1.什么叫DBSCAN聚类

DBSCAN(Density-Based Spatial Clustering of Applications with Noise，具有噪声的基于密度的聚类方法)是一种很典型的密度聚类算法。

2.密度聚类原理简述

DBSCAN是一种基于密度的聚类算法，这类密度聚类算法一般假定类别可以通过样本分布的紧密程度决定。同一类别的样本，他们之间是紧密相连的，也就是说，在该类别任意样本周围不远处一定有同类别的样本存在。

通过将紧密相连的样本化为一类，这样就得到了一个聚类类别。通过将所有各组紧密相连的样本化为各个不同的类别，则
我们就得到了最终的所有聚类类别结果。

3.DBSCAN密度定义

DBSCAN是基于一组邻域来描述样本集的紧密程度的，参数(ϵ, MinPts）用来描述邻域的样本分布紧密程度，其中ϵ描述了某一样本的邻域距离阈值，MInPts描述了某一样本距离为ϵ的邻域中样本个数的阈值。

假设我的样本集是D=(x1,x2,...,xm),则DBSCAN具体的密度描述定义如下：
1） ϵ-邻域：对于xj∈D，其ϵ-邻域包含样本集D中与xj的距离不大于ϵ的子样本集，即Nϵ(xj)={xi∈D|distance(xi,xj)≤ϵ}, 这个子样本集的个数记为|Nϵ(xj)|
通俗解释：对样本集中每一个点，以ϵ为半径画圆，计算圆里面包含的样本数，最少为1。

2) 核心对象：对于任一样本xj∈D，如果其ϵ-邻域对应的Nϵ(xj)至少包含MinPts个样本，即如果|Nϵ(xj)|≥MinPts，则xj是核心对象。
通俗解释：做个限定，如果圆内点的个数达到我们要求的最低限制数，则这个圆的中心点的样本为核心对象。

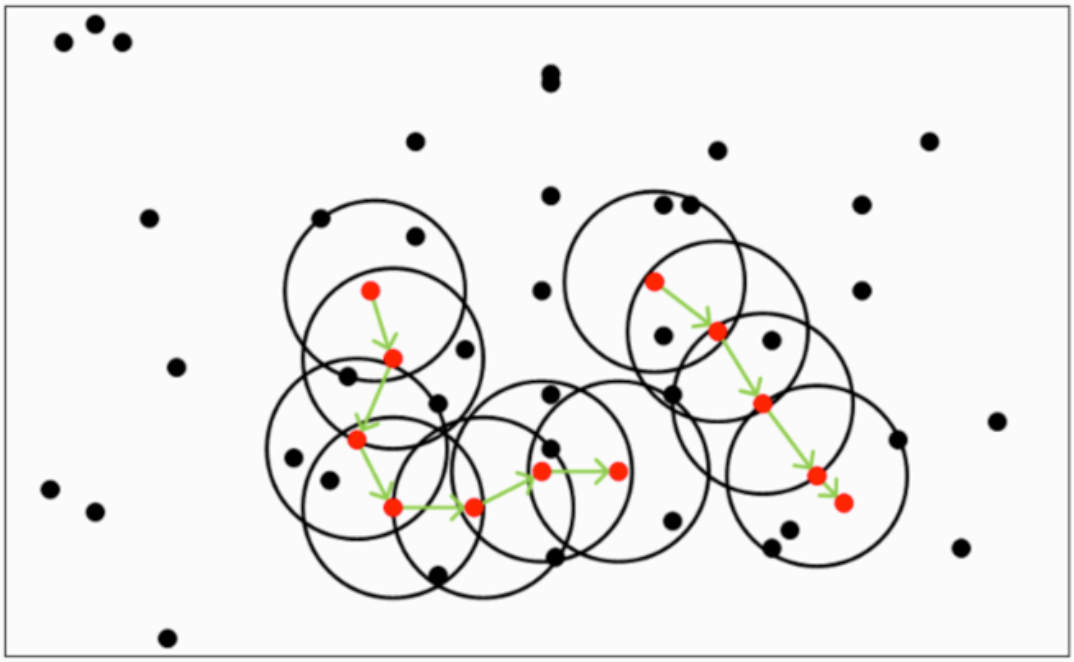
3）密度直达：如果xi位于xj的ϵ-邻域中，且xj是核心对象，则称xi由xj密度直达。注意反之不一定成立，即此时不能说xj由xi密度直达, 除非且xi也是核心对象。（没理解明白）
通俗解释：以核心对象作为中心，ϵ为半径画的圆，这个圆内包含的其他样本点，

