**scikit-learn——密度聚类**

**任务目标**

1.了解密度聚类的原理

2.掌握sklearn中dbscan函数的使用

**相关知识**

DBSCAN(Density-Based Spatial Clustering of Applications with Noise)是一个比较有代表性的基于密度的聚类算法。与划分和层次聚类方法不同，它将簇定义为密度相连的点的最大集合，能够把具有足够高密度的区域划分为簇，并可在噪声的空间数据库中发现任意形状的聚类。

DBSCAN中的几个定义：

（1）Eps邻域：给定对象p的半径为Εps内的区域称为该对象p的Εps邻域；

（2）核心点：如果给定对象p的Ε领域内的样本点数大于等于MinPts，则称该对象p为核心点；

（3）边界点（edge point）：边界点不是核心点，但落在某个核心点的E邻域内;

（4）噪音点（outlier point）：既不是核心点，也不是边界点的任何点;

（5）直接密度可达：对于样本集合D，如果样本点q在p的Eps领域内，并且p为核心点，那么对象q从对象p直接密度可达。

（6）密度可达：对于样本集合D，给定一串样本点p1,p2….pn，p= p1,q= pn,假如对象pi从pi-1直接密度可达，那么对象q从对象p密度可达。

（7）密度相连：存在样本集合D中的一点o，如果对象o到对象p和对象q都是密度可达的，那么p和q密度相联。

可以发现，密度可达是直接密度可达的传递闭包，并且这种关系是非对称的。密度相连是对称关系。

算法流程：

1.将所有点标记为核心点，边界点或噪声点

2.删除噪声点。

3.为距离在Eps之内的所有核心点之间赋予一条边

4.每组连通的核心点形成一个簇。

5.将每个边界点指派到一个与之关联的核心点的簇中。

算法优点：

1.与K-means方法相比，DBSCAN不需要事先知道要形成的簇类的数量。

2.与K-means方法相比，DBSCAN可以发现任意形状的簇类。

3.同时，DBSCAN能够识别出噪声点。对离群点有较好的鲁棒性，甚至可以检测离群点。

4.DBSCAN对于数据库中样本的顺序不敏感，即Pattern的输入顺序对结果的影响不大。但是，对于处于簇类之间边界样本，可能会根据哪个簇类优先被探测到而其归属有所摆动。

5.DBSCAN被设计与数据库一同使用，可以加速区域的查询。例如 使用R\*树

算法缺点：

1. DBScan不能很好反映高维数据。

2. DBScan不能很好反映数据集以变化的密度。

3.由于DBSCAN算法直接对整个数据集进行操作，并且在聚类之前需要建立相应的R\*树，并绘制k-dist图，因此算法所需的内存空间和I/O消耗都相当可观。在计算资源有限而数据量又非常庞大的情况下，DBSCAN算法的效率将受到很大影响。（DBSCAN算法将区域查询得到的所有未被处理过的点都作为种子点，留待下一步扩展处理。对于大规模数据集中的较大类而言，这种策略会使种子点的数目不断膨胀，算法所需的内存空间也会快速增加。）

4.由于DBSCAN算法使用了全局性表征密度的参数，因此当各个类的密度不均匀，或类间的距离相差很大时，聚类的质量较差。(当各个类的密度不均匀或类间的距离相差很大时，如果根据密度较高的类选取较小的Eps值，那么密度相对较低的类中的对象Eps 邻域中的点数将小Minpts，则这些点将会被错当成边界点，从而不被用于所在类的进一步扩展，因此导致密度较低的类被划分成多个性质相似的类。与此相反，如果根据密度较低的类来选取较大的Eps值，则会导致离得较近而密度较大的类被合并，而它们之间的差异被忽略。所以在上述情况下，很难选取一个合适的全局Eps值来获得比较准确的聚类结果。)

5.DBSCAN不是完全确定的，边界点从不同的簇中获得，可以使不同簇的一部分，取决于数据处理。

6.DBSCAN的质量取决于regionQuery(P,Eps)函数中距离的测量。最常用的距离度量是欧式距离，尤其是在高维数据中，由于所谓的维数灾难，这种度量基本上是无用的，很难为E找到一个恰当的值。虽然目前有一些基于欧式距离的算法，但是如果不能对数据和规模有很好的了解，也很难找一个有意义的距离阈值E。

