LB4 机器学习概论

PB19151769　马宇骁

目录

1 实验要求

本次聚类实验主要实现的是《Clustering by fast search and find of density peaks》一文中的算法（以下简称 DPC）

- By Alex Rodriguez and Alessandro Laio
- Published on SCIENCE, 2014
- https://sites.psu.edu/mcnl/files/2017/03/9-2dhti48.pdf

本次实验的总体流程是完成 DPC 算法的代码实现，并在给定数据集上进行可视化实验。

1. 读取数据集，（如有必要）对数据进行预处理
2. 实现 DPC 算法，计算数据点的 δ_i 和 ρ_i
3. 画出决策图，选择样本中心和异常点
4. 确定分簇结果，计算评价指标，画出可视化图

本次实验采用 Davis-Bouldin Index (DBI)(sklearn.metrics. davies_bouldin_score) 作为评价指标.

2 实验原理

2.1 DPC 算法

DPC 算法的两个假设为：

1. 类簇中心被类簇中其他密度较低的数据点包围；
2. 类簇中心间的距离相对较远。

2.1.1 局部密度

假设 N 为样本个数，M 为样本维数。对于样本点 i 的局部密度，局部密度有两种计算方式，离散值采用截断核的计算方式，连续值则用高斯核的计算方式。

- 截断核：

$$\rho_i = \sum_{i \neq j} \chi\left(d_{ij} - d_c\right)$$

$$\chi(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases}$$

- 高斯核：

$$\rho_i = \sum_{i \neq j} \exp\left[-\left(\frac{d_{ij}}{d_c}\right)^2\right]$$

式中，di 为数据点 i 与数据点 j 的欧氏距离，dc 为数据点 i 的邻域截断距离。采用截断核计算的局部密度 ρ_i 等于分布在样本点 i 的邻域截断距离范围内的样本点个数；而利用高斯计算的局部密度 ρ_i 等于所有样本点到样本点 i 的高斯距离之和。DPC 算法的原论文指出，对于较大规模的数据集，截断核的计算方式聚类效果较好；而对于小规模数据集，高斯核的计算方式聚类效果更为明显。

2.1.2 相对距离

相对距离 δ_i 指样本点 i 与其他密度更高的点之间的最小距离。在计算样本点 i 前需要对每个数据点的局部密度进行排序。

对于密度最高的样本，相对距离定义为:

$$\delta_i = \max_{\mathrm{i} \neq \mathrm{j}} (\mathrm{d_{ij}})$$

对于其余数据点，相对距离定义为：

$$\delta_{\mathrm{i}} = \min_{\mathrm{j}:\rho_{\mathrm{j}} > \rho_{\mathrm{i}}} (\mathrm{d_{ij}})$$

由于密度最高的样本不存在比其密度更高的点，DPC 认为该点必为密度峰值 (类簇中心)，人为设定其相对距离为最大值。剩余的密度峰值需要同时满足两个条件：局部密度 ρ 较高，相对距离 δ 较大。为此，DPC 算法的原论文通过决策值 γ 寻找这类密度峰值，下式给出了 γ_i 的定义：

$$\gamma_i = \rho_i \times \delta_i$$

找到密度峰值后，DPC 将剩余数据点分配给密度比它高的最近数据点所在类簇，形成多个从密度峰值出发的树状结构，每一个树状结构代表一个类簇。

2.1.3 DPC 算法的执行步骤

1. 利用样本集数据计算距离矩阵 dij;
2. 确定邻域截断距离 dc;
3. 计算局部密度 ρ_i 和相对距离 δ_i;
4. 绘制决策图，选取聚类中心点;
5. 对非聚类中心数据点进行归类，聚类结束。

2.2 DB 指数 (Davies-Bouldin Index)

如果真实标签未知，可以使用 Davies-Bouldin Index(sklearn.metrics.davies_bouldin_score) 来评估模型，其中较低的 Davies-Bouldin 值与具有更好的集群分离的模型。这个指数表示集群之间的平均“相似度”，相似度是比较集群之间的距离和集群本身的大小的度量。零分是最低分。接近于零的值表示更好的分区。

参数	说明
x	array-like, shape (n_samples,n_features) n_features 维数据点列表。每行对应一个数据点。
labels	array-like, shape (n_samples,) 每个样本的预测标签。