都算做由该核心对象的密度直达点

4）密度可达：对于xi和xj,如果存在样本序列p1,p2,...,pT,满足p1=xi,pT=xj, 且pt+1由pt密度直达，则称xj由xi密度可达。也就是说，密度可达满足传递性。此时序列中的传递样本p1,p2,...,pT−1均为核心对象，因为只有核心对象才能使其他样本密度直达。注意密度可达也不满足对称性，这个可以由密度直达的不对称性得出。**通俗解释**：注意这里的pt中的t可以为1，2，，，n中的任意值，p2由p1密度直达，p3由p2密度直达，依次类推，可知p1,p2,...,pT−1均为核心对象，这些序列内的核心对象是密度可达的

5）密度相连：对于xi和xj,如果存在核心对象样本xk，使xi和xj均由xk密度可达，则称xi和xj密度相连。注意密度相连关系是满足对称性的。

从下图可以很容易看出理解上述定义，图中MinPts=5，红色的点都是核心对象，因为其ϵ-邻域至少有5个样本。黑色的样本是非核心对象。所有核心对象密度直达的样本在以红色核心对象为中心的超球体内，如果不在超球体内，则不能密度直达。图中用绿色箭头连起来的核心对象组成了密度可达的样本序列。在这些密度可达的样本序列的ϵ-邻域内所有的样本相互都是密度相连的。



4、DBSCAN密度聚类思想

DBSCAN的聚类定义很简单：由密度可达关系导出的最大密度相连的样本集合，即为我们最终聚类的一个类别，或者说一个簇。

这个DBSCAN的簇里面可以有一个或者多个核心对象。如果只有一个核心对象，则簇里其他的非核心对象样本都在这个核心对象的ϵ-邻域里；如果有多个核心对象，则簇里的任意一个核心对象的ϵ-邻域中一定有一个其他的核心对象，否则这两个核心对象无法密度可达。这些核心对象的ϵ-邻域里所有的样本的集合组成的一个DBSCAN聚类簇。

那么怎么才能找到这样的簇样本集合呢？DBSCAN使用的方法很简单，它任意选择一个没有类别的核心对象作为种子，然后找到所有这个核心对象能够密度可达的样本集合，即为一个聚类簇。接着继续选择另一个没有类别的核心对象去寻找密度可达的样本集合，这样就得到另一个聚类簇。一直运行到所有核心对象都有类别为止。

基本上这就是DBSCAN算法的主要内容了，但是我们还是有三个问题没有考虑。

第一个是一些异常样本点或者说少量游离于簇外的样本点，这些点不在任何一个核心对象在周围，在DBSCAN中，我们一般将这些样本点标记为噪音点。

第二个是距离的度量问题，即如何计算某样本和核心对象样本的距离。在DBSCAN中，一般采用最近邻思想，采用某一种距离度量来衡量样本距离，比如欧式距离。这和KNN分类算法的最近邻思想完全相同。对应少量的样本，寻找最近邻可以直接去计算所有样本的距离，如果样本量较大，则一般采用KD树或者球树来快速的搜索最近邻。

第三种问题比较特殊，某些样本可能到两个核心对象的距离都小于ϵ，但是这两个核心对象由于不是密度直达，又不属于同一个聚类簇，那么如果界定这个样本的类别呢？一般来说，此时DBSCAN采用先来后到，先进行聚类的类别簇会标记这个样本为它的类别。也就是说BDSCAN的算法不是完全稳定的算法。

5、DBSCAN算法流程

下面我们对DBSCAN聚类算法的流程做一个总结。

输入：样本集D=(x1,x2,...,xm)，邻域参数(ϵ,MinPts), 样本距离度量方式

输出： 簇划分C.

1）初始化核心对象集合Ω=∅, 初始化聚类簇数k=0，初始化未访问样本集合Γ = D,

簇划分C = ∅

2) 对于j=1,2,...m, 按下面的步骤找出所有的核心对象：

a) 通过距离度量方式，找到样本xj的ϵ-邻域子样本集Nϵ(xj)

b) 如果子样本集样本个数满足|Nϵ(xj)|≥MinPts， 将样本xj加入核心对象样本集合：Ω=Ω∪{xj}

3）如果核心对象集合Ω=∅，则算法结束，否则转入步骤4.

4）在核心对象集合Ω中，随机选择一个核心对象o，初始化当前簇核心对象队列Ωcur={o}, 初始化类别序号k=k+1，初始化当前簇样本集合Ck={o}, 更新未访问样本集合Γ=Γ−{o}

5）如果当前簇核心对象队列Ωcur=∅，则当前聚类簇Ck生成完毕, 更新簇划分C={C1,C2,...,Ck}, 更新核心对象集合Ω=Ω−Ck， 转入步骤3。

6）在当前簇核心对象队列Ωcur中取出一个核心对象o′,通过邻域距离阈值ϵ找出所有的ϵ-邻域子样本集Nϵ(o′)，令Δ=Nϵ(o′)∩Γ, 更新当前簇样本集合Ck=Ck∪Δ, 更新未访问样本集合Γ=Γ−Δ，更新Ωcur=Ωcur∪(Nϵ(o′)∩Ω)，转入步骤5.

