CS131 Lecture10：语义分割和聚类

by：斯坦福大学计算机科学系

github: https://github.com/zhaoxiongjun/CS131_notes_zh-CN （包含中英文版课件及相关课程视频）

1 聚类和分割

图像分割是计算机视觉中的一项任务，它的目的是识别像素组和图像区域的相似性和归属性。可以使用不同的相似性度量对像素进行分组；这包括纹理和颜色特征。图像分割的一个实例如下所示。在图 1 中，目标是对构成老虎、草地、天空和沙子的所有像素进行分组。结果分割可以在图 2 中观察到。

图 1：输入图片

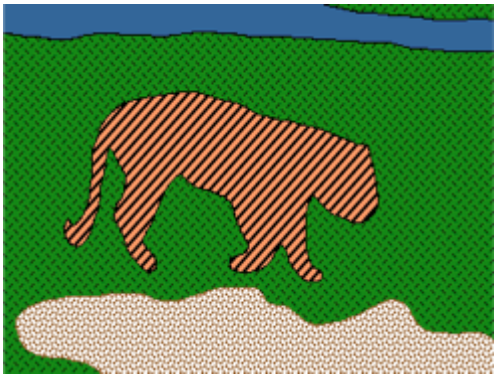

图 2：输出分割

图像分割的其他示例包括将视频帧分组为镜头，并分离图像的前景和背景。通过识别属于一起的像素组，可以将图像分解为不同的对象。除了直接用于对象识别之外，这还可以提高进一步处理的效率。

2 格式塔学派（Gestalt School）与因素

许多计算机视觉算法都是从计算机科学以外的领域提取的，图像分割也没有什么不同。计算机视觉研究者从心理学领域，特别是视觉知觉的格式塔理论中获得灵感。在很高的层次上，这一理论指出“整体大于各部分之和”，各部分之间的关系可以产生新的性质和特征。这个理论定义了格式塔因子，可以定义图像中的组