7.当密度差异大时，由于选取的MinPts-Eps组合不能同时适合所有的簇，DBSACN不能很好的进行数据聚类。（缺点4）

8.输入参数敏感，确定参数Eps , MinPts困难 ,若选取不当 ,将造成聚类质量下降。

9.由于经典的DBSCAN算法中参数Eps和MinPts在聚类过程中是不变的，使得该算法难以适应密度不均匀的数据集．

**scikit-learn提供的密度聚类算法模型为：**

[view plain](https://www.ipieuvre.com/e/410/311/9539) [copy](https://www.ipieuvre.com/e/410/311/9539)

1. **class** sklearn.cluster.DBSCAN(eps=0.5,min\_samples=5,metric='euclidean',algorithm='auto',leaf\_size=30,p=None,random\_state=None)

**参数说明：**

eps：float，用于确定邻域大小。

min\_samples：MinPts参数，用于判断核心对象。

metric：一个字符串或可调用对象，用于计算距离。如果是字符串，则必须是在metrics.pairwise.calculate\_distance中指定。

algorithm：一个字符串，用于计算两点间距离并找出最近邻的点，可以为如下：

* ‘auto’：由算法自动取舍合适的算法。
* ‘ball\_treeb’：用ball树来搜索。
* ‘kd\_tree’：用kd树搜索。
* ‘brute’：暴力搜索。

leaf\_size：int，默认是30，用于指定当algorithm=ball\_tree或kd\_tree时，树的叶节点大小。该参数会影响构建树，搜索最近邻的速度，同时影响树的内存。