表 1: sklearn.metrics.davies_bouldin_score

- 返回值

float 所得的 Davies-Bouldin 得分。

3 实验实现

3.1 实验调试

针对三个数据集，由于算法思路相同，第一个数据集运行时间相对较久，所以没有采用再次确定聚类的簇数进行重新拟合。对第二个和第三个进行了调试改进。

```
label1 = model1.fit()

# label2 = model2.fit(method=None) # 四个聚类
# label2 = model2.fit() # 10个聚类，还行
label2 = model2.fit(findtype = 2, K=14) # 根据图像手动指定了14个

# label3 = model3.fit() # 高斯4个聚类
# label3 = model3.fit(method=None) # 5个聚类
label3 = model3.fit(findtype = 2, K=7) # 根据图像手动指定了7个
```

其中，在定义的 class DPC() 中拟合的时候当不指定局部密度计算的时候默认用高斯核。

```
# DPC算法拟合
def fit(self, method="Gaussion", findtype = 1, K=1): # 指定1为find_centers_auto，2为
                                                      find_centers_K（需要自己输入K）

    # 距离矩阵
    dists = self.getDistanceMatrix()
    dc = self.select_dc()
    # 计算局部密度
    rho = self.get_density(method = method)
```

```
        # 计算密度距离
        deltas, nearest_neiber = self.get_deltas()
        # 获取聚类中心点
        if findtype == 1:
            centers = self.find_centers_auto()
        elif findtype == 2:
            centers = self.find_centers_K(K = K)
        labs = self.cluster_PD()

        return labs
```

3.2 实验结果

最终，三个聚类的结果如下：

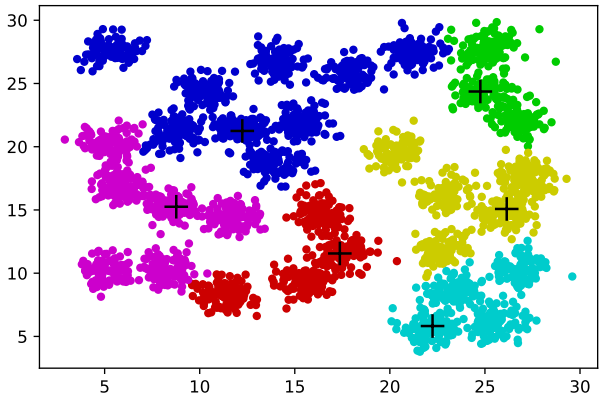

图 1: D31

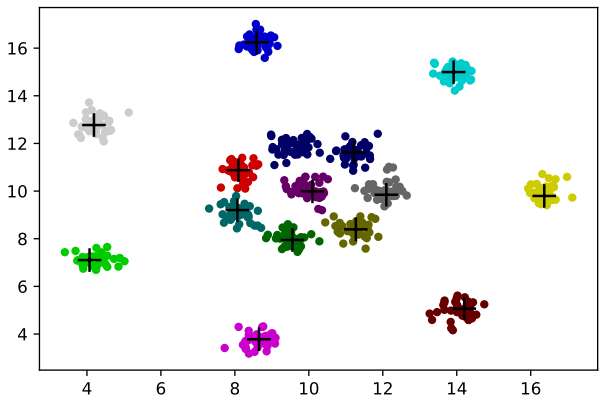

图 2: R15

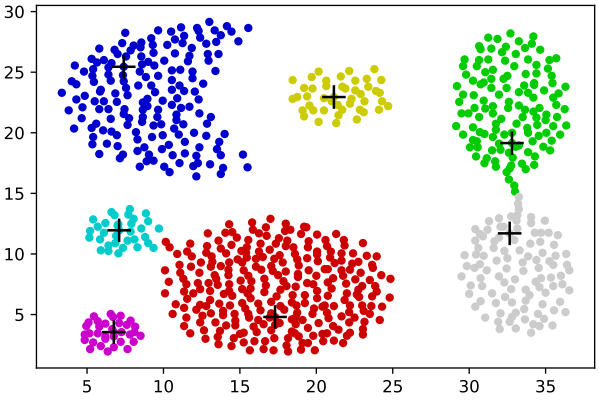

图 3: Aggregation

```
DBI(data1,label1), DBI(data2,label2), DBI(data3,label3)
(0.781987832988031, 0.37374794009855516, 0.5035680502991704)
```

可以看到，后面两个聚类结果非常漂亮！