

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

25 Clustering

25.1 Introduction

Clustering is the process of grouping similar objects together. There are two kinds of inputs we might use. In similarity-based clustering, the input to the algorithm is an $N \times N$ dissimilarity matrix or distance matrix D . In feature-based clustering, the input to the algorithm is an $N \times D$ feature matrix or design matrix X . Similarity-based clustering has the advantage that it allows for easy inclusion of domain-specific similarity or kernel functions (Section 14.2). Featurebased clustering has the advantage that it is applicable to “raw”, potentially noisy data. We will see examples of both below.

In addition to the two types of input, there are two possible types of output: flat clustering, also called partitional clustering, where we partition the objects into disjoint sets; and hierarchical clustering, where we create a nested tree of partitions. We will discuss both of these below. Not surprisingly, flat clusterings are usually faster to create ($O(ND)$ for flat vs $O(N^2 \log N)$ for hierarchical), but hierarchical clusterings are often more useful.

Furthermore, most hierarchical clustering algorithms are deterministic and do not require the specification of K , the number of clusters, whereas most flat clustering algorithms are sensitive to the initial conditions and require some model selection method for K . (We will discuss how to choose K in more detail below.)

The final distinction we will make in this chapter is whether the method is based on a probabilistic model or not. One might wonder why we even bother discussing non-probabilistic methods for clustering. The reason is two-fold: first, they are widely used, so readers should know about them; second, they often contain good ideas, which can be used to speed up inference in a probabilistic models.

25.1.1 Measuring (dis)similarity

A dissimilarity matrix D is a matrix where $d_{i,i} = 0$ and $d_{i,j} \geq 0$ is a measure of “distance” between objects i and j .

Subjectively judged dissimilarities are seldom distances in

the strict sense, since the triangle inequality, $d_{i,j} \leq d_{i,k} + d_{j,k}$, often does not hold. Some algorithms require D to be a true distance matrix, but many do not. If we have a similarity matrix S, we can convert it to a dissimilarity matrix by applying any monotonically decreasing function, e.g., $D = \max(S) - S$.

The most common way to define dissimilarity between objects is in terms of the dissimilarity of their attributes:

$$\Delta(x_i, x_j) = D \sum_{j=1}^D \Delta_j(x_{ij}, x_{ij}) \quad (25.1)$$

Some common attribute dissimilarity functions are as follows:

- Squared (Euclidean) distance:

$$\Delta_j(x_{ij}, x_{ij}) = (x_{ij} - x_{ij})^2 \quad (25.2)$$

Of course, this only makes sense if attribute j is real-valued.

- Squared distance strongly emphasizes large differences (because differences are squared). A more robust alternative is to use an L1 distance:

$$\Delta_j(x_{ij}, x_{ij}) = |x_{ij} - x_{ij}| \quad (25.3)$$

This is also called city block distance, since, in 2D, the distance can be computed by counting how many rows and columns we have to move horizontally and vertically to get from x_i to x_j .

- If x_i is a vector (e.g., a time-series of real-valued data), it is common to use the correlation coefficient (see Section 2.5.1). If the data is standardized, then $\text{corr}[x_i, x_j] = \frac{\sum_{k=1}^K x_{ik} x_{jk}}{\sqrt{\sum_{k=1}^K x_{ik}^2} \sqrt{\sum_{k=1}^K x_{jk}^2}}$, and hence $\Delta_j(x_{ij}, x_{ij}) = 2(1 - \text{corr}[x_i, x_j])$. So clustering based on correlation (similarity) is equivalent to clustering based on squared distance (dissimilarity).

- For ordinal variables, such as {low, medium, high}, it is standard to encode the values as real-valued numbers, say 1/3, 2/3, 3/3 if there are 3 possible values. One can then apply any dissimilarity function for quantitative variables, such as squared distance.

- For categorical variables, such as {red, green, blue}, we usually assign a distance of 1 if the features are different, and a distance of 0 otherwise. Summing up over all the categorical features gives

$$\Delta(x_i, x_j) = D \sum_{j=1}^D I(x_{ij} \neq x_{ij}) \quad (25.4)$$

This is called the hamming distance.