random\_state：被废弃的接口，将在scikit-learn v 0.18中移除。

p：闵可夫斯基矩阵的权重被使用去计算两点之间的距离

n\_jobs：并行计算的数量，如果为-1，jobs的数量被设置成cup核的数量。

**属性说明：**

core\_sample\_indices\_：核心样本在原始训练集中的位置。

components\_：核心样本的一份副本。

labels\_：每个样本所属的簇标记。对于噪声样本，其簇标记为-1副本。

**方法说明：**

fit(X[,y,sample\_weight])：训练模型。

fit\_predict(X[,y,sample\_weight])：训练模型并预测每个样本所属的簇标记。

**系统环境**

Linux Ubuntu 16.04

Python 3.6.1

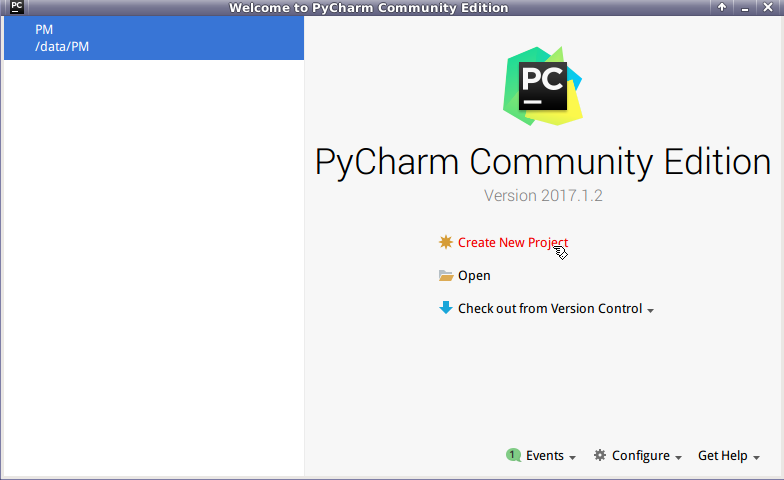
Jupyter

**任务内容**

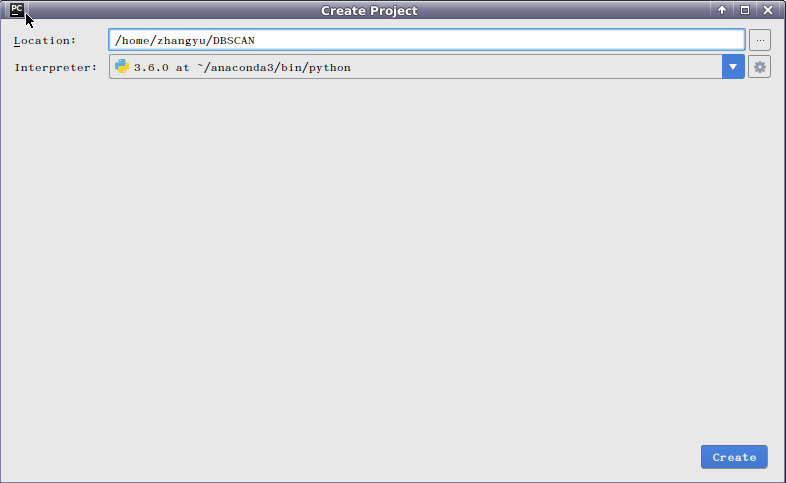
练习sklearn中密度聚类的操作过程。

**任务步骤**

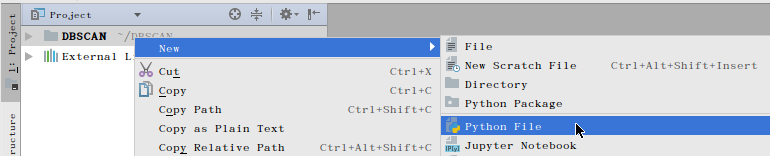
1.打开Pycharm，选择Create New Project，

[](https://www.ipieuvre.com/doc/exper/36989af0-91ad-11e9-beeb-00215ec892f4/img/804e776a-36d4-4c45-bbdc-c811ace41b67.png)

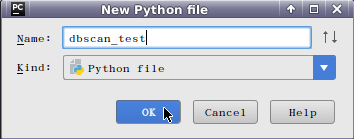
创建名为DBSCAN的项目。

[](https://www.ipieuvre.com/doc/exper/36989af0-91ad-11e9-beeb-00215ec892f4/img/7f3078bf-f1e3-469b-b160-c1b7f803f638.png)

2.打开DBSCAN项目，右键选择New=>Python File，

[](https://www.ipieuvre.com/doc/exper/36989af0-91ad-11e9-beeb-00215ec892f4/img/04551ec4-c551-4ec2-8477-bd090b3352b4.png)

创建名为dbscan\_test的Python文件。

[](https://www.ipieuvre.com/doc/exper/36989af0-91ad-11e9-beeb-00215ec892f4/img/085cb880-fa70-4c3c-b354-957437f95f1b.png)

3.打开dbscan\_test.py文件，编写代码，用于密度聚类。

导入实验所需要的模块

[view plain](https://www.ipieuvre.com/e/410/311/9539) [copy](https://www.ipieuvre.com/e/410/311/9539)

1. **import** numpy as np
2. from sklearn.cluster **import** DBSCAN
3. from sklearn **import** metrics
4. from sklearn.datasets import make\_blobs
5. from sklearn.preprocessing **import** StandardScaler

4.准备实验数据，使用make\_blobs函数创建训练数据X，标签数据labels\_true。

[view plain](https://www.ipieuvre.com/e/410/311/9539) [copy](https://www.ipieuvre.com/e/410/311/9539)

1. centers = [[1, 1], [-1, -1], [1, -1]]
2. X, labels\_true = make\_blobs(n\_samples=750, centers=centers, cluster\_std=0.4,
3. random\_state=0)

5.使用StandardScaler类创建对象，用对象fit\_transform函数对训练数据X进行标准化。

[view plain](https://www.ipieuvre.com/e/410/311/9539) [copy](https://www.ipieuvre.com/e/410/311/9539)

1. X = StandardScaler().fit\_transform(X)

6.对训练数据X进行密度聚类，使用DBSCAN类创建一个密度聚类对象，设置邻域半径大小参数eps=0.3，最小样本数参数min\_samples=10，然后使用fit()方法对训练数据X进行密度聚类返回值db，使用numpy中的zeros\_like函数通过bd.labels\_数据，生成一个值全为零矩阵，核心样本在原始训练集中的位置。

