

# DBSCAN 聚类简介

尹卓

北京工业大学

MichaelYinBJUT@163.com

2018 年 11 月 1 日

## 目录

1 引言	2
2 DBSCAN 的定义	2
3 DBSCAN 的基本流程	4
4 例子	6
4.1 使用 DBSCAN 对于轨迹停止点进行聚类 . . . . .	6
4.2 DBSCAN 对于几个特殊数据集进行聚类 . . . . .	7
5 总结	7

## 1 引言

聚类算法在基于地理信息的空间数据挖掘方面有着重要的应用。在空间聚类当中，针对聚类算法有这么几点要求：第一，聚类算法的涉及领域知识的超参数 (*Hyper parameters*) 越少越好；第二，聚类算法能够识别出任意形状的聚类；第三，能够高效的运行在大规模数据上。针对上述三点的要求，90 年代中期，Ester et al. [1] 提出了 DBSCAN 聚类算法，全称叫做 *Density Based Spatial Clustering of Applications with Noise*。在这里，我们首先简要介绍一下 DBSCAN 算法当中的一些基本定义，随后介绍 DBSCAN 的算法流程伪代码，最后给出了几个例子，来说明 DBSCAN 的可能用途以及缺点。

## 2 DBSCAN 的定义

DBSCAN 聚类算法被提出来的目的就是为了发现在空间当中带有噪声的任意形状的聚类。当我们观察二维平面上的点的分布情况的时候，如图1所示，一方面可以区分出平面上离群的噪声点（图 2, 3 中较为分散的点）；另一方面，我们也可以清晰的识别出有多少个聚类。能够如此清晰的辨识出聚类的主要原因是每一个聚类内部都有着明显的非常高的点的密度，而聚类外部点的密度又是非常低，而噪声点附近的密度也是明显低于任意的一个聚类的内部的密度的。

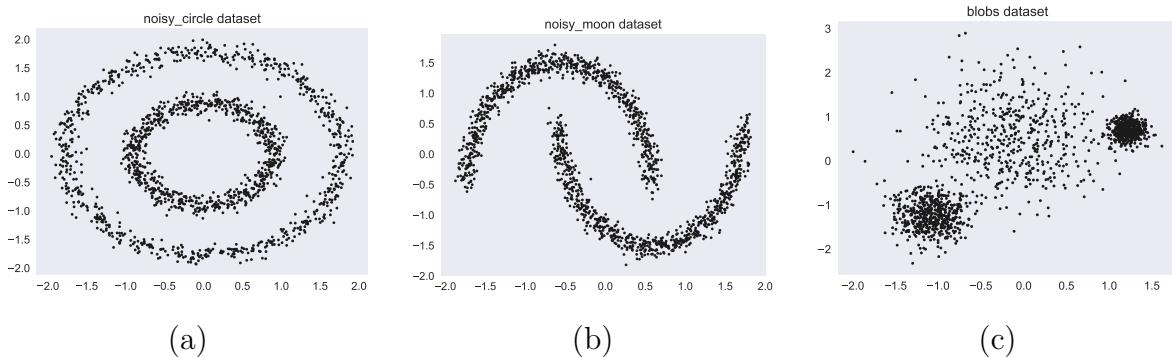


图 1: (a)circle 数据集 (b)moon 数据集 (c)blobs 数据集

由上所述，DBSCAN 聚类算法的核心思想就是：对于每一个属于某个聚类的数据点而言，以该数据点为中心的给定半径内，一定会至少包含一个最小数目的样本点的个数，这样才能形成聚类。由此思想，我们来介绍 DBSCAN 聚类算法的一些核心定义，这里  $D$  代表所有数据点的集合。

**定义 1 (数据点的 Eps 邻域)** 样本点  $p$  的  $Eps$  邻域我们使用  $N_{Eps}(p)$  来表示，并且定义为点的集合  $N_{Eps}(p) = \{q \in D | dist(p, q) \leq Eps\}$ ，其中  $|N_{Eps}(p)|$  代表  $Eps$  领域中点的个数。