25.1.2 Evaluating the output of clustering methods

The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong

effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage. — Jain and Dubes (Jain and Dubes 1988)

Clustering is an unupervised learning technique, so it is hard to evaluate the quality of the output of any given method. If we use probabilistic models, we can always evaluate the likelihood of a test set, but this has two drawbacks: first, it does not directly assess any clustering that is discovered by the model; and second, it does not apply to non-probabilistic methods. So now we discuss some performance measures not based on likelihood. Intuitively, the goal of clustering is to assign points that are similar to the same cluster, and to ensure that points that are dissimilar are in different clusters. There are several ways of measuring these quantities e.g., see (Jain and Dubes 1988; Kaufman and Rousseeuw 1990).

However, these internal criteria may be of limited use. An alternative is to rely on some external form of data with which to validate the method. For example, suppose we have labels for each object, as in Figure 25.1. (Equivalently, we can have a reference clustering; given a clustering, we can induce a set of labels and vice versa.) Then we can compare the clustering with the labels using various metrics which we describe below. We will use some of these metrics later, when we compare clustering methods.

25.1.2.1 Purity

Let N_{ij} be the number of objects in cluster i that belong to class j , and let $N_i = \sum_j N_{ij}$ be the total number of objects in cluster i . Define $p_{ij} = N_{ij}/N_i$; this is the empirical distribution over class labels for cluster i . We define the purity of a cluster as $\pi_i = \max_j p_{ij}$, and the overall purity of a clustering as

$$\text{purity} = \frac{1}{N} \sum_i N_i \pi_i \quad (25.5)$$

For example, in Figure 25.1, we have that the purity is
 $6 \cdot 17 / 56 + 6 \cdot 17 / 46 + 5 \cdot 17 / 35 = 5+4+3 / 17 = 0.71 \quad (25.6)$

The purity ranges between 0 (bad) and 1 (good). However, we can trivially achieve a purity of 1 by putting each object into its own cluster, so this measure does not penalize for the number of clusters.

25.1.2.2 Rand index

Let $U = \{u_1, \dots, u_R\}$ and $V = \{v_1, \dots, v_C\}$ be two different partitions of the N data points, i.e., two different (flat)

clusterings. For example, U might be the estimated clustering and V is reference clustering derived from the class labels. Now define a 2×2 contingency table, containing the following numbers: TP is the number of pairs that are in the same cluster in both U and V (true positives); TN is the number of pairs that are in the different clusters in both U and V (true negatives); FN is the number of pairs that are in the different clusters in U but the same cluster in V (false negatives); and FP is the number of pairs that are in the same cluster in U but different clusters in V (false positives). A common summary statistic is the Rand index:

$$R = \frac{TP + TN}{TP + TN + FP + FN} \quad (25.7)$$

This can be interpreted as the fraction of clustering decisions that are correct. Clearly $0 \leq R \leq 1$.

For example, consider Figure 25.1, The three clusters contain 6, 6 and 5 points, so the number of "positives" (i.e., pairs of objects put in the same cluster, regardless of label) is

$$TP + FP = 6 \cdot 2 + 6 \cdot 2 + 5 \cdot 2 = 40 \quad (25.8)$$

Of these, the number of true positives is given by

$$TP = 5 \cdot 2 + 5 \cdot 2 + 3 \cdot 2 + 2 \cdot 2 = 20 \quad (25.9)$$

where the last two terms come from cluster 3: there are 3 2 pairs labeled C and 2 2 pairs labeled A. So $FP = 40 - 20 = 20$. Similarly, one can show $FN = 24$ and $TN = 72$. So the Rand index is $(20 + 72)/(20 + 20 + 24 + 72) = 0.68$.

The Rand index only achieves its lower bound of 0 if $TP = TN = 0$, which is a rare event. One can define an adjusted Rand index (Hubert and Arabie 1985) as follows:

$$AR = \frac{index - expected\ index}{max\ index - expected\ index} \quad (25.10)$$

Here the model of randomness is based on using the generalized hyper-geometric distribution, i.e., the two partitions are picked at random subject to having the original number of classes and objects in each, and then the expected value of $TP + TN$ is computed. This model can be used to compute the statistical significance of the Rand index.

