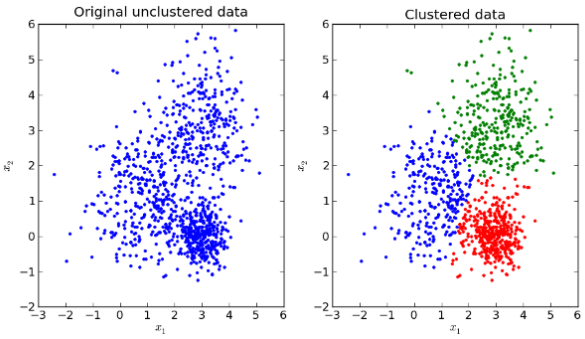
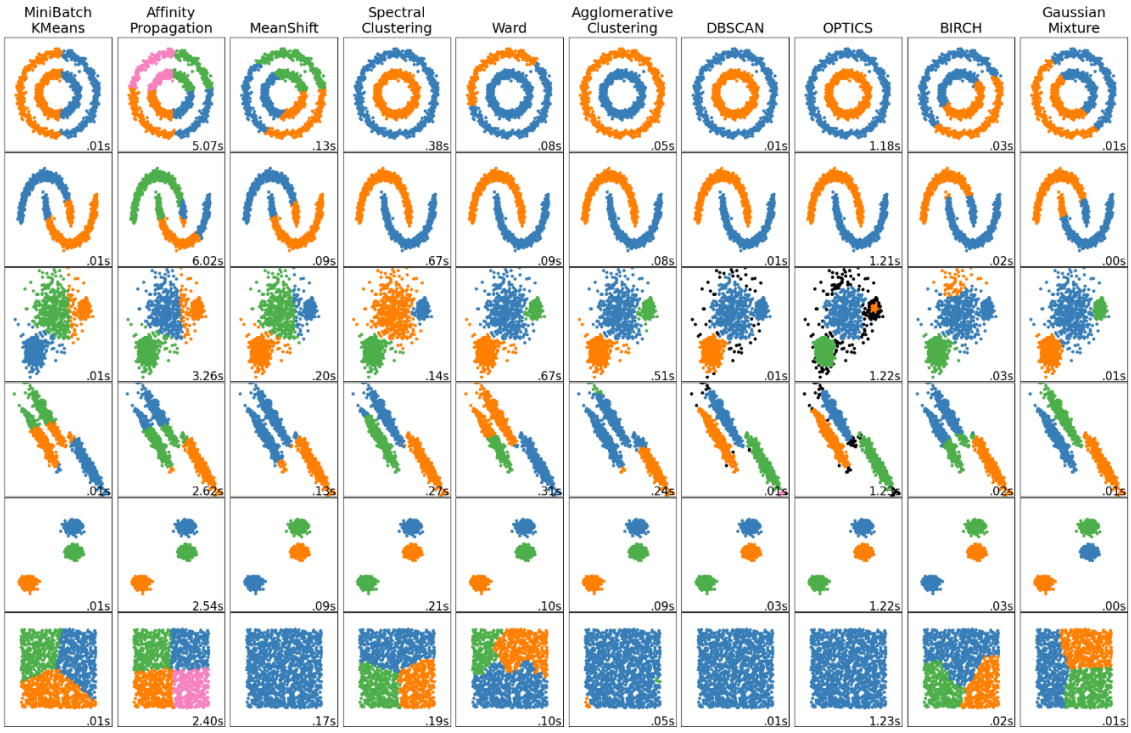
Clustering 聚类

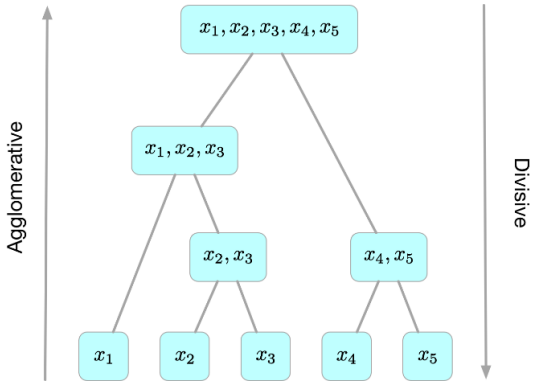


Clustering 是一个技巧来分类相似的 Data Points。每一个 Group 里的 Data Points 彼此都是比较相似。在 Clustering 里，准备的 Data 不需要给出 Label，因为 Clustering 是 Unsupervised Learning。

