**PythonK-means算法对iris数据聚类**

**相关知识**

K-means算法是将样本聚类成k个簇（cluster），具体算法描述如下:

1、 随机选取k个聚类质心点（cluster centroids）为

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/01.png)

2、 重复下面过程直到收敛 {

对于每一个样例i，计算其应该属于的类

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/02.png)

对于每一个类j，重新计算该类的质心

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/03.png)

}

K是我们事先给定的聚类数，

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/04.png)

代表样例i与k个类中距离最近的那个类，

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/04.png)

的值是1到k中的一个。质心

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/05.png)

代表我们对属于同一个类的样本中心点的猜测，拿星团模型来解释就是要将所有的星星聚成k个星团，首先随机选取k个宇宙中的点（或者k个星星）作为k个星团的质心，然后第一步对于每一个星星计算其到k个质心中每一个的距离，然后选取距离最近的那个星团作为

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/04.png)

这样经过第一步每一个星星都有了所属的星团；第二步对于每一个星团，重新计算它的质心

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/05.png)

（对里面所有的星星坐标求平均）。重复迭代第一步和第二步直到质心不变或者变化很小。求点群中心的算法：

一般来说，求点群中心点的算法你可以很简单使用各个点的X/Y坐标的平均值。不过，我这里想告诉大家另三个求中心点的的公式：

1）Minkowski Distance公式——λ可以随意取值，可以是负数，也可以是正数，或是无穷大。

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/06.png)

2）Euclidean Distance公式——也就是第一个公式λ=2的情况

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/07.png)

3）CityBlock Distance公式——也就是第一个公式λ=1的情况

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/08.png)

**系统环境**

Linux Ubuntu 16.04

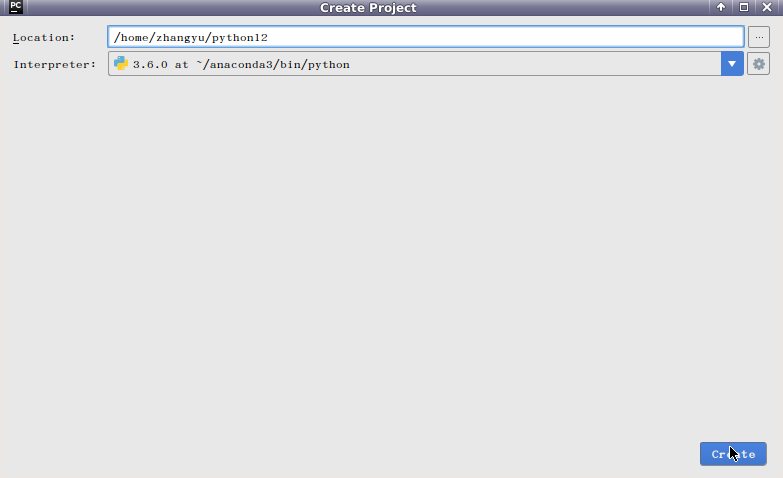
Python3.6

**任务内容**

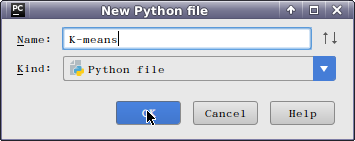
使用K-means算法对iris数据进行聚类，聚类的个数为3。iris数据（data，target），其中data包含了Petal\_Width, Petal\_Length，Sepal\_Width，Sepal\_Length这4个字段，共150条记录。在target中有3个类别分别为：Setosa、 Versicolour、Virginica。

**任务步骤**

1.新建python项目，名为python12

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/13.png)

2.在项目Python12中，新建Python file，名为：K-means

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/09.png)

3.在K-means文件中导入实验所需要的外包

[view plain](https://www.ipieuvre.com/e/181/311/9539) [copy](https://www.ipieuvre.com/e/181/311/9539)