The Rand index weights false positives and false negatives equally. Various other summary statistics for binary decision problems, such as the F-score (Section 5.7.2.2), can also be used. One can compute their frequentist sampling distribution, and hence their statistical

significance, using methods such as bootstrap.

25.1.2.3 Mutual information

Another way to measure cluster quality is to compute the mutual information between U and V (Vaithyanathan and Dom 1999). To do this, let $p_{UV}(i, j) = |u_i \cap v_j| / N$ be the probability that a randomly chosen object belongs to cluster u_i in U and v_j in V. Also, let $p_U(i) = |u_i| / N$ be the probability that a randomly chosen object belongs to cluster u_i in U; define $p_V(j) = |v_j| / N$ similarly. Then we have $I(U, V) = \sum_{i=1}^R \sum_{j=1}^C p_{UV}(i, j) \log \frac{p_{UV}(i, j)}{p_U(i)p_V(j)}$ (25.11)

This lies between 0 and $\min\{H(U), H(V)\}$. Unfortunately, the maximum value can be achieved by using lots of small clusters, which have low entropy. To compensate for this, we can use the normalized mutual information,

$$NMI(U, V) = I(U, V) / (H(U) + H(V)) / 2 \quad (25.12)$$

This lies between 0 and 1. A version of this that is adjusted for chance (under a particular random data model) is described in (Vinh et al. 2009). Another variant, called variation of information, is described in (Meila 2005).

已思考 24s

下面先给出**完整中文翻译**（保留原有小节与公式编号，并在个别明显的排版/符号处做了不改意图的微调），然后是**通俗讲解**，最后是**要点总结**。

译文

25 聚类 (Clustering)

25.1 引言 (Introduction)

聚类就是把相似的对象分到一组里。我们可能会用两种不同形式的输入数据：

- 基于相似度的聚类**: 算法的输入是一个 $N \times N$ 的不相似度矩阵或距离矩阵 D 。
- 基于特征的聚类**: 算法的输入是一个 $N \times D$ 的特征矩阵或设计矩阵 X 。

基于相似度的聚类的优点是，它很容易把领域特定的相似度或核函数（见第 14.2 节）纳入进来。基于特征的聚类的优点是，它可以直接用于“原始”的、可能含噪声的数据。下面我们会看到两者的例子。

除了两种输入类型外，输出也有两种可能：**扁平聚类** (flat clustering，也叫**划分式聚类** partitional clustering)，即把对象划分到互不重叠的集合中；以及**层次聚类** (hierarchical clustering)，即创建一个嵌套的划分树。我们都会讨论。毫不意外地，扁平

聚类通常构建更快（扁平约为 $O(ND)$ ，层次约为 $O(N^2 \log N)$ ），但层次聚类往往更有用。此外，多数层次聚类算法是确定性的，并且不需要指定簇的个数 K ；而多数扁平聚类算法对初始条件很敏感，并且需要某种方式来选择 K （我们稍后会讨论如何选 K ）。

本章还要作的最后一个区分是：方法是否基于概率模型。有人可能会问：既然有概率模型，为什么还要讲非概率的聚类方法？理由有二：第一，这些方法被广泛使用，读者应该了解；第二，它们常常包含一些好点子，可以用来加速概率模型中的推断。

25.1.1 衡量（不）相似度 (Measuring (dis)similarity)

不相似度矩阵 D 指满足 $d_{i,i} = 0$ 且 $d_{i,j} \geq 0$ 的矩阵，其中 $d_{i,j}$ 是对象 i 与 j 之间“距离”的度量。从主观判断得到的不相似度通常并不严格满足距离的性质，因为三角不等式 $d_{i,j} \leq d_{i,k} + d_{k,j}$ 往往不成立。有些算法要求 D 必须是真正的距离矩阵，但很多算法并不要求。如果我们有一个**相似度矩阵 S** ，可以通过施加任何**单调递减函数**把它转成不相似度矩阵，例如 $D = \max(S) - S$ 。

