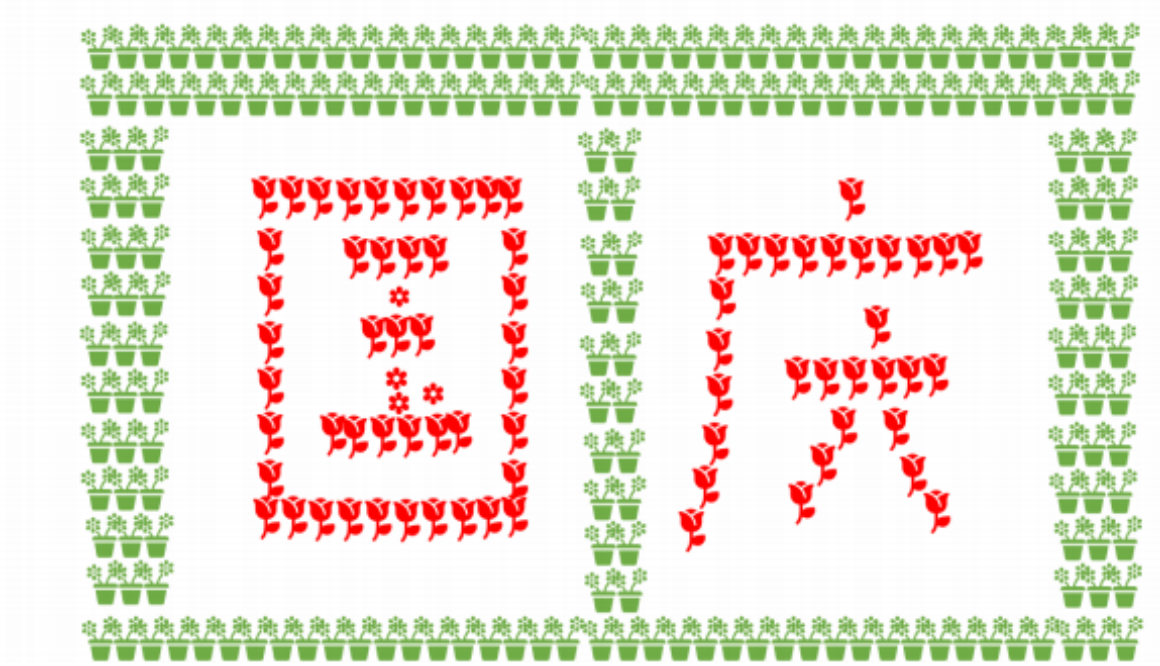
上一节课我讲解了 K-means 算法，那是一种基于划分的方法。今天我要介绍一种基于密度的聚类算法：DBSCAN（Density-Based Spatial Clustering of Applications with Noise），依旧先来看一个例子。

一个例子

想象有一个很大的广场，上面种了很多的鲜花和绿草。快要到国庆节了，园丁要把上面的鲜花和绿草打造成四个字：欢度国庆。于是园丁开始动手，用绿草作为背景去填补空白的区域，用红色的鲜花摆成文字的形状，鲜花和绿草之间都要留下至少一米的空隙，让文字看起来更加醒目。

国庆节过后，园丁让他的大侄子把这些花和草收起来运回仓库，可是大侄子是红绿色盲，不能通过颜色来判断，这些绿草和鲜花的面积又非常大，没有办法画出一个区域来告知大侄子。这可怎么办呢？

想来想去，园丁一拍脑袋跟大侄子说：“你就从一个位置开始收，只要跟它连着的距离在一米以内的，你就摞在一起；如果是一米以外的，你就再重新放一堆。”大侄子得令，开开心心地去收拾花盆了。最后呢，大侄子一共整理了三堆花盆：所有的绿草盆都摞在一起，“国”字用的红花摞在一起，“庆”字用的红花摞在了一起。这就是一个关于密度聚类的例子了。