的属性。下面一些影响因素的例子。

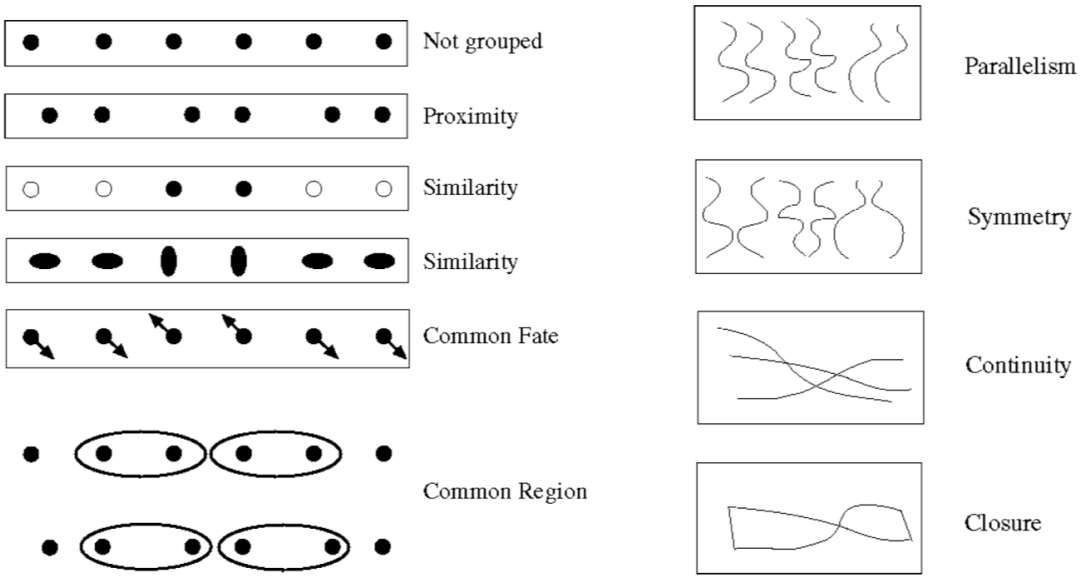

Figure 3:例子

这里是另一个因素的例子；虽然图 4 似乎没有有意义的视觉内容，但是在同一图像上添加重叠的灰色线（图 5）提供了与像素分组和图像内容相关的视觉提示。

图 4

图 5

现在我们可以更清楚地看到图像被一些 9 的灰色线条所遮挡。这是通过遮挡提示实现连续性的示例。灰色线条给我们一个暗示，黑色像素不是分开的，事实上应该组合在一起。通过将黑色像素组合在一起而不将它们视为单独的对象，我们的大脑能够识别出这张图片包含几个数字。

下面是另外一个例子：

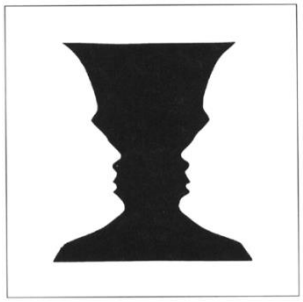

图 6

你看到什么了？你看到一个杯子还是两个头的侧面面对面？这两种选择都是正确的，这取决于你的观点。这种感知上的变化是由于我们所识别的前景和背景。如果我们把黑色像素作为前景，我们就会看到一个杯子。但是，如果我们把白色像素作为前景，我们会看到两张脸。
这是对影响人类感知视觉数据中像素/图像区域分组的一些因素的概述。然而，需要注意的一点是，尽管这些因素可能是直观的，但很难将这些因素转换为有效的算法。
我们如何进行图像分割？我们使用什么算法？考虑分割的一种方法是聚类。我们有几个像素，我们想把每个像素分配给一个簇。在下面的部分中，将详细介绍不同的集群方法。

3 凝聚聚类

集群是一种无监督的学习技术，其中多个数据点 x1,…,xn（每个数据点位于R^D中）被分组到集群中，而不知道正确的标签/集群。凝聚聚类是一种常用的聚类技术。
凝聚聚类背后的一般思想是观察点之间的相似性，以确定这些点应如何以合理的方式进行分组。在讨论算法的细节之前，我们必须首先决定如何确定相似性。

3.1 距离测量法

我们通过确定物体之间的距离来测量物体之间的相似性：距离越小，相似程度越高。有几个可能的距离函数，但很难确定什么是一个好的距离度量，所以通常重点放在标准的，研究的很好的距离度量上，如下面详细介绍的两个。

3.1.1欧几里得距离

一个常见的相似性度量是欧几里得距离；它通过考虑角度和大小来度量两个数据点（x 和 x’）之间的距离。我们可以将欧几里得距离写为：

$$sim(x, x') = x^T x'$$

这种距离测量并没有将向量归一化，因此将向量的大小考虑到相似度计算中。

3.1.2 弦相似度

另一个常见的解决方案是余弦相似性度量；它只考虑两个数据点之间的角度，即 x 和 x’。注意，与欧几里得距离不同，余弦度量只表示相似性，而不是距离。这意味着数据点 x 与其自身的相似性等于 1。顾名思义，这个度量依赖于两点之间的余弦，如：

$$sim(x, x') = cos(\theta)$$

$$= \frac{x^T x'}{||x|| \cdot ||x'||}$$

$$= \frac{x^T x'}{\sqrt{x^T x}\sqrt{x'^T x'}}$$

通过矢量大小的划分，可以使距离度量标准化，从而确保度量仅依赖于对象之间的角度。

3.2 理想的聚类特性

既然已经定义了潜在的距离度量，下一步就是选择一种聚类技术。在选择特定技术时，我们可能需要考虑聚类方法的各种属性：

- 可扩展性——从计算能力和内存方面来说
- 不同的数据类型——算法应支持在R^D内的所有数据的任意数据。
- 输入参数——算法的参数调整不应该很困难。如果算法在很大程度上不依赖于我们对数据的准确理解，那么它会更有用。
- 可解释——我们应该能够解释结果。
- 约束——算法应该有效地使用预先定义的约束（例如，我们知道两个点应该在同一个簇中，或者它们不应该属于一起）。