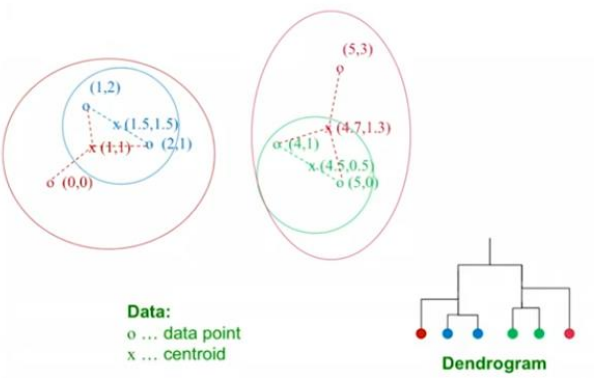


Hierarchical Clustering

在 Hierarchical Clustering 里，有 2 种



I. **Agglomerative (Bottom Up)**

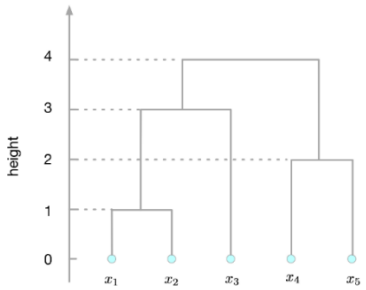


- 初始化时，每个 Data Point 自己都是一个 Cluster。计算过后，找出最靠近的 2 个点，然后把着 2 个点设为同一个 Cluster，之后求这个 Cluster 的中间值 (Centroid)，在继续寻找最靠近的 2 个点，然后再 Cluster 一起，再求这个 Cluster 的中心点，再继续循环下去。

II. **Divisive (Top Down)**

- 初始的时候，所有的 Data Points 都是同一个 Cluster，然后 Recursively 分开直到每个 Data Points 自己都是一个 Cluster。

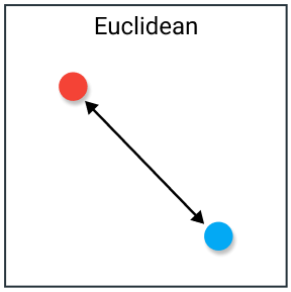
Dendrogram (树图形)



Dendrogram (树图形) 是用来知道 Cluster 的成果。从上图的 Dendrogram 可以看出 x_1 和 x_2 是最靠近的，因此合并成为同一个 Cluster (x_1, x_2)。训练完之后，只需要设置 h 的值，也就是横切这个 Dendrogram，就能的到不同数目的 Cluster。如果 $h = 2.5$，那么就会得到 3 个 Cluster，(x_1, x_2)，x_3，和 (x_4, x_5)。

Data Points 之间的距离计算

Euclidean Distance 欧氏距离



$$d(x,y) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$$

$$2\ Dimensions \rightarrow d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$3\ Dimensions \rightarrow d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

$$x\ \&\ y \rightarrow Different\ Point$$

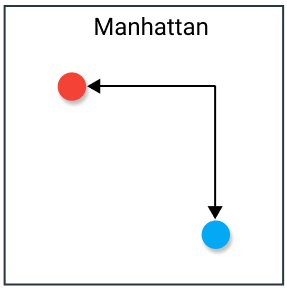
$$x = [x_1, x_2, \dots, x_d]^T$$

$$y = [y_1, y_2, \dots, y_d]^T$$

$$d \rightarrow Dimensional$$

Euclidean Distance 是计算两个点之间的直线距离

Manhattan Distance (L_1 Distance)



$$d(x,y) = \sum_{i=1}^{d}|x_i - y_i|$$

$$x\ \&\ y \rightarrow Different\ Point$$

$$x = [x_1, x_2, \dots, x_d]^T$$

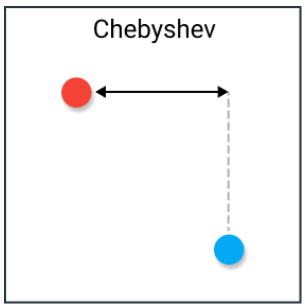
$$y = [y_1, y_2, \dots, y_d]^T$$

$$d \rightarrow Dimensional$$

Manhattan Distance 计算距离 Between two real-valued vectors。Manhattan Distance could only move right angles (No diagonal movement). 在高维的 Data 里，Manhattan Distance 没有 Euclidean Distance 来的好用。除此之外，在大距离的计算，它不能够求得最短距离。

Manhattan Distance 适合用在当 Dataset has discrete (离散的) and/or binary (二进制) attributes (属性)，比如 Chessboard。

Chebyshev Distance



$$d(x,y) = \max_{1 \le i \le d} |x_i - y_i|$$

$$x \,\&\, y \rightarrow Different\ Point$$

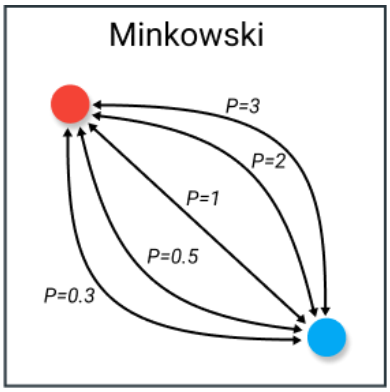
$$x = [x_1, x_2, \dots, x_d]^T$$

$$y = [y_1, y_2, \dots, y_d]^T$$

$$d \rightarrow Dimensional$$

Chebyshev Distance 只去最大值的那个 Axis 的值，Useful measure in games that allow unrestricted 8-way movement。Chebyshev Distance 不像 Euclidean Distance 还是 Cosine Similarity，使用 Chebyshev Distance 之前需要非常确认适合这个 Use-Case。