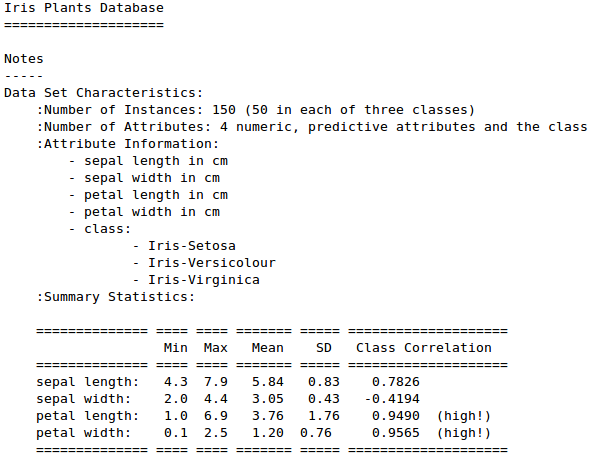
1. **import** numpy as np
2. **import** matplotlib.pyplot as plt
3. from mpl\_toolkits.mplot3d **import** Axes3D
4. from sklearn.cluster **import** KMeans
5. from sklearn **import** datasets

4.导入数据，并解析

[view plain](https://www.ipieuvre.com/e/181/311/9539) [copy](https://www.ipieuvre.com/e/181/311/9539)

1. iris=datasets.load\_iris()
2. print(iris.DESCR)

运行结果

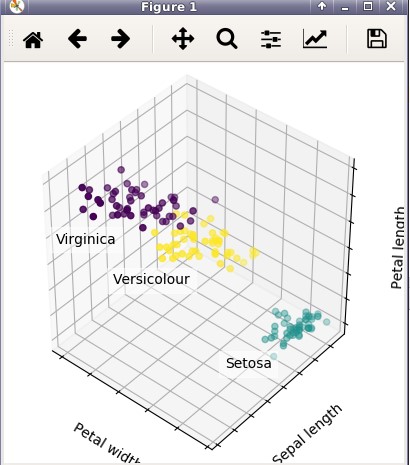
[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/10.png)

5.将下面代码追加到K-means文件中，用于探索分析

[view plain](https://www.ipieuvre.com/e/181/311/9539) [copy](https://www.ipieuvre.com/e/181/311/9539)

1. fig = plt.figure(num=1, figsize=(4, 3))
2. plt.clf()
3. ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
4. plt.cla()
5. **for** name, label **in** [('Setosa', 0),
6. ('Versicolour', 1),
7. ('Virginica', 2)]:
8. ax.text3D(X[y == label, 3].mean(),
9. X[y == label, 0].mean() + 1.5,
10. X[y == label, 2].mean(), name,
11. horizontalalignment='center',
12. bbox=dict(alpha=.5, edgecolor='w', facecolor='w'))
13. # Reorder the labels to have colors matching the cluster results
14. y = np.choose(y, [1, 2, 0]).astype(np.**float**)
15. ax.scatter(X[:, 3], X[:, 0], X[:, 2], c=y)
16. ax.w\_xaxis.set\_ticklabels([])
17. ax.w\_yaxis.set\_ticklabels([])
18. ax.w\_zaxis.set\_ticklabels([])
19. ax.set\_xlabel('Petal width')
20. ax.set\_ylabel('Sepal length')
21. ax.set\_zlabel('Petal length')
22. plt.show()

运行结果

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/11.jpg)

6.特征提取

[view plain](https://www.ipieuvre.com/e/181/311/9539) [copy](https://www.ipieuvre.com/e/181/311/9539)

1. X = iris.data
2. y = iris.target

7.sklearn中KMeans算法的结构

[view plain](https://www.ipieuvre.com/e/181/311/9539) [copy](https://www.ipieuvre.com/e/181/311/9539)

1. sklearn.cluster.KMeans(n\_clusters=8, init='k-means++', n\_init=10, max\_iter=300, tol=0.0001, precompute\_distances='auto', verbose=0, random\_state=None, copy\_x=True, n\_jobs=1, algorithm='auto')

参数：

n\_clusters ：int型，聚类的数量，默认为8.