**定义 2 (核心点)** 若是样本点  $p$  的  $Eps$  邻域对应的  $|N_{Eps}(p)| \geq MinPts$ ，定义点  $p$  为一个核心点 (*Core point*)。

在  $Eps$  邻域的定义之下，数据集  $D$  将会出现三种类型的点：位于聚类内部的点，被称之为核心点 (*Core points*)；位于聚类边界上的点，被称之为边界点 (*Border points*)，还有一种类型的点我们后面会提到。总的来说，边界点的  $Eps$  邻域内包含的点的个数，会显著的少于核心点的  $Eps$  邻域包含的点的个数。同时我们给出如下定义：

**定义 3 (密度直达)** 对于给定的  $Eps$  和  $MinPts$  值，若是：(1) $p \in N_{Eps}(q)$ ；(2) $|N_{Eps}(q)| \geq MinPts$ （说明  $q$  是个核心点）。则我们定义数据点  $p$  由点  $q$  密度直达 (*Directly density-reachable*)，也就是  $p$  到  $q$  密度直达。

显然，若是  $p$  是核心点的话，密度可达的定义是对称的，也就是说  $p$  到  $q$  密度直达，反之亦然。但是若是有  $p$  是边界点的话，那么就会导致密度直达的定义不对称。同时，我们考虑若是一个核心点的  $Eps$  邻域里也具有核心点，而这个核心点的  $Eps$  邻域又含有另外的核心点，那么我们的密度直达就可以通过核心点“传递”下去，根据此，我们给出密度可达 (*Density-reachable*) 的定义：

**定义 4 (密度可达)** 对于给定的  $Eps$  和  $MinPts$  参数，若是有一系列的点  $p_1, p_2, \dots, p_n$ ，其中任意两点  $p_{i+1}$  到  $p_i$  为密度直达，并且  $p_1 = q$ ,  $p_n = p$ ，则我们称  $p$  到  $q$  密度可达 (*Density-reachable*)。

密度可达是密度直达定义的扩充。密度可达的关系是传递的，但是同样和密度直达一样，密度可达关系不一定对称。但是若是  $p$  也是核心点的话，密度可达关系也将会是对称关系。我们考虑同一个聚类  $C$  里的两个边界点，对于这两个边界点可能并不会有密度可达关系，因为两个点都不是核心点，但是一定会存在一个位于聚类  $C$  里的核心点，这两个边界点到该核心点都将会是密度可达的关系。这样，我们给出密度连通 (*Density-connected*) 密度连通的定义：

**定义 5 (密度连通)** 对于给定的  $Eps$  和  $MinPts$  参数，若是存在一个点  $o$ ，而  $p$  与  $q$  到点  $o$  都是密度可达，则我们称点  $p$  到  $q$  是密度连通 (*Density-connected*)。

这里应当注意到，密度连通的概念是对称的，也就是说  $p$  到  $q$  密度连通，也就说明了  $q$  到  $p$  密度连通。有了上面的一系列的定义，我们现在可以来从直观上定义在 DBSCAN 这种聚类算法里的“聚类”的概念了。直观上说，我们指的“聚类”定义为一系列的具有最大密度可达性的密度连通的点。这里，我们给出聚类的严格定义，以及给出除了核心点和边界点之外，第三种点：噪声点 (*Noisy points*) 的定义：

**定义 6 (聚类)** 对于给定的  $Eps$  和  $MinPts$  参数，聚类 (*Cluster*) $C$  定义为数据点集  $D$  的满足如下条件的非空子集：(1) $\forall p, q$ : 若是  $q \in C$ ，并且对于给定的  $Eps$  和  $MinPts$  参数， $p$  到  $q$  密度可达，那么  $p \in C$  (最大性)。(2) $\forall p, q \in C$ : 对于给定的  $Eps$  和  $MinPts$ ， $p$  到  $q$  是密度连通的 (连通性)。