以下部分介绍了集群的实现及其优缺点。

3.3 集群实现

凝聚聚类通过将更近的点分组来计算数据点之间的相似性。新创建的组可以进一步合并到靠近它们的其他组；这种迭代过程会生成更大的组，直到只剩下一个组为止。这创建了一个层次结构，最好将其视为树形图。
我们可以在下图中看到这一点，它显示了数据点和聚集聚类算法的结果。

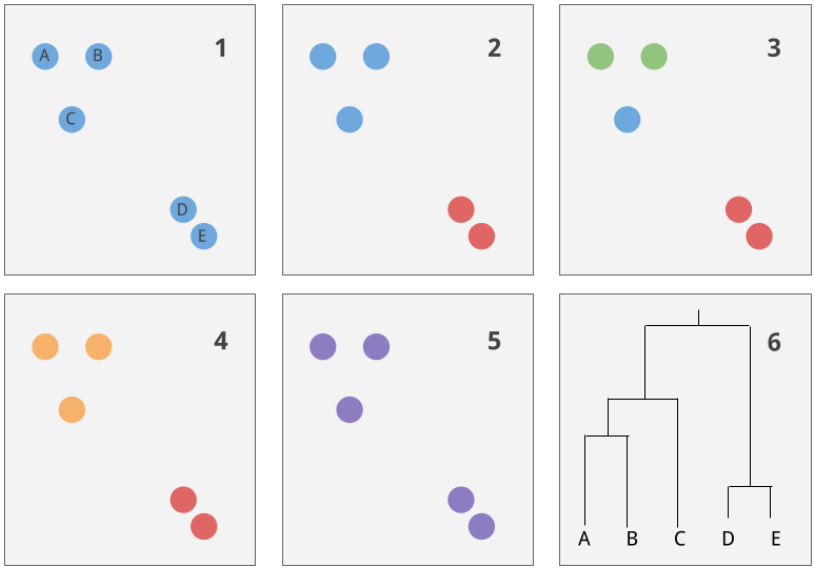

图 7：样本输入上的凝聚聚类和结果树形图

第一张图显示了所有的数据点，图 2 到图 5 显示了聚类算法的各个步骤。步骤 2 将两个红点组合在一起；步骤 3 将两个绿点组合在一起；步骤 4 将上一个绿点组和附近的蓝点组合成一个新的橙色组；步骤 5 将所有点组合成一个大组。这将创建图 6 中的树形图。

3.3.1 算法

我们的通用算法基于我们的视觉，有四个主要步骤：

1. 将每个点初始化为其自己的群集
2. 找到最相似的一对簇
3. 将相似的集群对合并到父集群中
4. 重复步骤 2 和 3，直到我们只有一个集群。

3.3.2问题

虽然聚集聚类是一种强有力的技术，但在实现时应考虑各种因素。例如：

1. 我们如何定义集群之间的相似性？我们如何测量两个群之间的距离？
   我们可以用各种不同的方法测量两个集群之间的距离，包括点之间的平均距离、集群中点之间的最小距离、每个集群的平均值之间的距离以及集群中点之间的最大距离。用于测量集群间距离的方法会对结果产生很大影响。

2. 我们选择了多少个集群？
   当我们创建树形图时，我们可以根据距离阈值决定需要多少簇。或者，我们可以在不同的层次水平切割树形图，以创建尽可能多的簇。

3.4 最近集群的不同度量

在分割数据集时，我们可以使用三个主要模型来确定簇点之间的距离：单个、完整和平均。

1. 单链接：

距离的计算公式为：

$$d(C_i, C_j) = min_{x \in C_i, x' \in C_j} d(x, x')$$

对于单链接结构，我们利用两个集群中点之间的最小距离进行集群。

这也被称为最小生成树。一旦我们通过了集群之间可接受距离的阈值，就可以停止集群。该算法倾向于生成长而细的簇（因为很容易将远点链接到同一簇中，因为我们只关心该簇中距离最小的点）。