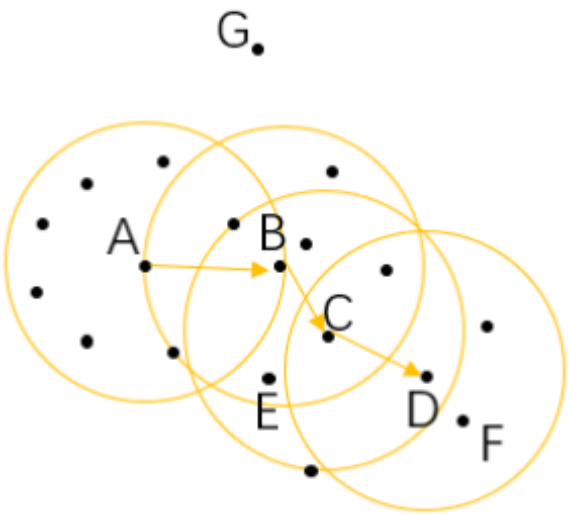
算法原理

上面的例子看起来比较简单，但是在算法的处理上我们首先有个问题要处理，那就是如何去衡量密度。在 DBSCAN 中，衡量密度主要使用两个指标，即半径和最少样本量。

对于一个已知的点，以它为中心，以给定的半径画一个圆，落在这个圆内的就是与当前点比较紧密的点；而如果在这个圆内的点达到一定的数量，即达到最少样本量，就可以认为这个区域是比较稠密的。

在算法的开始，要给出半径和最少样本量，然后对所有的数据进行初始化，如果一个样本符合在它的半径区域内存在大于最少样本量的样本，那么这个样本就被标记为核心对象。

这里我画了一幅图，假设我们的最小样本量为 6，那么这里面的 A、B、C 为三个核心对象。



对于在整个样本空间中的样本，可以存在下面几种关系：

直接密度可达： 如果一个点在核心对象的半径区域内，那么这个点和核心对象称为直接密度可达，比如上图中的 A 和 B 、B 和 C 等。

密度可达： 如果有一系列的点，都满足上一个点到这个点是密度直达，那么这个系列中不相邻的点就称为密度可达，比如 A 和 D。

密度相连： 如果通过一个核心对象出发，得到两个密度可达的点，那么这两个点称为密度相连，比如这里的 E 和 F 点就是密度相连。

一口气介绍了这么多概念，其实对照着图片都很好理解的，我们再来看 DBSCAN 接下来的处理步骤。

经过了初始化之后，再从整个样本集中去抽取样本点，如果这个样本点是核心对象，那么从这个点出发，找到所有密度可达的对象，构成一个簇。

如果这个样本点不是核心对象，那么再重新寻找下一个点。

不断地重复这个过程，直到所有的点都被处理过。

这个时候，我们的样本点就会连成一片，也就变成一个一个的连通区域，其中的每一个区域就是我们所获得的一个聚类结果。

当然，在结果中也有可能存在像 G 一样的点，游离于其他的簇，这样的点称为异常点。

DBSCAN 的原理你只是看字面解释的话可能会有点迷惑，最好结合图片来进行理解，自己手动画一下图，来分析一下上面的几种概念，应该就比较容易理解了。接下来我们看看它都有哪些优缺点。

算法优缺点

优点

- **不需要划分个数。** 跟 K-means 比起来，DBSCAN 不需要人为地制定划分的类别个数，而可以通过计算过程自动分出。
- **可以处理噪声点。** 经过 DBSCAN 的计算，那些距离较远的数据不会被记入到任何一个簇中，从而成为噪声点，这个特色也可以用来寻找异常点。
- **可以处理任意形状的空间聚类问题。** 从我们的例子就可以看出来，与 K-means 不同，DBSCAN 可以处理各种奇怪的形状，只要这些数据够稠密就可以了。

缺点

- **需要指定最小样本量和半径两个参数。** 这对于开发人员极其困难，要对数据非常了解并进行很好的数据分析。而且根据整个算法的过程可以看出，DBSCAN 对这两个参数十分敏感，如果这两个参数设定得不准确，最终的效果也会受到很大的影响。
- **数据量大时开销也很大。** 在计算过程中，需要对每个簇的关系进行管理。所以当数据量大的话，内存的消耗也非常严重。
- **如果样本集的密度不均匀、聚类间距差相差很大时，聚类质量较差。**

关于算法的优缺点就先介绍这么多，在使用的过程中十分要注意的就是最小样本量和半径这两个参数，最好预先对数据进行一些分析，来加强我们的判断。下面我们进入到动手环节，用代码来实现 DBSCAN 的使用。