输出结果为： 簇划分C={C1,C2,...,Ck}

6、DBSCAN小结

和传统的K-Means算法相比，DBSCAN最大的不同就是不需要输入类别数k，当然它最大的优势是可以发现任意形状的聚类簇，而不是像K-Means，一般仅仅使用于凸的样本集聚类。同时它在聚类的同时还可以找出异常点，这点和BIRCH算法类似。

那么我们什么时候需要用DBSCAN来聚类呢？一般来说，如果数据集是稠密的，并且数据集不是凸的，那么用DBSCAN会比K-Means聚类效果好很多。如果数据集不是稠密的，则不推荐用DBSCAN来聚类。

下面对DBSCAN算法的优缺点做一个总结。

DBSCAN的主要优点有：

1） 可以对任意形状的稠密数据集进行聚类，相对的，K-Means之类的聚类算法一般只适用于凸数据集。

2） 可以在聚类的同时发现异常点，对数据集中的异常点不敏感。

3） 聚类结果没有偏倚，相对的，K-Means之类的聚类算法初始值对聚类结果有很大影响。

DBSCAN的主要缺点有：

1）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，这时用

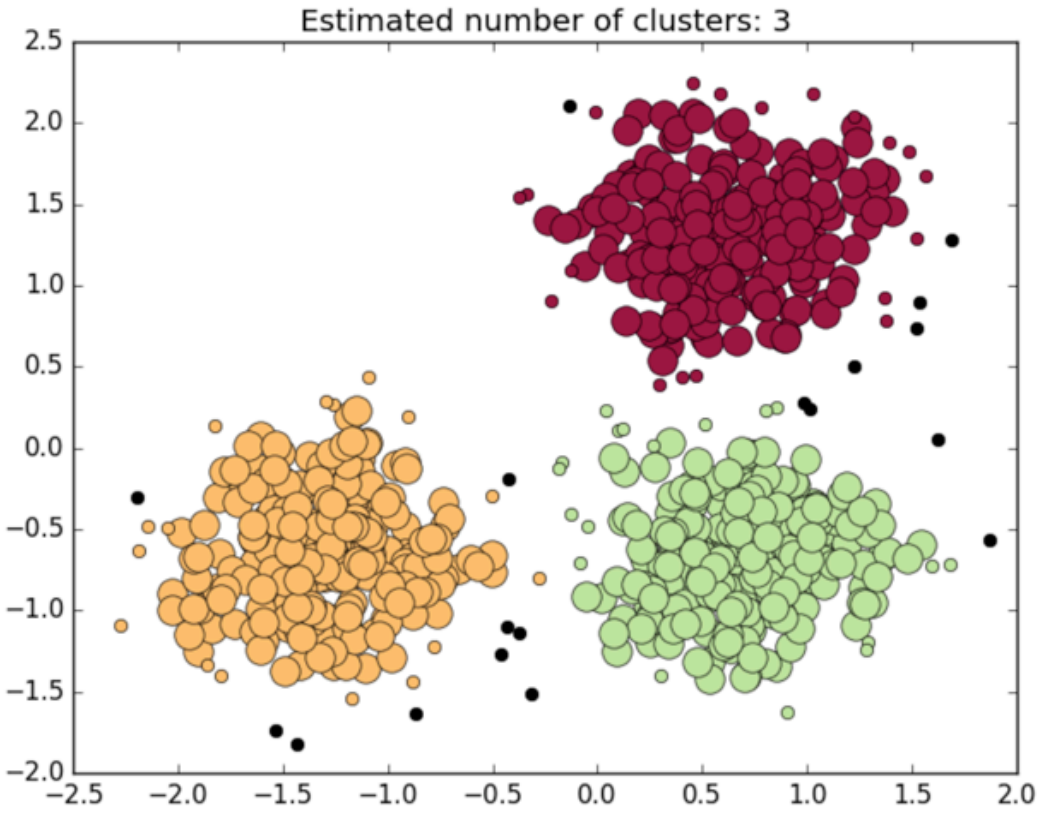
DBSCAN聚类一般不适合。

2） 如果样本集较大时，聚类收敛时间较长，此时可以对搜索最近邻时建立的KD树或者球树进行规模限制来改进。

3） 调参相对于传统的K-Means之类的聚类算法稍复杂，主要需要对距离阈值ϵ，邻域样本数阈值MinPts联合调参，不同的参数组合对最后的聚类效果有较大影响。

7、如何理解ϵ值和MinPts阈值

为什么DBSCAN设置两个参数ϵ和MinPts就能有效地分类

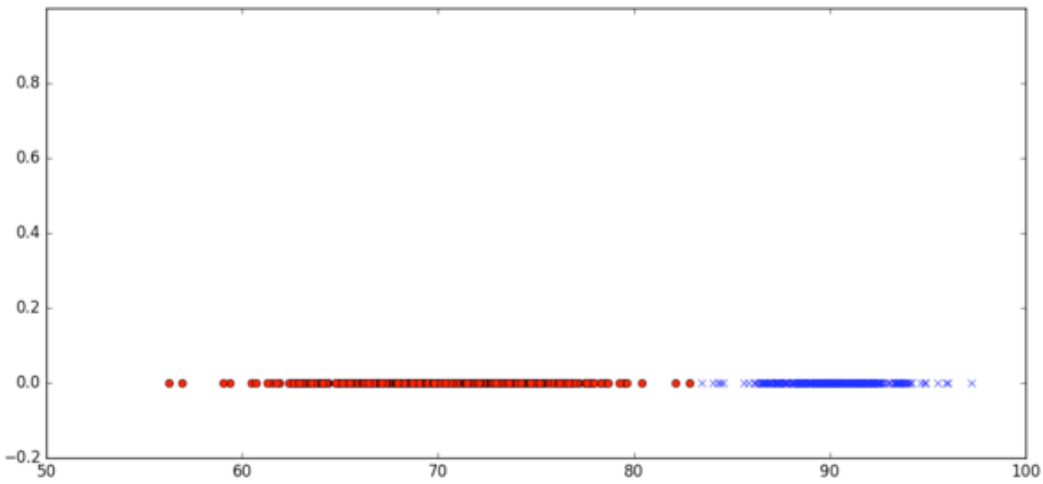