Minkowski Distance



$$d(x,y) = \left[\sum_{i=1}^{d}(x_i - y_i)^p\right]^{\frac{1}{p}}, p \geq 1$$

$$x \,\&\, y \rightarrow Different\ Point$$

$$x = [x_1, x_2, \dots, x_d]^T$$

$$y = [y_1, y_2, \dots, y_d]^T$$

$$d \rightarrow Dimensional$$

$$p = 1 \rightarrow Manhattan\ Distance$$

$$p = 2 \rightarrow Euclidean\ Distance$$

$$p = \infty \rightarrow Chebyshev\ Distance$$

Minkowski Distance 有很高的 Flexibility 在计算距离，设置不同的 p 值，能得到不同的效果。在计算 Minkowski Distance 的时候，有 3 个 Requirements

- Zero Vector – 从一个点移动到同一个点，这时候的值为 0。移动的值永远都是正数
- Scalar Factor – 当距离乘与一个正数，方向没变只是距离变了
- Triangle Inequality – 最短的距离是两个点的直线距离

使用 Minkowski Distance 的时候需要对设置 p 值非常了解，p 值没设对会导致计算结果不有效。

Cluster 之间的距离衡量

计算 2 个 Cluster 之间的距离，有 3 种方法

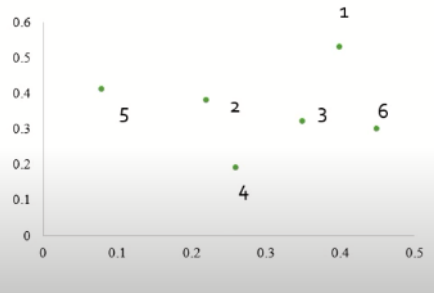
- Single Linkage – Single Linkage 是讲 2 个 Cluster 的 Data Points 里，距离最近的 2 个点作为两个 Cluster 的距离。这种方法容易受到极端的影响，在两个**相似**的 Cluster 里，可能由于其中一个极端的 Data Point 距离比较近导致两个 Cluster 组合在一起。

- Complete Linkage – Complete Linkage 是和 Single Linkage 相反的，将两个 Clusters 里距离最远的两个 Data Points 作为 Clusters 之间的距离。在两个**不相似**的 Data Points 可能因为一个极端的 Data Points 而组合在一起。

- Average Linkage (UPGMA Unweighted Pair Group Mean Averaging) – Average Linkage 是将 2 个 Clusters 里每个 Data Point 的距离做平均，然后作为两个 Cluster 之间的距离。这个方法的计算量比较大，当时结果比 Single Linkage 和 Complete Linkage 来的合理。

Example:

	X	Y
P1	0.40	0.53
P2	0.22	0.38
P3	0.35	0.32
P4	0.26	0.19
P5	0.08	0.41
P6	0.45	0.30



在这里使用了 Euclidean Distance 的方法来计算 Cluster 之间的距离，初始化每个 Data Points 自己都是一个 Cluster。

- Calculate Euclidean distance, create the distance matrix.

$$\text{Distance } [(x,y), (a,b)] = \sqrt{(x-a)^2+(x-b)^2}$$

$$\text{Distance (P1,P2)} = \sqrt{(0.40-0.22)^2+(0.53-0.38)^2}$$

$$\text{(0.40,0.53), (0.22,0.38)} = \sqrt{(0.18)^2+(0.15)^2}$$

$$= \sqrt{0.0324+0.0225}$$

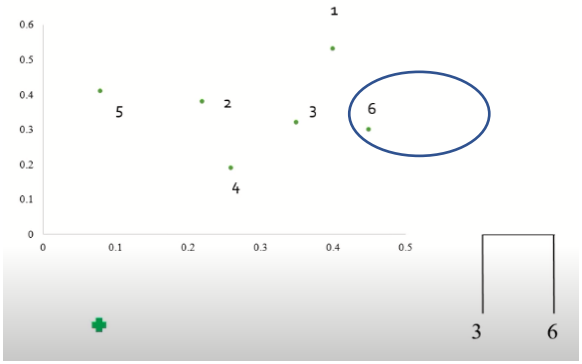
$$= \sqrt{0.0549}$$

$$= 0.23$$

计算了每个 Cluster 之间的距离就会得到以下的这个 Matrix Table。

	P1	P2	P3	P4	P5	P6
P1	0					
P2	0.23	0				
P3	0.22	0.15	0			
P4	0.37	0.20	0.15	0		
P5	0.34	0.14	0.28	0.29	0	
P6	0.23	0.25	0.11	0.22	0.39	0

之后找出最小的值，把这 2 个 Clusters 组成一个 Cluster。最小值是 0.11，也就是说 P3 和 P6 要组成同一个 Cluster



之后再继续计算 Cluster 之间的距离，在这里使用的是 Average Linkage。

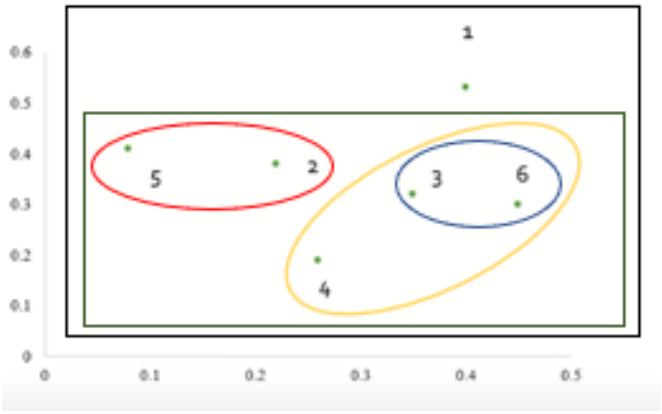
- The distance matrix is, AVG[dist(P3,P6),P1]

$$\text{dist}((P3,P6),P1) = \tfrac{1}{2}\,(\text{dist}(P3,P1)+\text{dist}(P6,P1)$$
$$= \tfrac{1}{2}\,(0.22+0.23)$$
$$= \tfrac{1}{2}\,(0.45)$$
$$= 0.23$$