**定义 7 (噪声点)** 对于给定的  $Eps$  和  $MinPts$  参数，经过聚类之后，不属于任何聚类的点我们称之为噪声点 (*Noisy points*)。

注意到，聚类的边界点虽然不满足  $Eps$  下的  $MinPts$  条件，但是并不是噪声点。下面我们给出图2图形的例子来帮助理解上面所说的定义。如图所示，对于平面上的点集  $D = \{a, b, c, d, e\}$ ，我们给定  $Eps$  领域的范围，如图中包裹样本点的红色圆圈所示；同时我们给定  $MinPts = 3$ 。

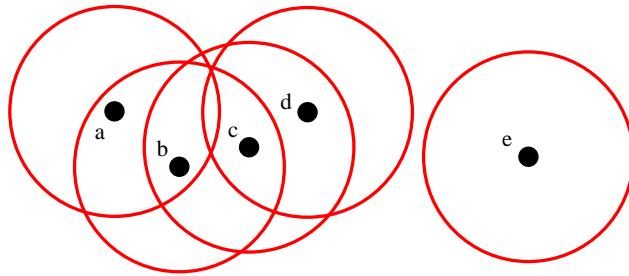


图 2: 给定  $Eps$  的条件下， $MinPts = 3$  的数据集

有了上面的定义，我们可以知道： $a, c$  到  $b$  满足密度直达 (*Directly density-reachable*) 的定义，其中  $b$  是一个核心点，而  $a$  与  $c$  不满足核心点的定义，所以是边界点； $d$  点到  $b$  点为密度可达 (*Density reachable*)，路径为  $b -> c -> d$ ，并且我们可以很明显的看出， $b$  与  $c$  点都是核心点，而  $d$  是边界点。 $a$  到点  $d$  之间是密度连通 (*Density-connected*)，因为存在  $b$  到  $a$  与  $d$  之间都是密度可达的。集合  $\{a, b, c, d\}$  组成了聚类，因为它们同时满足聚类所具有的特性，而聚类结束后， $e$  既不是核心点，又不属于任何聚类，所以就是一个噪声点。

### 3 DBSCAN 的基本流程

在给定了 DBSCAN 聚类算法的一些基础定义之后，我们可以来借助这些定义，来给出聚类算法的流程。其中 DBSCAN 形成聚类的思想上很简单：我们任意选定  $D$  中的一个点  $p$ ，判断  $p$  是不是核心点；若  $p$  是核心点的话，所有到  $p$  点密度可达的点组成的集合就是聚类；按照这种做法我们遍历整个数据集即可。我们给定的 DBSCAN 标准算法包括 3 个函数：遍历数据集的 DBSCAN 主函数；用来检索任意点的  $Eps$  邻域的点的 *RegionQuery* 函数；以及指定核心点之后，扩大核心点作为聚类的 *ExpandCluster* 函数。借助这 3 个函数，我们可以设计出最原始的 DBSCAN 算法。同时应当注意到，检索核心点  $Eps$  邻域的算法，是一个标准的  $k$  近邻 ( $k$  Nearest Neighbors) 问题，可以由高效的  $KD$  树，球树 (Ball Tree) 或者是  $R^*$  树等算法来计算。

---

#### Algorithm 1 RegionQuery

---

输入:  $P, Eps$

输出: 位于点  $P$  的  $Eps$  领域内的所有的点的集合（包括点  $P$  自身），这里可以使用标准的  $KD$  树算法或者是球树算法。

---

---

**Algorithm 2** ExpandCluster

---

**输入:**  $P, NeighborPts, C, Eps, MinPts$

**输出:** 无

```

1: 将点  $P$  加入聚类点集合  $C$ ,  $NeighborPts$  代表点  $P$  的 RegionQuery 的点的集合。
2: for  $NeighborPts$  中每一点  $P'$  do
3:   if  $P'$  没有被访问过 then
4:     将  $P'$  标记为已访问 (Visited)
5:      $NeighborPts' = \text{RegionQuery}(P', Eps)$ 
6:     if  $\text{sizeof}(NeighborPts') \geq MinPts$  then
7:       将  $NeighborPts'$  内所有的点加入  $NeighborPts$  集合中
8:     end if
9:   end if
10:  if  $P'$  不属于任何一个聚类 then
11:    将  $P'$  加入聚类点集合  $C$ 
12:  end if
13: end for