定义对象间不相似度的最常见方式，是基于**它们各自属性的不相似度之和**：

$$\Delta(\mathbf{x}_i, \mathbf{x}_{i'}) = \sum_{j=1}^D \Delta_j(x_{ij}, x_{i'j}) \quad (25.1)$$

(注：原文处有少量排版痕迹，这里用 i' 表示“另一个对象”。)

一些常见的属性层面的不相似度函数如下：

- **平方（欧氏）距离**：

$$\Delta_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2 \quad (25.2)$$

当然，这在第 j 个属性是实数值时才有意义。

- 平方距离会**强烈放大大差异**（因为差异被平方）。一个更稳健的替代是使用 l_1 距离：

$$\Delta_j(x_{ij}, x_{i'j}) = |x_{ij} - x_{i'j}| \quad (25.3)$$

这也叫**城市街区距离** (city block distance, 曼哈顿距离)，因为在二维里，从 \mathbf{x}_i 到 $\mathbf{x}_{i'}$ 的距离可以通过“水平和垂直方向分别走多少行/列”的步数来计算。

- 如果 \mathbf{x}_i 是一个向量（例如一段实值时间序列），常用**相关系数**（见 2.5.1 节）。若数据已标准化，则 $\text{corr}[\mathbf{x}_i, \mathbf{x}_{i'}] = \sum_j x_{ij}x_{i'j}$ ，于是 $\sum_j (x_{ij} - x_{i'j})^2 = 2(1 - \text{corr}[\mathbf{x}_i, \mathbf{x}_{i'}])$ 。因此，基于**相关性（相似度）的聚类与基于平方距离（不相似度）**的聚类是等价的。

- 对于**有序变量** (ordinal) , 如 {低, 中, 高}, 通常把值编码成实数, 比如在 3 个可能值时编码为 1/3, 2/3, 3/3。随后就可以应用任何用于定量变量的不相似度 (如平方距离) 。
- 对于**分类变量** (categorical) , 如 {红, 绿, 蓝}, 通常若两个取值不同则赋距离 1, 相同则赋距离 0。把所有分类特征的差异加总得到

$$\Delta(\mathbf{x}_i, \mathbf{x}_{i'}) = \sum_{j=1}^D \mathbf{I}(x_{ij} \neq x_{i'j}) \quad (25.4)$$

这叫**汉明距离**。 (注: 原文处指标函数写成“=”应为“不等于”以符合“不同为 1、相同为 0”的定义。)

25.1.2 评估聚类输出 (Evaluating the output of clustering methods)

“对聚类结构的验证是聚类分析中最困难且最让人沮丧的部分。如果不在这方面下大功夫, 聚类分析将依然是一门只有那些真正的信徒——拥有经验且极具勇气的人——才能掌握的黑艺。”——Jain 与 Dubes (1988)

聚类是**无监督学习**技术, 因此很难评估某个方法输出的质量。若我们使用概率模型, 确实总能评估测试集的似然, 但这有两点缺点: 第一, 它并不直接评估模型“发现的聚类”; 第二, 它不适用于**非概率**的方法。因此, 我们现在讨论一些**不基于似然**的性能度量。

直观地说, 聚类的目标是: 把**相似**的点分到**同一簇**, 同时让**不相似**的点分到**不同簇**。衡量这些的方式有很多 (见 Jain & Dubes 1988; Kaufman & Rousseeuw 1990)。不过, 这些**内部指标**有时用处有限。另一种做法是依赖某种**外部数据**来验证方法。例如, 假设我们为每个对象都有一个**标签**, 如图 25.1 (等价地, 我们也可以有一个**参考聚类**; 已知聚类即可导出一组标签, 反之亦然)。那么, 我们可以用下面的各种度量把聚类与这些标签做比较。稍后在比较聚类方法时, 我们会使用其中的一些指标。

25.1.2.1 纯度 (Purity)