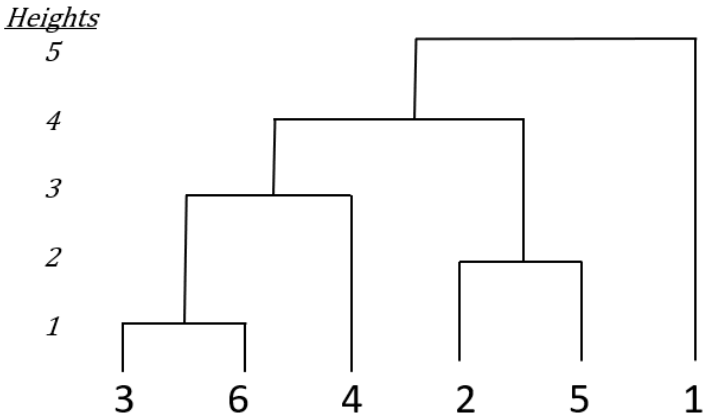
Update 新的 Cluster 与其他 Cluster 之间的距离，需要使用 P3 和 P4 同时对其他的 Cluster 去算出之间的距离然后取平均值

	P1	P2	P3,P6	P4	P5
P1	0				
P2	0.23	0			
P3,P6	0.23	0.2	0		
P4	0.37	0.20	0.19	0	
P5	0.34	0.14	0.34	0.29	0

计算之后，选出最小值，然后再继续求 Clusters 之间的距离然后再组成同一个 Cluster 然后再继续直到每个 Data Points 都是同一个 Cluster。



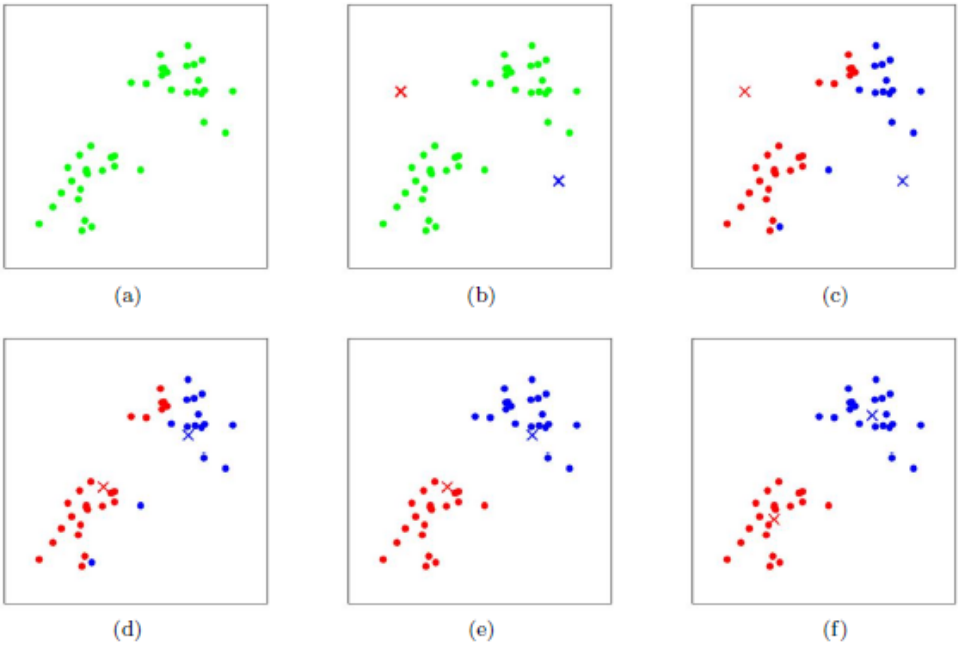
当所有 Data Point 已经组成同一个 Cluster 的时候，就能依据自己的需求对 Dendrogram 进行切割。



如果设置$height\ h$ 在 3.5 的话就会有 3 个 Clusters 就是 $(\mathbf{3},\mathbf{6},\mathbf{4})$，$(\mathbf{2},\mathbf{5})$ 和 $\mathbf{1}$。

K-Mean Clustering

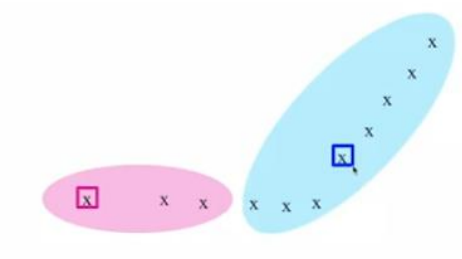
K-Mean Clustering 最大的特点是好理解，运算速度快，收敛速度快，但是只能应用于连续型数据，并且要在聚类前指定要分类成几类。