```

---



---

**Algorithm 3** Original DBSCAN

---

**输入:**  $D, Eps, MinPts$

**输出:** 类标记

```

1:  $C = 0$ 
2: for  $D$  中的每一点  $P$  do
3:   if  $P$  被访问过 (Visited) then
4:     continue
5:   end if
6:   将  $P$  标记为已访问 (Visited)
7:    $NeighborPts = \text{RegionQuery}(P, Eps)$ 
8:   if  $\text{sizeof}(NeighborPts) < MinPts$  then
9:     将  $P$  标记为  $NOISE$ 
10:  else
11:     $C = \text{next cluster}$ 
12:     $\text{ExpandCluster}(P, NeighborPts, C, Eps, MinPts)$ 
13:  end if
14: end for

```

---

同时, 应当注意到以上算法只是从原理上说明了 DBSCAN 的聚类流程, 并不是 DBSCAN 的最高效实现形式, 更加高效的实现方式可以参考文献 [2]。

## 4 例子

这里我们给出两个使用 DBSCAN 算法的例子，第一个例子是使用 DBSCAN 算法来发现轨迹数据中的停止点；而第二个例子，则使用 3 个人造的数据集来说明 DBSCAN 这种聚类算法的不足之处。

### 4.1 使用 DBSCAN 对于轨迹停止点进行聚类

第一个例子，我们利用 DBSCAN 对车辆行驶的轨迹数据中的停止点进行分析。我们的轨迹数据格式为 (时间,  $X$  坐标,  $Y$  坐标, 速度, 加速度)，由于轨迹数据中的停止点，或者是速度比较低的点都具有一定的聚集性，而在运动中采集的轨迹，相对的具有一定的稀疏性，并且轨迹数据也没有固定的形状，这就使得 DBSCAN 中算法非常适合于轨迹数据的停止点或是速度比较低的点的聚类。这里我们采用标准的 DBSCAN 算法对停止点进行聚类，这里只使用到了轨迹数据的坐标信息，但是改进的 DBSCAN 算法能够将时间信息也考虑到聚类算法中去，详情请参考文献 [3]，作者定义新的核心点为“前后轨迹点平均速度低于某一范围的点”，更加好的描述了轨迹数据停止点的一些性质。