首先思考一个问题，数据为什么会呈现聚类这种奇怪的现象？这其实是统计学中的一个概念，即基于大数据背后，每个数据样本都有它们的不确定性。聚类需要满足两个条件：第一，类与类之间有一个影响它们本质的gap，也就是说在2维世界中两堆数据是有一定的间距的。影响间距的本质原因，便是现实世界中影响人们区分这两类事物的判别依据，只是在数据的坐标轴世界中我们以distance来衡量；第二，每个类在自己的群体内，有某种微小的不确定性影响着它们的分布，但不足以跨越那个gap。即这些微小的不确定性，导致了数据样本在某个特定的样

本空间附近随机出现，从而在坐标轴中能够看到大量数据堆积在一块。

现在，我们来考虑一个实际的问题来理解什么是聚类。

范例：某两所高校为了迎战高考，准备一次联合模拟考，现在我们已知了两所学校各学生的数学成绩，如下图所示。现在我们希望能够根据这些数据来区分这些学生都属于哪所高校，假设两高校学生的平均水平未知。



这里面，我们已经对数据样本做了区分。红色点代表A校学生的分数分布情况，蓝色点代表B校学生的分数分布情况。那么图中的信息告诉我们什么样的事实?

很明显，数据出现了某种群聚现象。A校学生的平均分集中在70分，而B校学生的成绩集中在90分，并且分别向两边扩散，且分布密度逐渐递减。在83分这个边界上两者有所交叉。我们在统计成绩时，往往一个群体的成绩分布情况就呈现于类似的分布，由大量数据统计得，其背后是符合高斯模型的。（难道不能是其他分布？）这就是所谓的“物以类聚，人以群分”。如A校，B校问题，在历史进程中，A校，B校在招人时，就有一种自然规则，中考成绩优异的进入B校，而中考稍差的进入A校。这种历史因素我们虽然观测不到，却着实影响了未来成绩的分布情况。

现在，我们从考试这个例子中解释为何会出现gap，以及gap是如何帮助我们区分A校B校的学生。

一张理想的数学卷子，它的难易程度是不一样的，为了方便阐述问题，我们假设一张数学卷子有80道易答题，每道1分。一道10分的中等难度题，和10道1分的有难度题。将这张数学卷子给A,B校考生做后，得到了如下的数据分布。问，哪些学生是A校，哪些学生是B校?