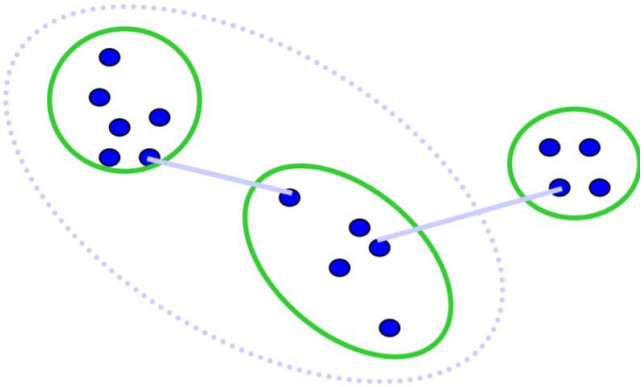

图 9：使用最近集群的单个链路测量的图像分割示例

2. 完整链接：

距离的计算公式为：

$$d(C_i, C_j) = max_{x \in C_i, x' \in C_j} d(x, x')$$

在完整链接的情况下，我们利用两个群中点之间的最大距离进行聚类。

该算法倾向于生成直径大致相等的紧凑和紧密的簇（因为它倾向于将所有点紧密地连接在一起）。

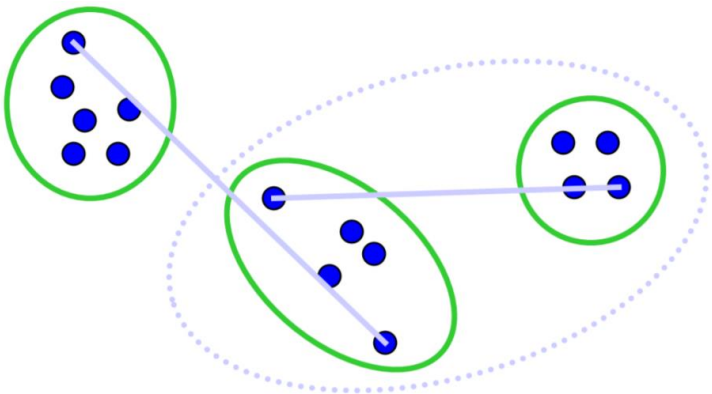

图 9：使用最近集群的完整链路测量的图像分割示例

3. 平均链接：

距离的计算公式为：

$$d(C_i, C_j) = \frac{\sum x \in C_i, x' \in C_j d(x, x')}{|C_i| \cdot |C_j|}$$

对于平均连接，我们利用两个簇中点之间的平均距离进行聚类。

该模型对噪声具有鲁棒性，因为与单链接和完整链接不同，距离不依赖于单对点，而单对点可能受数据中的伪影影响。

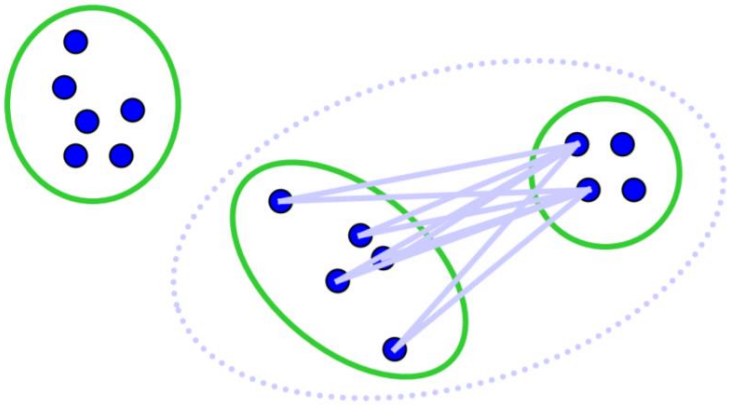

图 10：使用最近集群的平均链路测量的图像分割示例

3.5 聚类结论

聚集聚类的正特征：

1. 易于实现和应用
2. 群集形状适应数据集
3. 产生一个集群层次结构
4. 不需要指定初始化时的集群数

负特征：

1. 可能返回不平衡的集群
2. 必须指定群集数的阈值
3. 运行时为 0（n3）时缩放不好
4. 贪心的合并会陷入局部极小

4 群集分析（K-Means Clustering）

另一种算法是 k 均值聚类。它将固定数量的集群“中心”标识为其集群的代表，并根据其最接近的中心标记每个点。K 均值和聚集聚类的一个主要区别是，K 均值需要输入目标数的聚类来运行算法。