要使用 K-Mean Clustering 首先设定要分类成几类，也就是设定 k 的值，然后选取 k 个初始点。

$$z_i = \arg\min_c \|x_i - \mu_c\|^2$$

然后开始计算每个点，靠近哪一个初始点，就与那个点形成为同一个 Cluster。

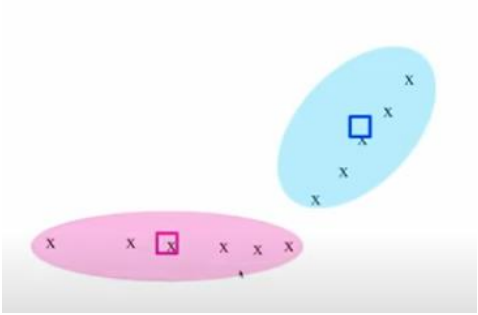


$$\mu_c = \frac{1}{m_c}\sum_{i \in S_c} x_i$$

$$S_c = \{i : z_i = c\} \rightarrow All\ data\ points\ that\ have\ z_i\ equal\ to\ C$$

$$m_c = |S_c| \rightarrow number\ of\ data\ points\ in\ the\ cluster$$

当看完和 Cluster 完所有的点之后，取每一个 Cluster 里面全部 Data Points 的 Average 作为新的 Centroid μ_c。之后再以新的 Centroid μ_c 来从新 Cluster 所有的点。

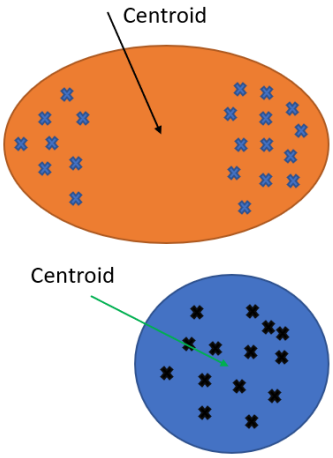


$$Cost \rightarrow C(z,\mu) = \sum_{i} \|x_i - \mu_{zi}\|^2$$

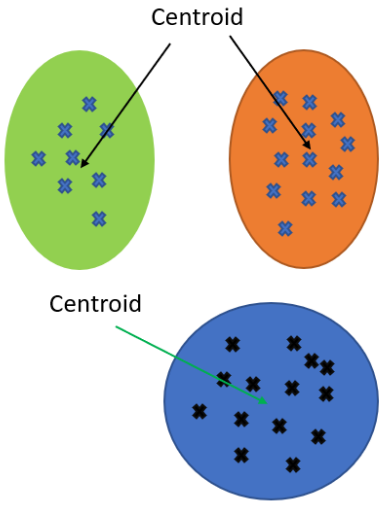
当计算的 $C(z,\mu)$ 开始没有太大的变化就代表已经 Cluster 好这笔 Data，也称之为 Convergence。

如何选择 k 的值

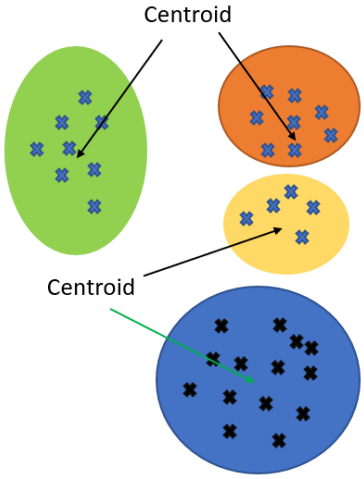
在做 K-Mean Clustering 之前需要设定 k 的值，表示需要聚类成几个类别。找到最适合的 k 值需要通过设定不同的 k 值，然后观察聚类结果。当 k 的值太小时，一个 Cluster 里 Data Points 与 Centroid 的距离会很大。



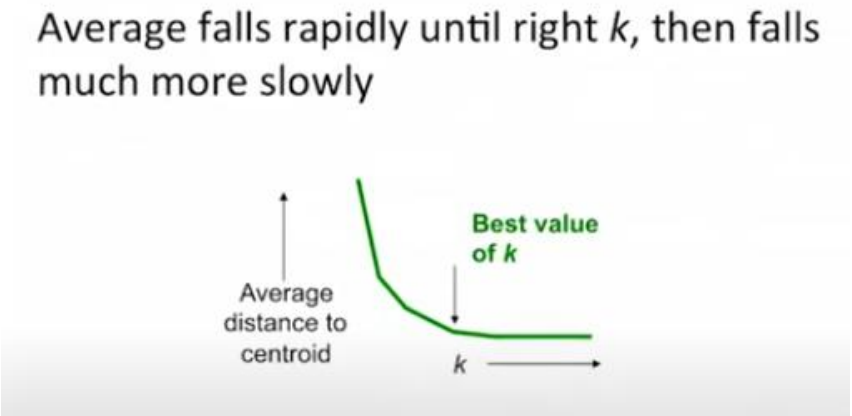
上图是一个例子当 k 设置太小，回导致的问题，因为 k 值设置不足，导致本该不是同一个 Cluster 的数据被归类成同一个 Cluster。