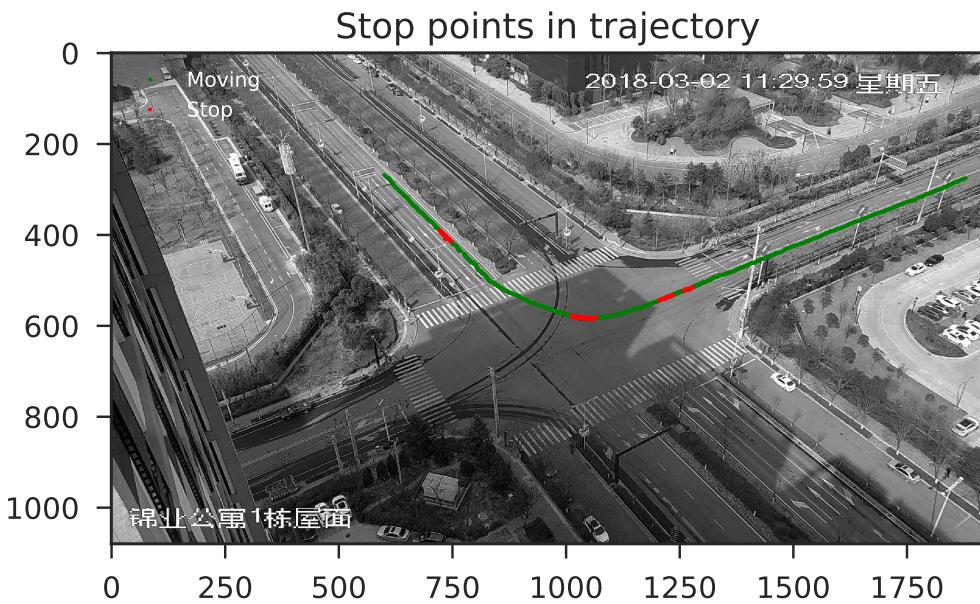


图 3: 轨迹停止点聚类结果，其中红色点代表停止点

从图中，我们可以看到，DBSCAN 聚类算法可以准确的找到停止点，也就是红色点。其中绿色的轨迹点代表原始 DBSCAN 聚类算法中定义的“噪声”点，因为这些点在  $Eps$  邻域内， $MinPts$  的条件不满足，既不是核心点，并且也没有其他点与该点密度直达，所以被判定为噪声点。

## 4.2 DBSCAN 对于几个特殊数据集进行聚类

我们这里给出几个特殊数据集，来说明 DBSCAN 这种算法的优秀效果。由图我们可以看出，我们不需要人为的选择聚类的可以比较好的识别出平面上的所包含的聚类的数目，尤其是对于 (c) 图中的数据集，从三个高斯分布中生成而来，(c) 图的数据集可以准确的识别出和高斯分布中离“中心”比较远的噪声点。

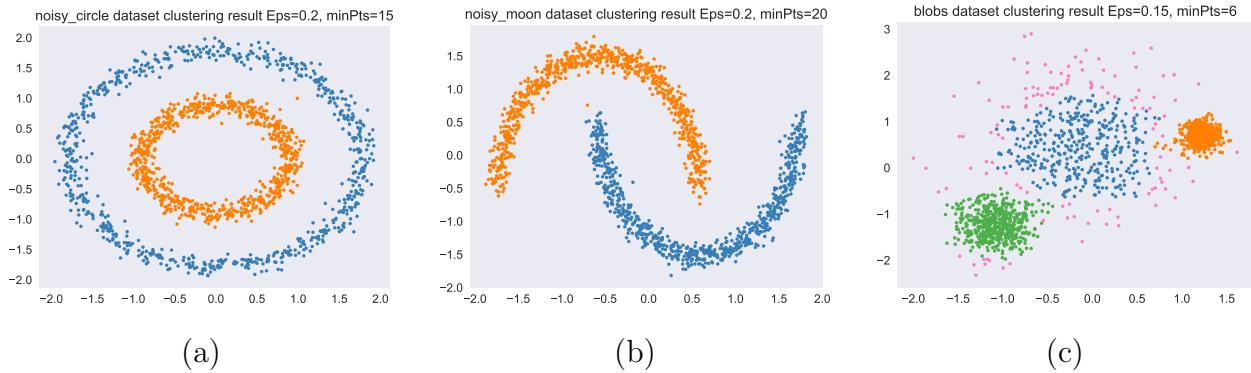


图 4: (a)circle 数据集聚类结果 (b)moon 数据集聚类结果 (c)blobs 数据集聚类结果

## 5 总结

DBSCAN 聚类算法可以在数据带有“噪声”的前提之下发现任意形状的聚类，并且聚类速度快。但是由于它对于数据集聚类的时候，用到了全局化的参数，也就是说对于每一个聚类都采用同样的  $Eps$  和  $MinPts$  值，导致在对待密度不均一的数据时导致聚类的质量会变差，例如 (c) 中聚类所示，若是  $Eps$  参数和  $MinPts$  参数设置不当，很容易橙色的聚类会与蓝色的聚类合并，因为其之间会有少量的密度连通的点作为“桥梁”。但是这并不影响 DBSCAN 成为一种优秀的聚类算法。

## 参考文献

- [1] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.
- [2] K Mahesh Kumar and A Rama Mohan Reddy. A fast dbSCAN clustering algorithm by accelerating neighbor searching using groups method. *Pattern Recognition*, 58:39–48, 2016.
- [3] Andrey Tietbohl Palma, Vania Bogorny, Bart Kuijpers, and Luis Otavio Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 863–868. ACM, 2008.