[view plain](https://www.ipieuvre.com/e/410/311/9539) [copy](https://www.ipieuvre.com/e/410/311/9539)

1. db = DBSCAN(eps=0.3, min\_samples=10).fit(X)
2. core\_samples\_mask = np.zeros\_like(db.labels\_, dtype=bool)
3. core\_samples\_mask[db.core\_sample\_indices\_] = True
4. labels = db.labels\_

7.聚类的数量

[view plain](https://www.ipieuvre.com/e/410/311/9539) [copy](https://www.ipieuvre.com/e/410/311/9539)

1. n\_clusters\_ = len(set(labels)) - (1 **if** -1 **in** labels **else** 0)
2. print('Estimated number of clusters: %d' % n\_clusters\_)

模型评估

8.同质性homogeneity：每个群集只包含单个类的成员。

[view plain](https://www.ipieuvre.com/e/410/311/9539) [copy](https://www.ipieuvre.com/e/410/311/9539)

1. print("Homogeneity: %0.3f" % metrics.homogeneity\_score(labels\_true, labels))

9.完整性completeness：给定类的所有成员都分配给同一个群集。

[view plain](https://www.ipieuvre.com/e/410/311/9539) [copy](https://www.ipieuvre.com/e/410/311/9539)

1. print("Completeness: %0.3f" % metrics.completeness\_score(labels\_true, labels))

10.两者的调和平均V-measure：

[view plain](https://www.ipieuvre.com/e/410/311/9539) [copy](https://www.ipieuvre.com/e/410/311/9539)

1. print("V-measure: %0.3f" % metrics.v\_measure\_score(labels\_true, labels))

11.调整兰德系数

[view plain](https://www.ipieuvre.com/e/410/311/9539) [copy](https://www.ipieuvre.com/e/410/311/9539)

1. print("Adjusted Rand Index: %0.3f"
2. % metrics.adjusted\_rand\_score(labels\_true, labels))

12.互信息：衡量两个数据分布的吻合程度

[view plain](https://www.ipieuvre.com/e/410/311/9539) [copy](https://www.ipieuvre.com/e/410/311/9539)

1. print("Adjusted Mutual Information: %0.3f"
2. % metrics.adjusted\_mutual\_info\_score(labels\_true, labels))

13.轮廓系数：

[view plain](https://www.ipieuvre.com/e/410/311/9539) [copy](https://www.ipieuvre.com/e/410/311/9539)

1. print("Silhouette Coefficient: %0.3f"
2. % metrics.silhouette\_score(X, labels))

模型可视化

14.使用matplotlib.pyplot模块，绘制密度聚类的结果图

[view plain](https://www.ipieuvre.com/e/410/311/9539) [copy](https://www.ipieuvre.com/e/410/311/9539)

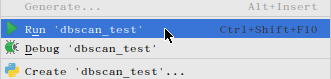
1. **import** matplotlib.pyplot as plt
2. # Black removed and is used for noise instead.
3. unique\_labels = set(labels)
4. colors = [plt.cm.Spectral(each)
5. **for** each **in** np.linspace(0, 1, len(unique\_labels))]
6. **for** k, col **in** zip(unique\_labels, colors):
7. **if** k == -1:
8. # Black used for noise.
9. col = [0, 0, 0, 1]
10. class\_member\_mask = (labels == k)
11. xy = X[class\_member\_mask & core\_samples\_mask]
12. plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
13. markeredgecolor='k', markersize=14)
14. xy = X[class\_member\_mask & ~core\_samples\_mask]
15. plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
16. markeredgecolor='k', markersize=6)
17. plt.title('Estimated number of clusters: %d' % n\_clusters\_)
18. plt.show()

15.完整代码如下：

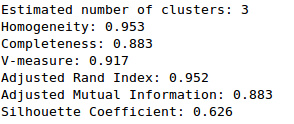
[view plain](https://www.ipieuvre.com/e/410/311/9539) [copy](https://www.ipieuvre.com/e/410/311/9539)