当 k 值设定的刚刚好，所有数据都能被 Cluster 正确，在 Cluster 里面的 Data Points 与 Centroid 的距离会比较靠近。



当 k 值设置过大，会让 Cluster 里的 Data Points 与 Centroid 的距离更近，但是不会有太大的帮助。

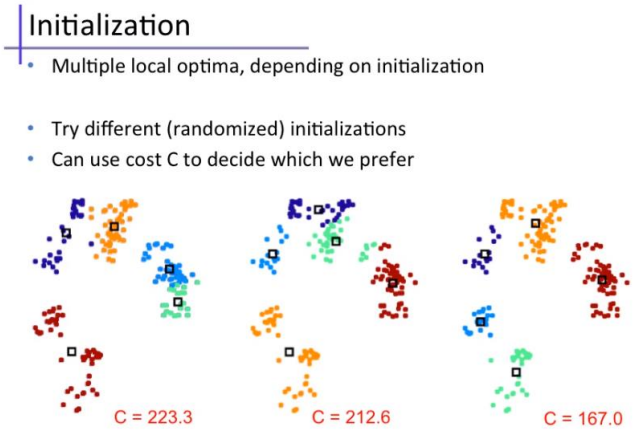


在测试不同 k 的值的时候，可以 Plot 一个 Graph 来看，当 k 的值到一定数目的时候，可以观察出 Average Distance to Centroid 没有明显的下降，能够从观察 Plot 的 Graph 选出最适合的 k 值。

如何选择 k 个初始点

Approach 1

测试不同的 Randomized 的初始点，然后计算 Cost 了，选出最适合的初始点。



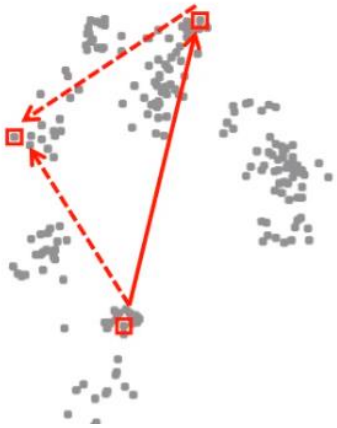
但是 Randomized 的初始点，可能会造成的问题是 2 个初始点太靠近，这会导致 Cluster 的结果不是最好。

Approach 2

可以先 Sample 一部 Data，然后使用 Hierarchical Clustering 来 Obtain k clusters，然后从么个 Cluster 里选出最靠近 Centroid 的 Data Point 当作初始点，在使用 k means 来做 Cluster。

Approach 3

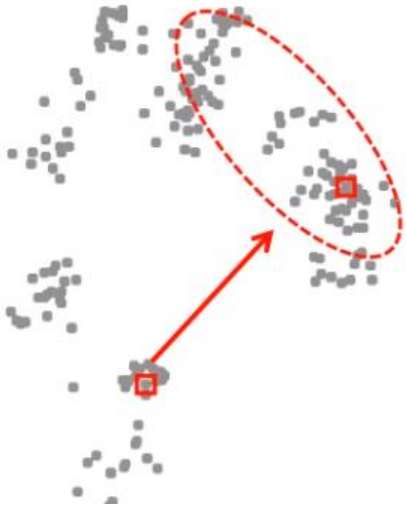
首先先随机在所有的 Data Points 里面选出一个随机点。然后 之后的点都求离与选好的初始点最远的距离，直到选完 k 个初始点。



当初始值离彼此都比较远的时候，更能够把数据 Cluster 的更平均，更正确。但是可能造成的问题是会选到 Outlier (离群值)。还有就是因为初始点不是随机的，导致不管从跑几次初始点都差不多，结果也都会差不多，会导致没有太大的机会 Improve on the quality of overall clustering by starting with a better initialization。

Approach 4 (K-means++)

比较好的方法来选出初始点是使用 K-means++ 的方法。这个方法是先随机选出第一个初始点。



Instead of 选出最远的初始点，这个方法则是计算每一个 Data Point 与已选出的初始点的距离，然后计算出一个几率 (Probability to Choose Next Centroid is Proportional to Distance)，来随机选择下一个初始点。

DBSCAN (Density-Based Spatial Clustering Applications with Noise)

DBSCAN 是一个 Unsupervised Learning Technic，也是争对高密度数据集做出的应对方法。DBSCAN **能够将高密度的数据集 Cluster 出来**之外，还**能无视检测到 Noise (Outliers)**。但是 DBSCAN **不擅长解决不同密度的数据集**，和 DBSCAN 也**不擅长解决高纬度的数据集**。