尝试动手

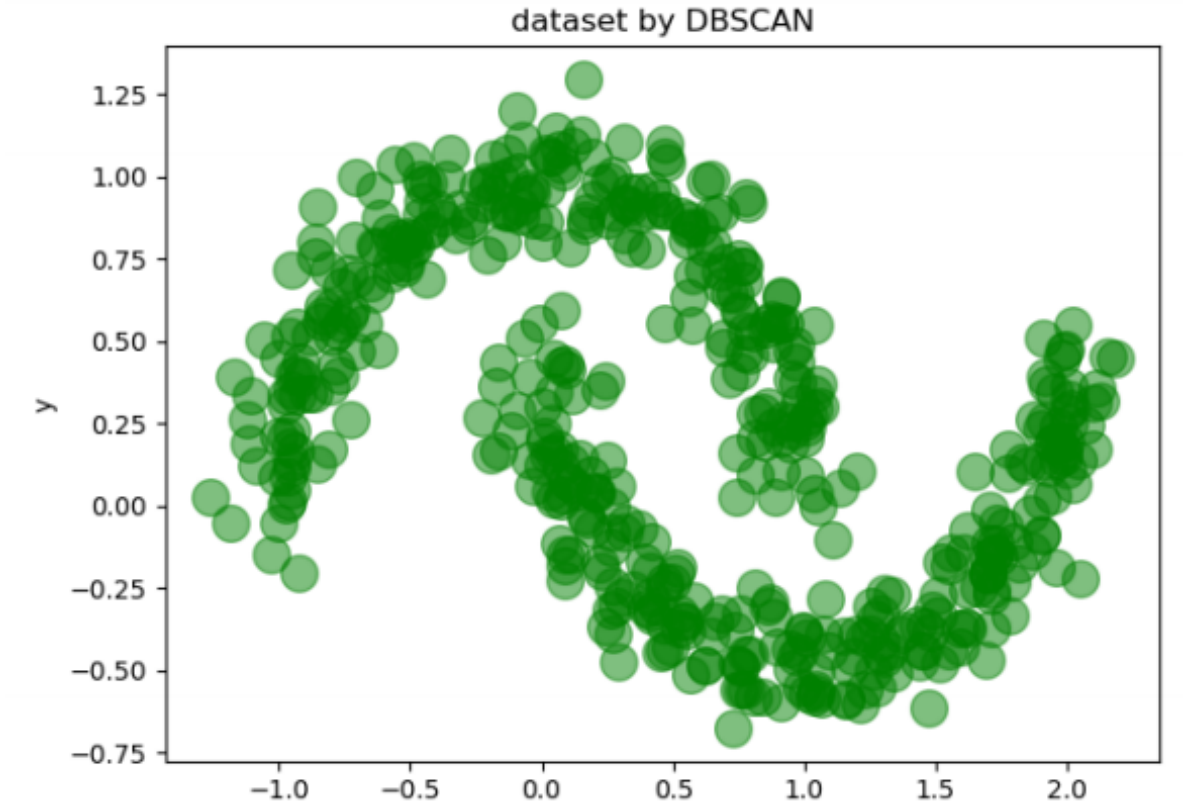
今天我们使用的数据集不再是鸢尾花数据集，我们要使用 datasets 的另外一个生成数据的功能。

在下面的代码中可以看到，我调用了 make_moons 这个方法，在 sklearn 的官网上，我们可以看到关于这个方法的介绍：生成两个交错的半圆环，从下面的生成图像我们也能够看到，这里生成的数据结果，是两个绿色的半圆形。

此外，我们今天调用的聚类方法是 sklearn.cluster 中的 dbscan。

```
from sklearn import datasets


import pandas as pd


import matplotlib.pyplot as plt


from sklearn.cluster import dbscan #今天使用的新算法包


#生成500个点 噪声为0.1


X, _ = datasets.make_moons(500, noise=0.1, random_state=1)


df = pd.DataFrame(X, columns=['x', 'y'])


df.plot.scatter('x','y', s = 200,alpha = 0.5, c = "green", title = 'dataset by
DBSCAN')


plt.show()
```