4.1 图像分割例子

在图 11 的左上角，我们有一个具有三个不同颜色区域的图像，因此可以通过将右上角显示的每个颜色强度分配给不同的簇来使用颜色强度分割图像。然而，在左下角的图像中，图像中却杂乱无章。为了分割图像，我们可以使用 k-means。

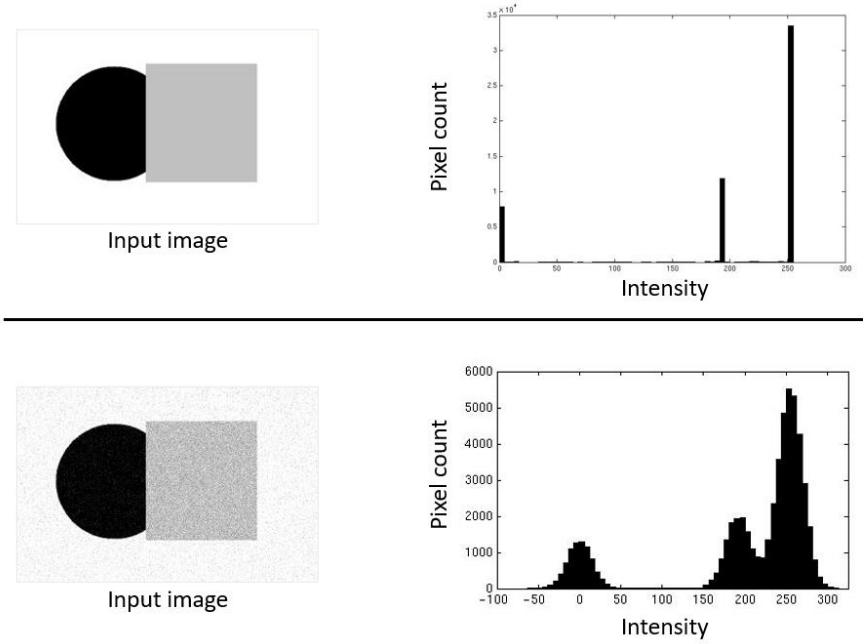

图 11:使用 k - means 图像分割的例子。左上角的图片有三个不同的颜色,但左下角图片有高斯噪声

使用 k-均值，这里的目标是确定三个簇中心作为代表强度，并根据其最近的中心标记每个像素。最好的聚类中心是那些将所有点与其最近的聚类中心之间的平方距离之和最小化的聚类中心 ci。

$$SSD = \sum_{i \in \text{clusters}} \sum_{x \in \text{cluster}_i} (x - c_i)^2$$

当我们使用 k-means 来汇总数据集时，目标是最小化分配给每个集群的数据点的差异。考虑到一定数量的集群，我们希望尽可能多地保存信息。这可以通过下面的公式来证明。