max\_iter ： int，执行一次k-means算法所进行的最大迭代数，默认为 300。

n\_init： 整形，用不同的质心初始化值运行算法的次数，最终解是在inertia意义下选出的最优结果，缺省值=10。

init： 有三个可选值：'k-means++'， 'random'，或者传递一个ndarray向量。此参数指定初始化方法，默认值为 ‘k-means++’。‘k-means++’ 用一种特殊的方法选定初始质心从而 能加速迭代过程的收敛，参见 k\_init 的解释获取更多信息。‘random’ 随机从训练数据中选取初始质心。如果传递的是一个ndarray，则应该形如 (n\_clusters, n\_features) 并 给出初始质心。

precompute\_distances：三个可选值，‘auto’，True 或者 False。预计算距离，计算速度更快但占用更多内存。‘auto’：如果 样本数乘以聚类数大于 12million 的话则不预计算距 离。True：总是预先计算距离。False：永远不预先计算距离。

tol：float形，默认值= 1e-4，与inertia结合来确定收敛条件。

n\_jobs：整形数。指定计算所用的进程数。内部原理是同时进行n\_init指定次数的计算。若值为 -1，则用所有的CPU进行运算。若值为1，则不进行并行运算，这样的话方便调 试。若值小于-1，则用到的CPU数为(n\_cpus + 1 + n\_jobs)。因此如果 n\_jobs值为-2，则用到的CPU数为总CPU数减1。

random\_state：整形或 numpy.RandomState 类型，可选用于初始化质心的生成器（generator）。如果值为一个整数，则确定一个seed。此参数默认值为numpy的随机数生成器。

verbose：整形，默认值=0

属性：

cluster\_centers\_：向量，[n\_clusters, n\_features]

Coordinates of cluster centers (每个簇中心的坐标？？)

Labels\_:每个点的分类

inertia\_：float形，每个点到其簇的质心的距离之和。

8.建立KMeans聚类模型

[view plain](https://www.ipieuvre.com/e/181/311/9539) [copy](https://www.ipieuvre.com/e/181/311/9539)

1. kmeans = KMeans(n\_clusters=3, random\_state=0).fit(X)

9.模型评估

[view plain](https://www.ipieuvre.com/e/181/311/9539) [copy](https://www.ipieuvre.com/e/181/311/9539)

1. labels=kmeans.labels\_
2. acc = **float**((labels==y).sum())/len(y)
3. print('KMeans 的准确率为：%.2f' %(acc ))

10.完整代码如下：

[view plain](https://www.ipieuvre.com/e/181/311/9539) [copy](https://www.ipieuvre.com/e/181/311/9539)

1. **import** numpy as np
2. **import** matplotlib.pyplot as plt
3. from mpl\_toolkits.mplot3d **import** Axes3D
4. from sklearn.cluster **import** KMeans
5. from sklearn **import** datasets
6. centers = [[1, 1], [-1, -1], [1, -1]]
7. iris = datasets.load\_iris()
8. print(iris.DESCR)
9. X = iris.data
10. y = iris.target
11. kmeans = KMeans(n\_clusters=3, random\_state=0).fit(X)
12. labels=kmeans.labels\_
13. acc = **float**((labels==y).sum())/len(y)
14. print('KMeans 的准确率为：%.2f' %(acc ))
15. # Plot the ground truth
16. fig = plt.figure(num=1, figsize=(4, 3))
17. plt.clf()
18. ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
19. plt.cla()
20. **for** name, label **in** [('Setosa', 0),
21. ('Versicolour', 1),
22. ('Virginica', 2)]:
23. ax.text3D(X[y == label, 3].mean(),
24. X[y == label, 0].mean() + 1.5,
25. X[y == label, 2].mean(), name,
26. horizontalalignment='center',
27. bbox=dict(alpha=.5, edgecolor='w', facecolor='w'))
28. # Reorder the labels to have colors matching the cluster results
29. y = np.choose(y, [1, 2, 0]).astype(np.**float**)
30. ax.scatter(X[:, 3], X[:, 0], X[:, 2], c=y)
31. ax.w\_xaxis.set\_ticklabels([])
32. ax.w\_yaxis.set\_ticklabels([])
33. ax.w\_zaxis.set\_ticklabels([])
34. ax.set\_xlabel('Petal width')
35. ax.set\_ylabel('Sepal length')
36. ax.set\_zlabel('Petal length')
37. plt.show()

运行结果：

[](https://www.ipieuvre.com/doc/exper/286ba7ad-91ad-11e9-beeb-00215ec892f4/img/12.png)