使用上面的方法，会生成一份数据，我们最后还调用了 plot 方法把数据绘制出来，就是下图所显示的样子，就像两个弯弯的月亮互相缠绕在一起。

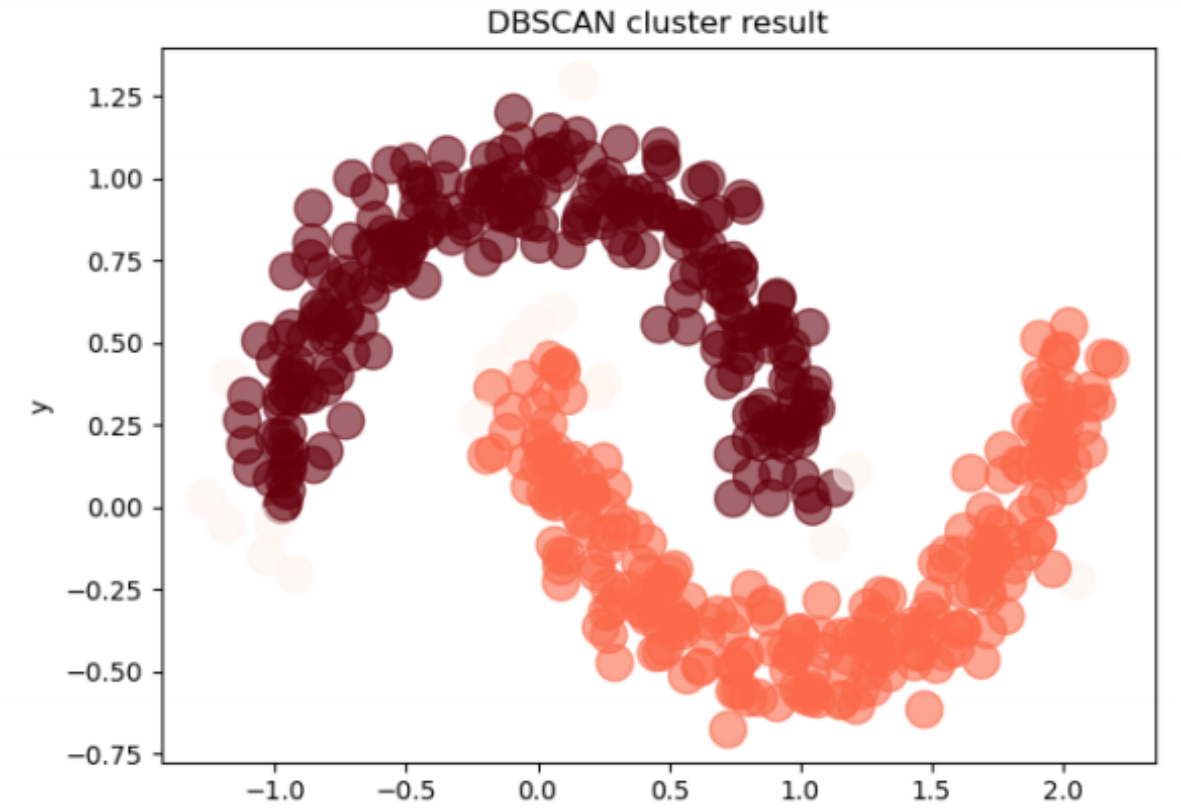


接下来，我们就开始使用 dbscan 算法来进行聚类运算。可以看到我为 dbscan 算法配置了初始的邻域半径和最少样本量。

```
# eps为邻域半径，min_samples为最少样本量

core_samples, cluster_ids = dbscan(X, eps=0.2, min_samples=20)

# cluster_ids中-1表示对应的点为噪声

df = pd.DataFrame(np.c_[X, cluster_ids], columns=['x', 'y', 'cluster_id'])

df['cluster_id'] = df['cluster_id'].astype('i2')

#绘制结果图像

df.plot.scatter('x', 'y', s=200,

c=list(df['cluster_id']), cmap='Reds', colorbar=False,

alpha=0.6, title='DBSCAN cluster result')

plt.show()
```

最后，我们使用不同的颜色来标识聚类的结果，从图上可以看出有两个大类，也就是两个月亮的形状被聚类算法算了出来。

但是眼尖的同学可能看到，在月亮两头的区域有一些非常浅色的点，跟两个类别的颜色都不一样，这里就是最后产生的噪声点，根据我们设置的参数计算，这些点不属于任何一个类别。



总结

完成了动手环节，这节课的主要内容就介绍完了。这节课我们学习了聚类算法的第二个方法 “DBSCAN” 算法。它是基于密度的聚类方法，与前面讲的 K-means 不同的是，它可以很好地解决数据形状不规则的情况。

在算法原理环节，有几个概念需要你仔细去理解，只要明白了那几个概念，DBSCAN 算法的核心也就可以掌握了。

总体来讲，DBSCAN 是一个比较简单明了的算法，没有太多复杂的数学运算，但是在实践中要想用好 DBSCAN 却不是十分容易，这主要是因为两个初始化参数比较难以设定，对于新手来说可能会有些困难，但是不要怕，你终会成长为一名有经验的数据挖掘工程师的。