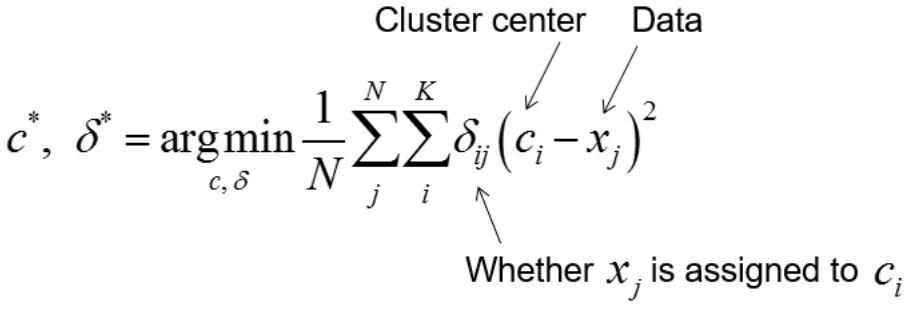

图 12:集群的总结

4.2 算法

找到聚类中心和点的群成员可以被认为是一个“鸡和蛋”的问题。如果我们知道聚类中心，我们可以通过将每个点分配给最近的中心来将点分配给组。另一方面，如果我们知道群成员的属性，我们可以通过计算每个小组的平均数找到中心。因此，我们在任务之间交替进行。

为了找到中心和群成员身份，我们首先初始化 k 个集群中心，通常是通过随机分配它们。然后，我们运行一个迭代过程，计算特定迭代次数的群成员身份和聚类中心，或者直到聚类中心的值收敛。流程概述如下：

1. 初始化聚类中心 c1,…,ck。通常，这些中心是随机选择的数据点。
2. 将数据集中的每个点指定给最近的中心。在凝聚聚类中，我们可以使用欧氏距离或余弦距离度量来计算到每个中心的距离。
3. 将群集中心更新为群集组中点的平均值。
4. 重复步骤 2-3，直到簇中心的值停止更改或算法已达到最大迭代次数。

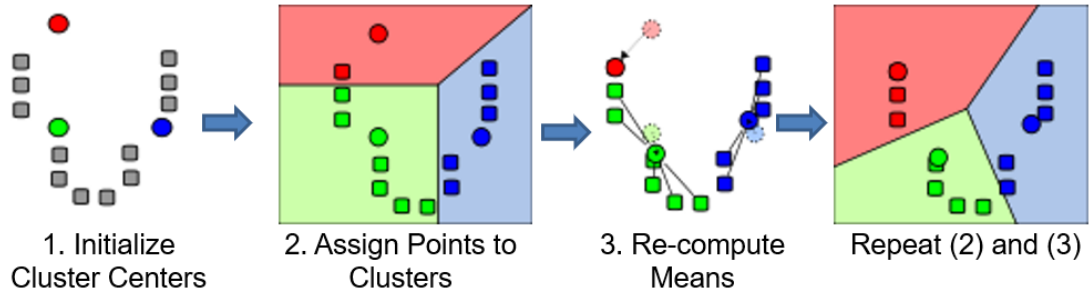

图 13:可视化的 k - means 聚类

4.3 输出

每次运行时，k-均值都收敛到局部最小解。此外，由于中心是随机初始化的，因此每次运行算法都可能返回不同的结果。因此，初始化多个 k-means 运行，然后选择最具代表性的集群将得到最佳结果。最佳聚类可以通过最小化到中心的平方距离之和或每个聚类的方差来衡量。k-means 最适用于球面数据。

4.4 用聚类做分割

与第 4.1 节中的示例一样，可以使用聚类来分割图像。虽然颜色强度单独在图 11 这样的情况下是有效的，但是图 14 这样的其他情况可能需要我们定义一个特征空间，在这个空间中我们选择像素的哪些特征作为集群的输入。换句话说，特征空间的选择直接影响到数据点之间相似性度量的计算；特征空间的创造性选择使我们能够以更容易区分聚类的方式“表示”数据点。

图 14：熊猫图

除了像素强度，使用图像特征空间的像素分组示例还包括 RGB 颜色相似性、纹理相似性和像素位置。基于颜色相似性的聚类可以使用红色、绿色和蓝色的单独特征进行建模。纹理可以通过应用特定过滤器后像素的相似性来测量。位置特征包括图像中像素的坐标。强度和位置可以一起用于根据相似性和接近性对像素进行分组。