$$\boldsymbol{\varepsilon} \rightarrow \textit{Epsilon 最为Radius使用}$$

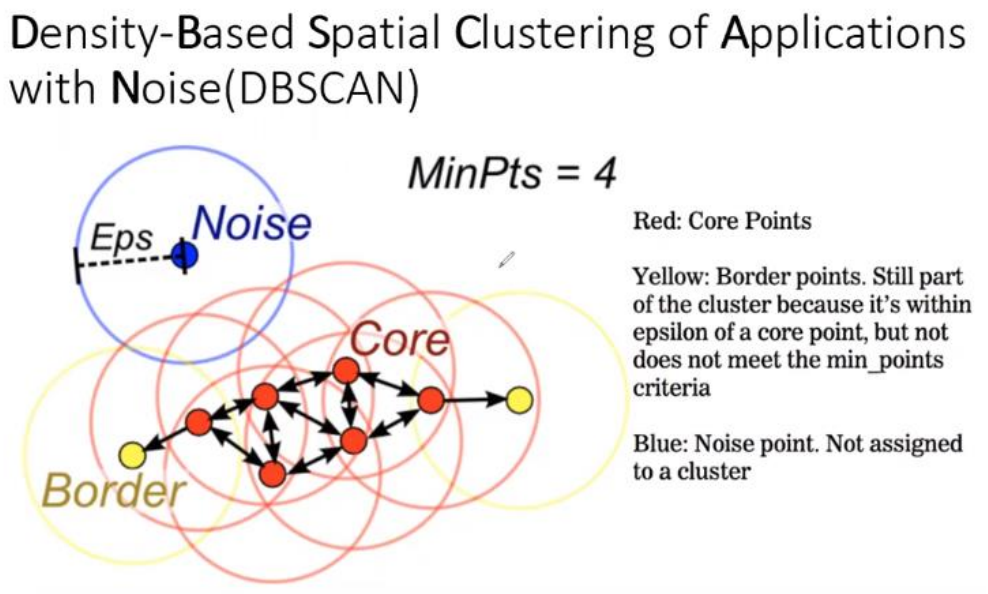
$$\boldsymbol{Minimum\ Points} \rightarrow \textit{作为 Core Point 的Threshold } (\geq \textit{ MinPts 就是Core Point})$$

在 DBSCAN 里分类了 3 种 Points

$$\boldsymbol{Core\ Point} \rightarrow \textit{一个圈里的 Point如果} \geq \textit{设定好的 MinPts就是 Core Point}$$

$$\boldsymbol{Border\ Point} \rightarrow \textit{一个圈里没有} \geq \textit{MinPts不过有最少一个 Core Point}$$

$$\boldsymbol{Noise\ Point} \rightarrow \textit{一个圈里没有 Core Point 也 } < \textit{MinPts}$$

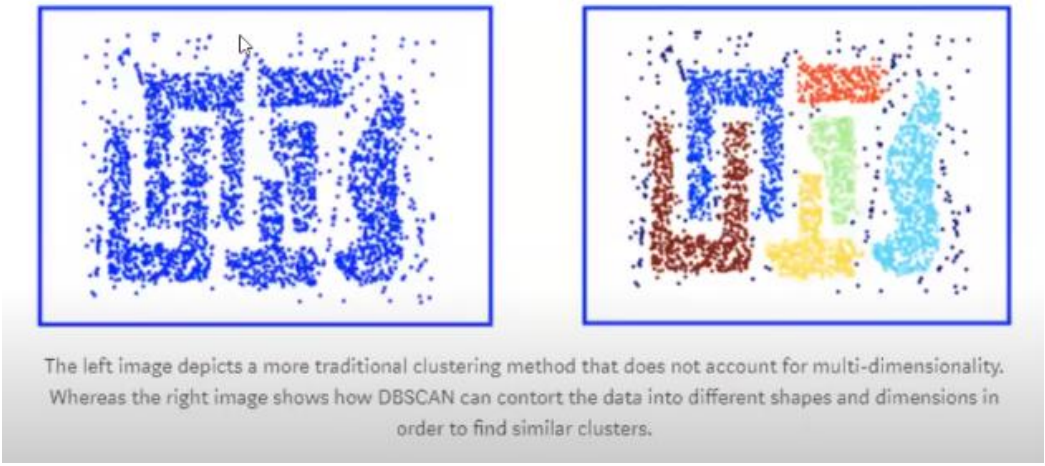


随机从一个 Data Point 开始，以初始点为中心和设定好的 ε 值作为 Radius 画出一个圆圈。计算有多少个 Data Points 在这个圆圈里，如果在这个圆圈里的 Data Points ≥ 先前设定好的 Minimum Points，就将这个点设为 Core Points，然后换下一个 Data Points 进行计算。

如果计算的圈里，Data Point < Minimum Points 但是有至少一个 Core Point 在里面，就将这个点设置为 Border Points。

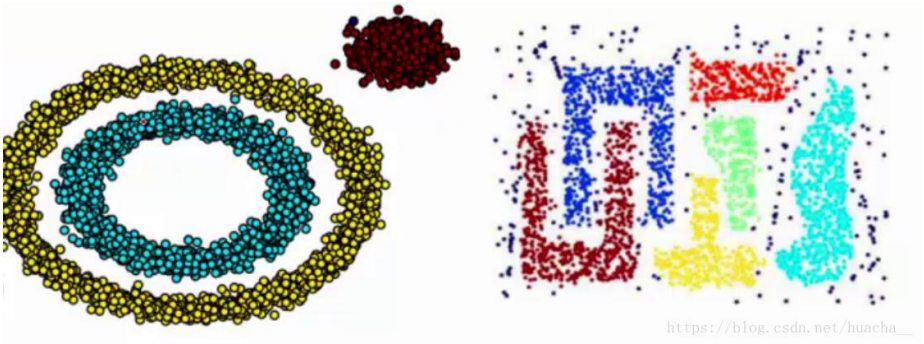
如果计算的圈里，Data Point < Minimum Points 而圈里没有任何的 Core Point 那么就定义这个点为 Noise Point 后，忽略这个点。

这个步骤一直持续到检测完所有的 Data Points。



The left image depicts a more traditional clustering method that does not account for multi-dimensionality. Whereas the right image shows how DBSCAN can contort the data into different shapes and dimensions in order to find similar clusters.