我们知道B校在历史上的学生是优于A校的学生，所以我们可以简单认为，对于80分题目他们回答的正确率为100%，大题10分准确率80%，10道有难度的题该

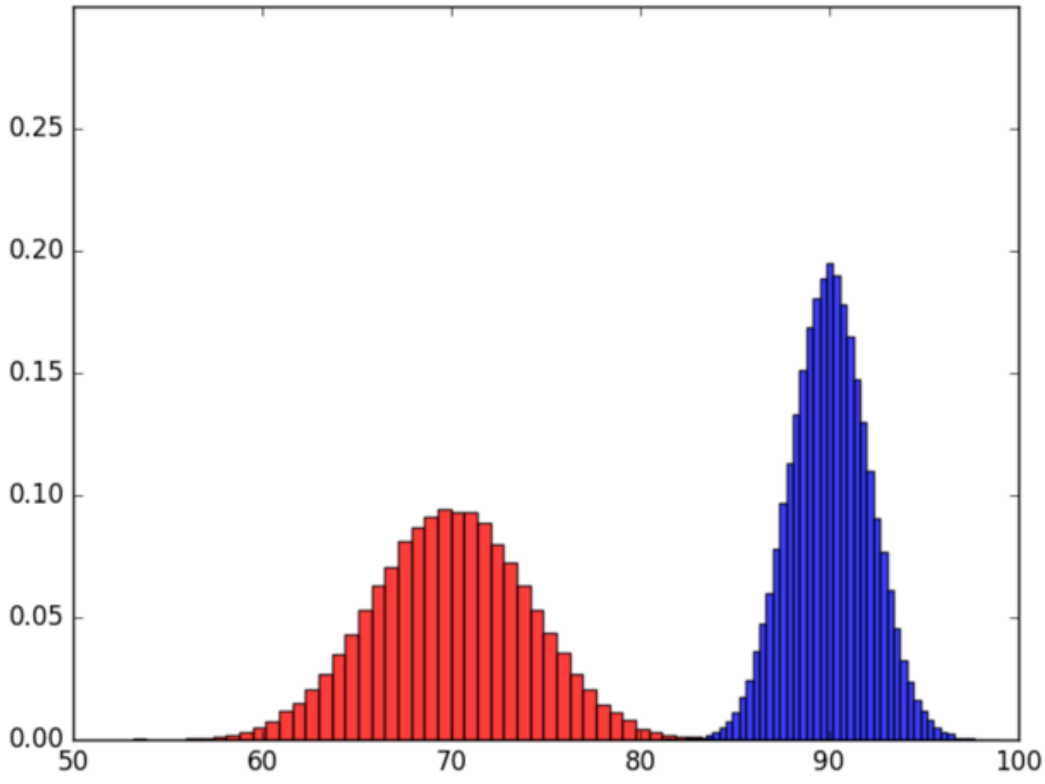
校学生的平均回答准确率在50%。而A校学生，80分题目的回答的正确率为80%，大题10分正确率为10%，10道有难度题回答准确率在1%。当然上述假定是没有任何依据的，这需要真正的去统计该学校每一道题的回答准确率才能得出准确的数字，这里是为了方便解释背后的概率模型。

对于B校的学生，80分的题目对他们来说都不是事，因此绝大多数的学生成绩必然高于80分，且对于中等难度的题，大部分学生依然能够完成。所以成绩都80朝上，甚至90朝上。可A校的学生，中等难度的10分题是他们的一个障碍，以他们的水平几乎没人能够完成这道题，由此他们的成绩只能80分朝下。这10分便是数据的gap，天然的分割了A校和B校的学生。

但聚类的另外一个特征是大家都会聚集在一块，这又是什么原因造成的呢？前文讲了一种不确定性，不确定性来源于概率统计的不确定性。拿A校学生的数学成绩分布来讲（B校依此类推），80分的题，他们准确率为80%，那么从概率上来说，全部答对的可能性是相当小的。我们可以写出在80道题中，答对n道题的概率：

$$P(n|\theta = 0.8) = C_{80}^{n} 0.8^{n} 0.2^{80-n}$$

由此得到了，如下的伯努利概率分布函数：



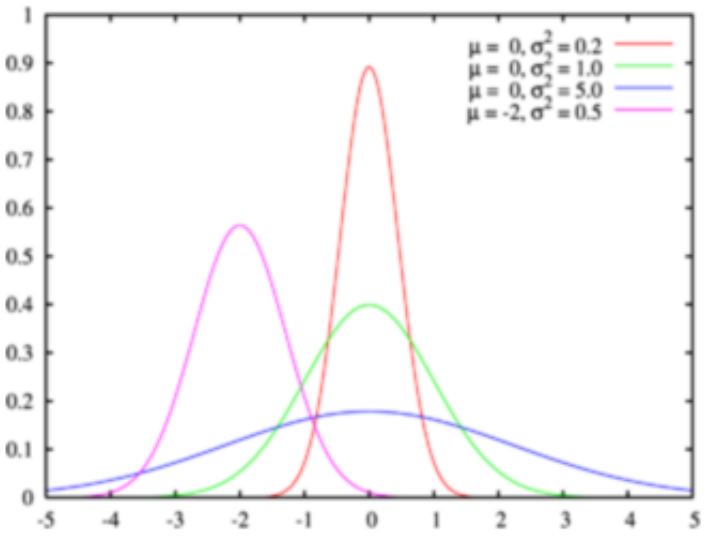
红色长条为A校学生分数出现的概率，蓝色长条为B校学生各分数出现的概率。当统计的样本很大时，伯努利概率分布函数将等价于高斯分布函数。