4.5 K-Means++

k - means 方法吸引人是因为它的速度和简单但不是其准确性。通过对 k 均值聚类问题的初始种子选择进行变量扩充，可以避免有时是 k 均值聚类结果的任意不良聚类。为 k-means++选择初始种子的算法概述如下：

1. 从数据点随机选择起点

2. 计算距离 d（x），即每个数据点 x 到所选中心的距离。通过使用加权概率分布，与 d（x）平方成比例的概率选择新的数据点作为新的中心。

3. 重复前面的步骤，直到选择了 K 中心，然后在选择初始种子时继续进行通常的 K 均值聚类过程。

K-means++的复杂度为 Olong（K）

4.6 集群评估

聚类结果可以用各种方法进行评估。例如，有一个内部评估措施，它包括给一个单一的质量分数的结果。另一方面，外部评估将聚类结果与现有的真实分类进行比较。更定性地说，我们可以根据聚类的生成度量来评估聚类的结果：从聚类中重建点的效果如何，或者聚类的中心是数据的良好表示。另一种评估方法是一种识别方法，在这种方法中，我们评估集群与标签的对应程度。我们检查集群是否能够分离出应该分离的东西。由于没有与无监督学习相关的标签，因此此度量只能用于有监督学习。

4.7 优缺点

优点：

1. 简单,容易实现
2. 低维数据的快速
3. 好表示的数据(集群中心条件方差最小化)

缺点:

1. 不识别异常值
2. 需要指定 k 值,它是未知的
3. 不能处理 non-globular 数据不同的大小和密度
4. 局限于数据中心的概念
5. 收敛到局部最小的目标函数,而不是不能保证收敛到全局最小的目标函数

为了选择簇数或 k 值，可以根据不同的 k 值绘制目标函数。目标函数在某个 k 值上的突然变化暗示了数据中特定数量的簇。这项技术被称为“knee-finding″ or ″elbow-finding″

5 Mean-shift Clustering

Mean-shift 聚类是另一种聚类算法。从本质上讲，Mean-shift 聚类是指在我们的特征空间中找到最密集的区域。该算法有四个主要步骤：

1. 初始化随机种子和窗口 W
2. 计算窗口的重心 $\mathrm{W}: \sum_{x \in W} xH(x)$
3. 改变搜索窗口 W 到平均
4. 重复步骤 2,直到收敛

在思想上可视化此算法的一种方法是将每个数据点描绘为大理石。如果每一个大理石都被高密度的区域吸引,那么所有的大理石最终都会聚集到一个或多个中心。我们还可以尝试通过这张图片来可视化算法：

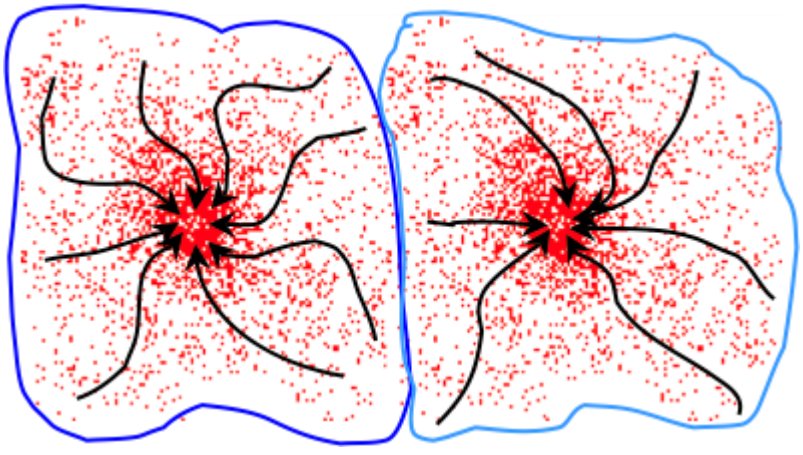

图 15:Mean-shift 聚类的结果

在这张图中，我们看到算法将生成 2 个集群。左边的所有数据点集中在一个中心，右边的所有数据点集中在另一个中心。
要了解更多关于 Mean-shift 聚类的信息，请参阅下一章。