从上图的表现可以看出，在高密度的数据里，DBSCAN 要比 K-Means 或者 Hierarchical Clustering 好用。K-Means 的算法只能处理球形的 Cluster，这是因为 K-Means 算法本身计算平均距离的局限。

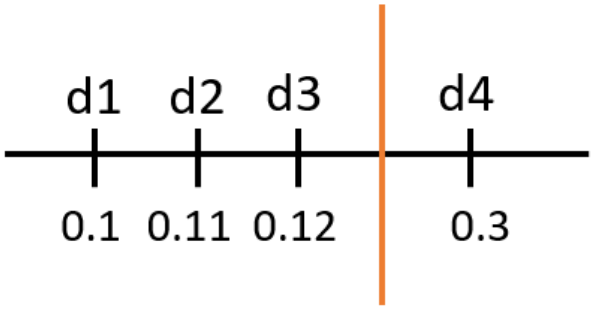


当数据是环形或者不规则的时候，DBSCAN 的计算能够更好的完成 Cluster 的任务。

选择参数

在 DBSCAN 里，需要定义的参数有 ε 和 Minimum Points。定义 ε 是比较困难，如果 ε 的值太大，就会导致 Cluster 变少，如果 ε 的值太小，Cluster 就会过多。这会影响 Clustering 的 Performance。

选择 ε 值的方法可以随机选出一个 Data Point 然后计算这个点与其他的点的距离，然后将算出的距离排成顺序。之后观察点跟点之间的距离差别。



这时候可以发现，d3 与 d1 和 d2 的距离是比较靠近，而 d4 的距离是比较远。这时候可以将 ε 设为 0.12 的值。不过通常这种方法比较麻烦，一般的做法都是选择不同的 ε 值训练完进行观察。

设置 Minimum Points 的方法也是，一般先设定成比较小的值，然后逐渐增加，来观察训练好的结果，再选择出适合的值。

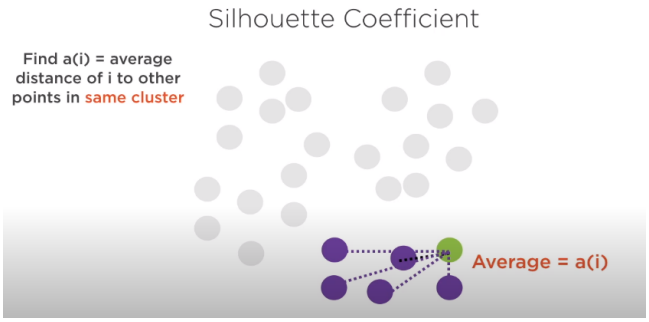
常用评估方法

这里使用的评估方法是 Silhouette Coefficient。

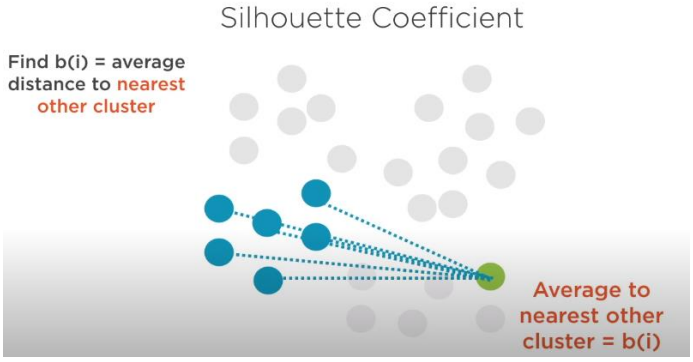
$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

$$s(i) = \begin{cases} 1 - \dfrac{a(i)}{b(i)}, & a(i) < b(i) \\ 0, & a(i) = b(i) \\ \dfrac{b(i)}{a(i)} - 1, & a(i) > b(i) \end{cases}$$

最好的 Silhouette Coefficient 的值是 1，而最差的值是 -1。



首先先在一个 Cluster 里面选出一个$Point\ i$，然后将同一个 Cluster 里面的所有 Data Points 与 $Point\ i$ 之间取距离后做平均值 $a(i)$。



然后选择一个离当前 Cluster 最近的一个 Cluster，计算 Cluster 里面所有的 Data Points 与 $Point\ i$ 之间的距离后取平均值 $b(i)$。

当计算的 $a(i)$ 值大于 $b(i)$，就表示当前的 $Point\ i$ 跟倾向与另一个 Cluster。当计算的 $a(i)$ 值小于 $b(i)$，就代表非常适合在当前的 Cluster。

Mean Shift Clustering