这张图很好的解释了类的群聚效应，虽然每个学生是独特的，但由于他们先天的资质和后天的努力在一个类群中相差不多，导致做数学题时，每个群体对于不同难度的题的回答准确率不同,可在群体内正确率是相近的。而又由于并非百分之百的回答准确率，这种不确定性导致了大量数据呈现中间高，两边低的分布形态。最终有了数据样本背后的模型大都是高斯分布模型。

其实说了那么多，只是为了解释聚类的两个本质特征，gap和群聚。在上图中，70分和90分附近群聚了大量数据，并分别向两边递减，而在83分左右附近，出现了一个gap。而DBSCAN就是利用了这两个特征构建出了一套算法。

我们来看看一般的高斯概率密度函数：

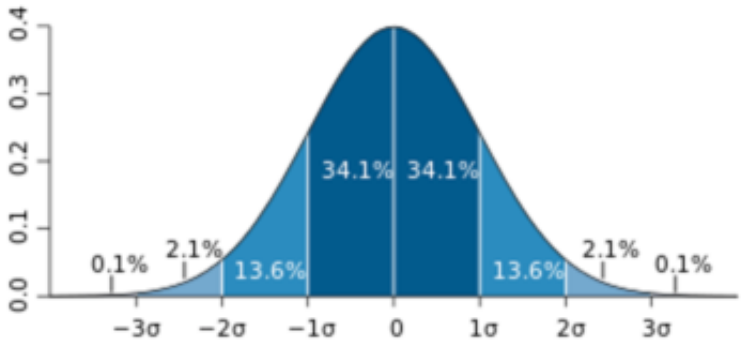
$$N(\mu, \delta^2) = \frac{1}{\delta\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\delta^2}}$$



还记得DBSCAN算法需要输入的两个参数嘛？ϵ和MinPts，我们逐一来解释下，ϵ本质上是一个核心点距离一个点的距离。在前述例子中，我们可以设置ϵ为几？显然设置距离1≤ϵ≤10比较合适，因为1是每道题最小的分值，小于1的邻域是不会有任何点存在的，那为什么要小于10呢，因为我们知道了区分两类群体的关键gap在于那道中等难度的题，它的分值为10。超过10分，那在83分处很容易就将这两类群体连接在一起了（由DBSCAN算法决定）。

接下来我们继续来分析MinPts参数为何能够实现数据样本的分类。由高斯概

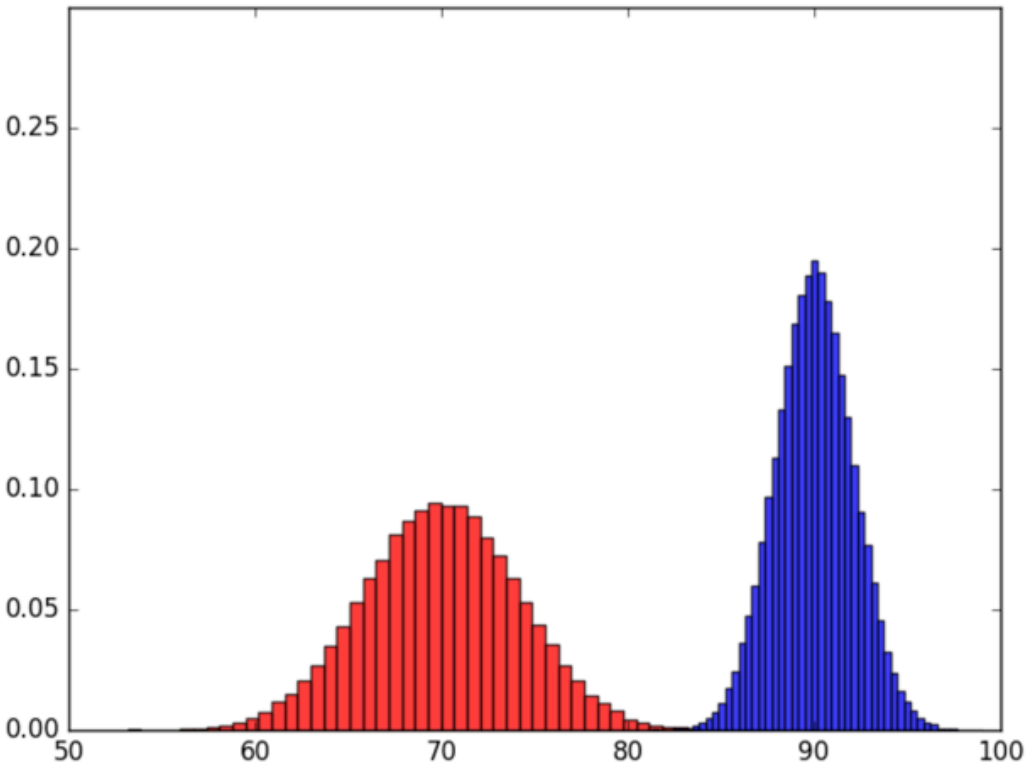
率分布函数我们容易得到，越靠近对称轴的位置，数据分布越密，即在ϵ邻域内的数据点越多，而越靠近两边，数据分布开始变得稀疏，数据点不断减小，如下图所示。



