[:,2],y=self.center\_list[:,3],c=self.colors,marker='o',s=50,edgecolors='k')

选取聚类未收敛前的部分结果查看，并查看最终收敛结果。

将iris两维特征与另外两维特征放在同一行，不同循环次数放在同一列方便对比

1. 实验分析

设定输出循环深度为3，即返回前三次的聚类结果和收敛结果



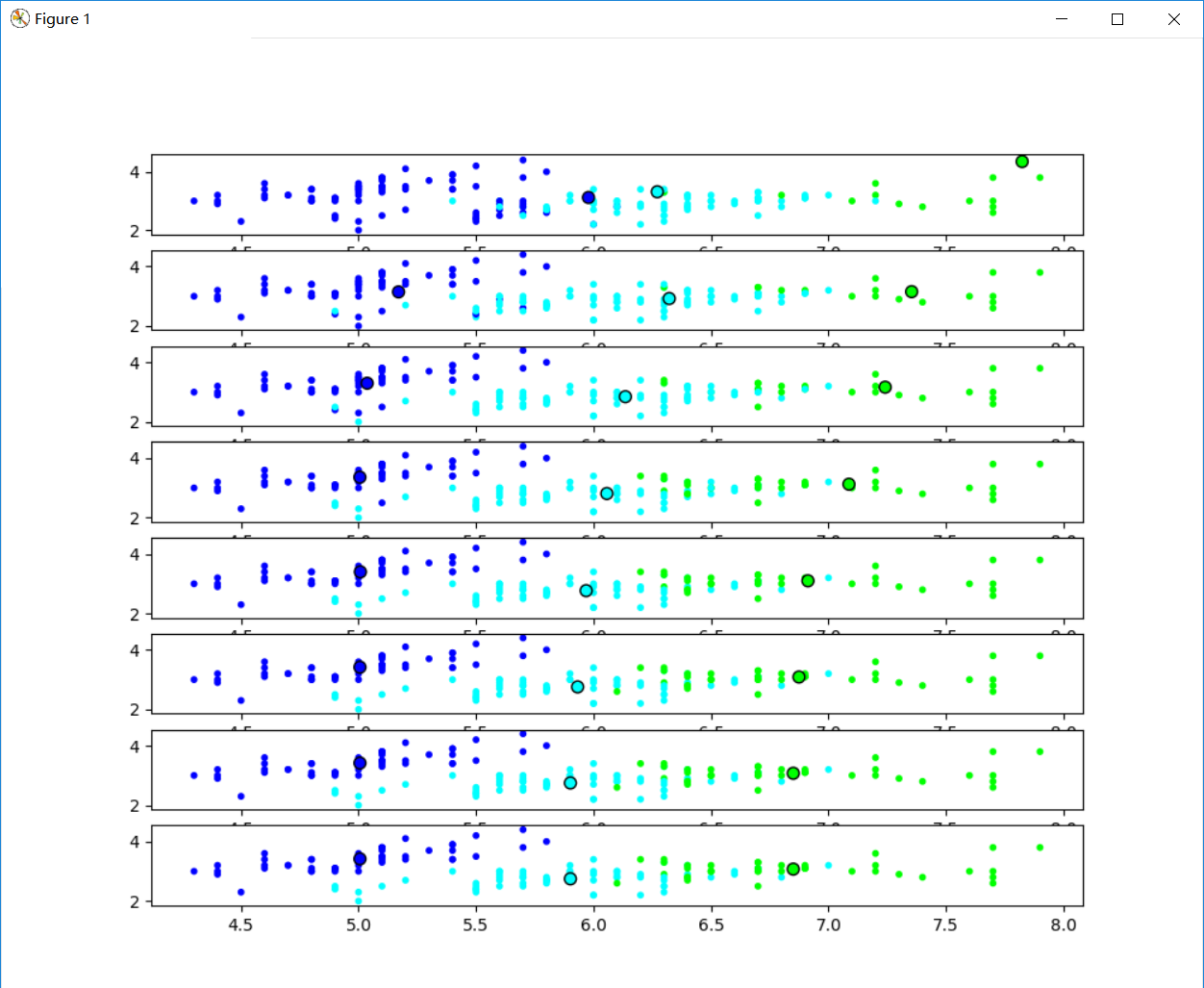
前三行分别对应前三次循环，最后一行为收敛以后的聚类结果。

左边一列的横纵坐标x:sepal length,y:sepal width

右边一列的横纵坐标x:petal length,y:petal width

显然二维图的质心不是那么直观的几何中心，从每个图来看，总存在那些看似离质心很远，但却又归到质心对应的簇的数据点。

设定循环次数为8，只看iris的两个特征



如图，在8次循环后聚类中心几乎不变了

1. 实验心得

虽然K-means算法简单，但是实现起来也不容易。尤其是关于调matplotlib和numpy库，我花了一些时间读了部分文档，总算有了一定了解。对于K-means这样的算法尚且如此，只有在完全理解算法后才能自己真正地实现。