令 N_{ij} 为簇 i 中**属于类别** j 的对象数, 令 $N_i = \sum_{j=1}^C N_{ij}$ 为簇 i 的总对象数。定义 $p_{ij} = N_{ij}/N_i$; 它是簇 i 上类别标签的经验分布。定义**簇的纯度**为 $p_i = \max_j p_{ij}$, 并将**整体纯度**定义为

$$\text{purity} = \sum_i \frac{N_i}{N} p_i \quad (25.5)$$

例如, 在图 25.1 中, 我们有

$$\frac{6}{17} \cdot \frac{5}{6} + \frac{6}{17} \cdot \frac{4}{6} + \frac{5}{17} \cdot \frac{3}{5} = \frac{5+4+3}{17} = 0.71 \quad (25.6)$$

纯度取值范围为 0 (差) 到 1 (好)。然而, 我们可以把每个对象单独成簇来“轻松地”达到纯度 1, 所以该指标**不惩罚簇数**。

25.1.2.2 Rand 指数 (Rand index)

令 $U = \{u_1, \dots, u_R\}$ 与 $V = \{v_1, \dots, v_C\}$ 为对 N 个数据点的两种不同的划分 (即两种不同的**扁平聚类**)。例如, U 可以是我们估计得到的聚类, V 是由类别标签导出的参考聚类。现在定义一个 2×2 的列联表, 其中:

- TP : 在 U 和 V 中都**同簇**的对象对 (真正例) 数;
- TN : 在 U 和 V 中都**异簇**的对象对 (真反例) 数;
- FN : 在 U **异簇**、但在 V **同簇**的对象对 (假反例) 数;
- FP : 在 U **同簇**、但在 V **异簇**的对象对 (假正例) 数。

常用的汇总统计量是 **Rand 指数**:

$$R = \frac{TP + TN}{TP + FP + FN + TN} \quad (25.7)$$

它可以解释为**聚类二元判断中“判对”的比例**。显然 $0 \leq R \leq 1$ 。

例如, 考虑图 25.1。三个簇分别包含 6、6、5 个点, 因此“正例”(即不管标签、只看被分到同一簇的对象对) 数量为

$$TP + FP = \binom{6}{2} + \binom{6}{2} + \binom{5}{2} = 40 \quad (25.8)$$

其中, 真正例的数量为

$$TP = \binom{5}{2} + \binom{5}{2} + \binom{3}{2} + \binom{2}{2} = 20 \quad (25.9)$$

最后两项来自第 3 个簇: 其中同为 C 类的有 $\binom{3}{2}$ 对, 同为 A 类的有 $\binom{2}{2}$ 对。于是 $FP = 40 - 20 = 20$ 。类似可得 $FN = 24$ 且 $TN = 72$ 。因此

$$R = (20 + 72)/(20 + 20 + 24 + 72) = 0.68.$$

只有在 $TP = TN = 0$ 时 Rand 指数才会达到下界 0, 但这是罕见事件。可以定义**调整后的 Rand 指数** (adjusted Rand index, Hubert 与 Arabie, 1985) :

$$AR = \frac{\text{index} - \text{expected index}}{\max \text{index} - \text{expected index}} \quad (25.10)$$

这里的“随机性模型”基于**广义超几何分布**: 即在**保持原先的类别数与各类对象数的条件**下, 随机选择两种划分, 然后计算 $TP + TN$ 的期望值。这个模型也可用来计算 Rand 指数的**统计显著性**。

Rand 指数等权对待假正与假反。二分类问题的其他汇总统计量（比如第 5.7.2.2 节中的 F 分数）也可以用。它们的频率学采样分布以及统计显著性可以用诸如**自助法（bootstrap）**来估计。

25.1.2.3 互信息 (Mutual information)

评估聚类质量的另一种方式，是计算 U 与 V 之间的互信息（Vaithyanathan 与 Dom, 1999）。为此，令

$p_{UV}(i,j) = \frac{|u_i \cap v_j|}{N}$ 表示随机选一个对象，它同时属于 U 中的簇 u_i 和 V 中的簇 v_j 的概率。又令 $p_U(i) = |u_i|/N$ 表示随机选一个对象属于 U 中簇 u_i 的概率；类似地定义 $p_V(j) = |v_j|/N$ 。则

$$I(U, V) = \sum_{i=1}^R \sum_{j=1}^C p_{UV}(i,j) \log \frac{p_{UV}(i,j)}{p_U(i)p_V(j)} \quad (25.11)$$