深蓝色区域是距平均值小于一个标准差之内的数值范围。在正态分布中，此范围所占比率为全部数值之68%。根据正态分布，两个标准差之内（蓝，棕）的比率合起来为95%。根据正态分布，三个标准差之内（深蓝，橙，黄）的比率合起来为99%。

在这里我们重新解释一波，在相同的ϵ范围内，你能看到数据样本分布的概率向两边逐渐递减，在−3δ和−2δ附近样本量只有总数的2.1%，因此我们可以设置MinPts为样本总数的2.1%，当小于这个值，便不再是我们的core points，而是outlier points。

那么深蓝和蓝色区域均为我们的核心点，也就是算法中密度相连的点，而一旦靠近高斯分布的底部，由于样本量小于一定数值，算法不再认为是核心点，转而区分了两类人群。



再看这图，结合MinPts和参数ε是不是豁然开朗了。

8、DBSCAN实战

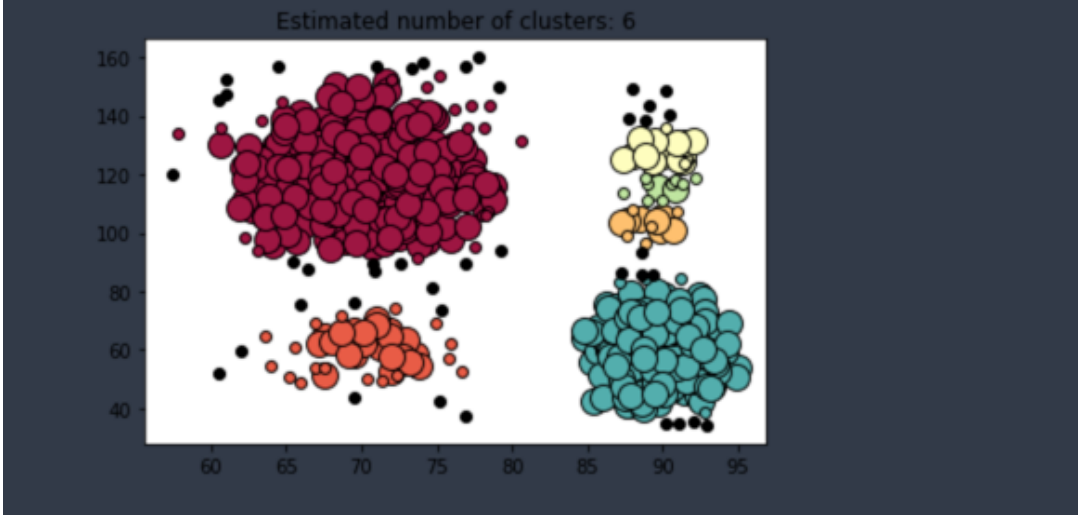
```python
import numpy as np
import matplotlib.pyplot as plt

# A校生
mu1, sigma1 = 70, 4.2
x1 = mu1 + sigma1 * np.random.randn(400)

iq_mu1,iq_sigma1 = 120,15 # 学习时间长
iq_mu2,iq_sigma2 = 60,10  # 学习时间短

tmp1 = iq_mu1 + iq_sigma1 * np.random.randn(350)
tmp2 = iq_mu2 + iq_sigma2 * np.random.randn(50)

y1 = np.append(tmp1,tmp2)

# B校生
mu2,sigma2 = 90,2.1
x2 = mu2 + sigma2 * np.random.randn(300)

tmp3 = iq_mu1 + iq_sigma1 * np.random.randn(50)
tmp4 = iq_mu2 + iq_sigma2 * np.random.randn(250)

y2 = np.append(tmp3,tmp4)

# 水平组合
dataSet1 = np.column_stack((x1,y1))
dataSet2 = np.column_stack((x2,y2))

dataSet = np.vstack((dataSet1,dataSet2))
```