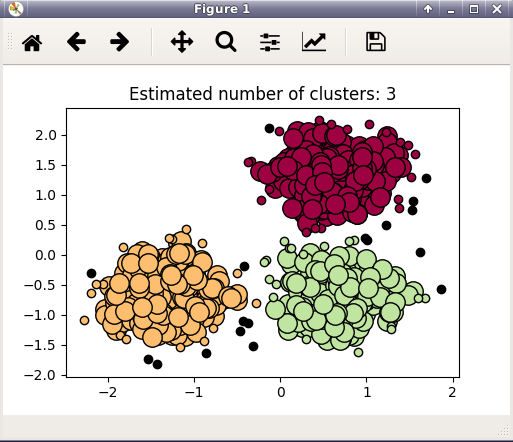
1. **import** numpy as np
2. from sklearn.cluster **import** DBSCAN
3. from sklearn **import** metrics
4. from sklearn.datasets.samples\_generator **import** make\_blobs
5. from sklearn.preprocessing **import** StandardScaler
6. # Generate sample data
7. centers = [[1, 1], [-1, -1], [1, -1]]
8. X, labels\_true = make\_blobs(n\_samples=750, centers=centers, cluster\_std=0.4,
9. random\_state=0)
10. X = StandardScaler().fit\_transform(X)
11. # Compute DBSCAN
12. db = DBSCAN(eps=0.3, min\_samples=10).fit(X)
13. core\_samples\_mask = np.zeros\_like(db.labels\_, dtype=bool)
14. core\_samples\_mask[db.core\_sample\_indices\_] = True
15. labels = db.labels\_
16. # Number of clusters in labels, ignoring noise if present.
17. n\_clusters\_ = len(set(labels)) - (1 **if** -1 **in** labels **else** 0)
18. print('Estimated number of clusters: %d' % n\_clusters\_)
19. print("Homogeneity: %0.3f" % metrics.homogeneity\_score(labels\_true, labels))
20. print("Completeness: %0.3f" % metrics.completeness\_score(labels\_true, labels))
21. print("V-measure: %0.3f" % metrics.v\_measure\_score(labels\_true, labels))
22. print("Adjusted Rand Index: %0.3f"
23. % metrics.adjusted\_rand\_score(labels\_true, labels))
24. print("Adjusted Mutual Information: %0.3f"
25. % metrics.adjusted\_mutual\_info\_score(labels\_true, labels))
26. print("Silhouette Coefficient: %0.3f"
27. % metrics.silhouette\_score(X, labels))
28. # Plot result
29. **import** matplotlib.pyplot as plt
30. # Black removed and is used for noise instead.
31. unique\_labels = set(labels)
32. colors = [plt.cm.Spectral(each)
33. **for** each **in** np.linspace(0, 1, len(unique\_labels))]
34. **for** k, col **in** zip(unique\_labels, colors):
35. **if** k == -1:
36. # Black used for noise.
37. col = [0, 0, 0, 1]
38. class\_member\_mask = (labels == k)
39. xy = X[class\_member\_mask & core\_samples\_mask]
40. plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
41. markeredgecolor='k', markersize=14)
42. xy = X[class\_member\_mask & ~core\_samples\_mask]
43. plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
44. markeredgecolor='k', markersize=6)
45. plt.title('Estimated number of clusters: %d' % n\_clusters\_)
46. plt.show()

16.代码编写完毕，在dbscan\_test.py文件内，点击右键=》Run ‘dbscan\_test’，执行dbscan\_test.py文件。

[](https://www.ipieuvre.com/doc/exper/36989af0-91ad-11e9-beeb-00215ec892f4/img/3795947f-a4f9-437c-89bd-fea46996258f.png)

17.运行结果：

[](https://www.ipieuvre.com/doc/exper/36989af0-91ad-11e9-beeb-00215ec892f4/img/fca44437-c1b2-4659-9995-cb59195eca1d.png)

[](https://www.ipieuvre.com/doc/exper/36989af0-91ad-11e9-beeb-00215ec892f4/img/37e37835-9a91-4714-b3ef-7ceb2773ac77.png)