这里我们分别取e=5和MinPts =10，利用sklearn中的DBSCAN算法：

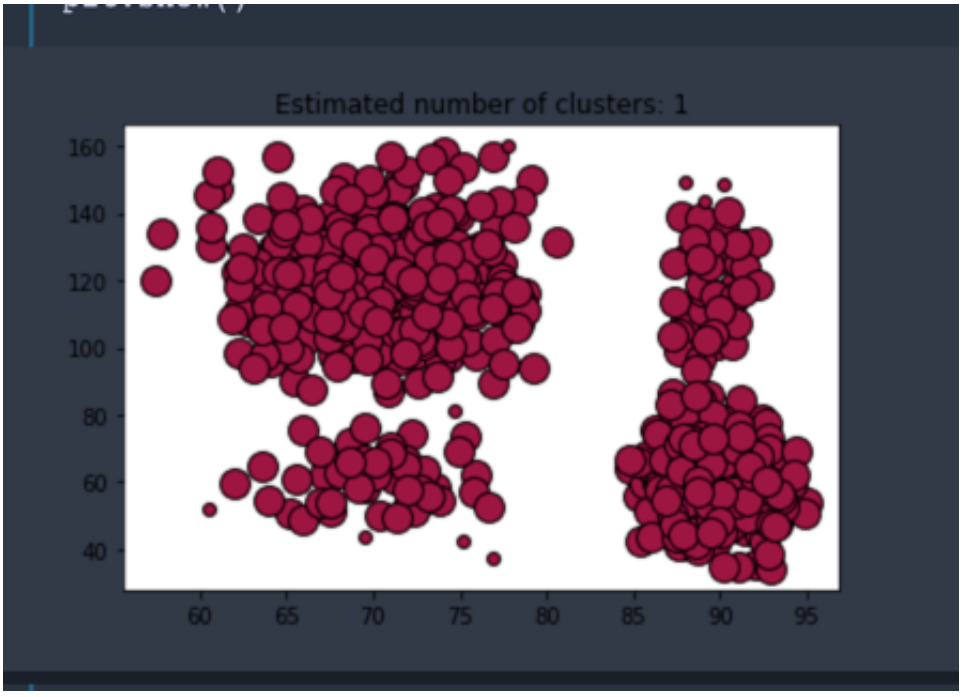
```
# DBSCAN算法
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets.samples_generator import make_blobs
from sklearn.preprocessing import StandardScaler

db = DBSCAN(eps=5, min_samples=10).fit(dataSet)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)


import matplotlib.pyplot as plt

# Black removed and is used for noise instead.
unique_labels = set(labels)
colors = plt.cm.Spectral(np.linspace(0, 1, len(unique_labels)))
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = 'k'

    class_member_mask = (labels == k)

    xy = dataSet[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
             markeredgecolor='k', markersize=14)

    xy = dataSet[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
             markeredgecolor='k', markersize=6)

plt.title('Estimated number of clusters: %d' % n_clusters_)
plt.show()
```

得到的分类图如下：

```
plt.title('Estimated number of clusters: %d' % n_clusters_)
plt.show()
```

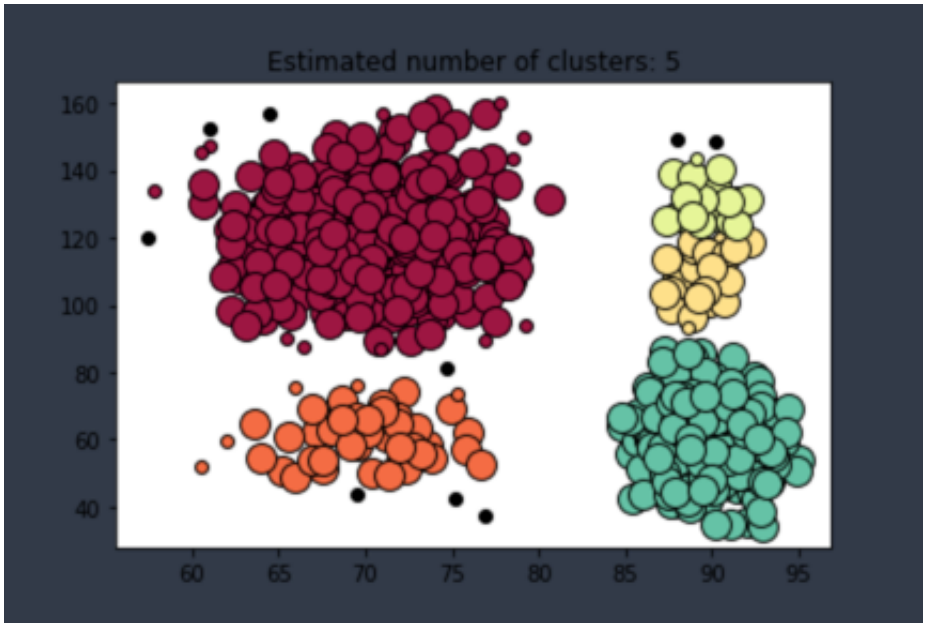


当我们改成ϵ=10和MinPts =10时，分类图如下：



显然ϵ跨越了gap，所以导致分类失败。

同样的当ϵ=5和MinPts =5，得到如下分类图：



把更多的样本划分到了类，显然效果要优于第一次，所以说，DBSCAN算法很依赖参数的设置，在特定领域参数的设置需要参考领域业务经验丰富的人来完成。