它介于 0 与 $\min\{H(U), H(V)\}$ 之间。不幸的是，大量的小簇（熵低）可能达到较大的上界。为补偿这一点，我们可以使用归一化互信息（normalized mutual information）：

$$\text{NMI}(U, V) = \frac{I(U, V)}{(H(U) + H(V))/2} \quad (25.12)$$

它的取值在 0 到 1 之间。基于特定随机数据模型、校正偶然性的版本见 Vinh 等（2009）。另一种变体称为信息变差（variation of information），见 Meilă (2005)。

通俗解释（一步步讲清楚）

1) 两种“输入视角”

- 想像你在分组照片。
 - **相似度矩阵 D** : 我不看照片本身，只看“每两张照片有多不像（远）”。表格里是任意两张照片之间的距离/不相似度。优点：可以直接把“专业知识里的相似度公式/核函数”塞进去。
 - **特征矩阵 X** : 我直接看每张照片的像素/颜色直方图/文本描述等特征。优点：能直接吃原始数据，哪怕有噪声。

2) 两种“输出形态”

- **扁平聚类**: 一次性把所有对象分成 K 个不重叠的小组（像把班级分成 K 个互不交叉的小队）。
 - **层次聚类**: 先粗分，再细分，形成一棵“从粗到细”的树（像家谱）。
- 对比：扁平速度快；层次通常更有分析价值，还常常不需要你先告诉它 K 。

3) 距离/不相似度怎么量

- **欧氏距离（平方差）**：差大时惩罚更重（因为平方）。
- l_1 /曼哈顿距离（绝对值差）：对离群值更稳健。
- **相关系数**：对“形状相似”但尺度不同的向量/时间序列很有用。标准化后，“用相关做相似度”等价于“用平方距离做不相似度”。
- **有序变量**：给它们排个号（如 1/3, 2/3, 3/3），就能用上面的距离。
- **分类型变量**：相同记 0，不同记 1，逐维相加就是**汉明距离**。

4) 聚类结果怎么评估

- **内部指标**：只看数据与簇本身（例如簇内是否紧、簇间是否远）。优点：随手就算；缺点：可能不符合任务目标。
- **外部指标**：如果手里恰好有“真标签”或“参考聚类”，就把你的聚类跟它比。下面三种常见：

① 纯度 (Purity)

- 对每个簇，看看其中“占比最大的标签”占了多少（这就是该簇的纯度）。
- 再按簇大小加权平均。
- 缺点：簇越多越容易高纯度——极端地，每个样本自己一个簇，纯度=1（没意义）。

② Rand 指数 (R)

- 把“任何一对样本”看成一次二选一判断：“应不应该在同一簇里？”
- 统计四种情况：TP（两边都同簇）、TN（两边都异簇）、FP（你同簇但真异簇）、FN（你异簇但真同簇）。
- $R = (TP + TN)/\text{总对数}$ 。好理解：判对的比例。
- **调整后 Rand (ARI) **会扣掉“随机凑上的好运气”，更公平地反映质量。

③ 互信息 (MI/NMI)

- 看你的聚类 U 与参考聚类 V 之间“信息共享多少”。
- MI 越大表示越一致；但很多小簇会“拉高上限”。
- **归一化互信息 (NMI) **把 MI 除以平均熵，标准化到 0–1，更好横向比较。
- 还有变体 VI (信息变差)，是另一种比较两种划分差异的度量。

5) 为什么评估难

- 聚类是**无监督**：没有“标准答案”，所以“好不好”常常依赖任务、业务目标或外部信息。
- 即使有概率模型，我们也只能评估“似然”，这并不直接告诉你“分组是否理想”。

要点总结

- **输入两派**: 用“对象两两不相似度矩阵 D ”(便于融入领域相似度/核) 或用“特征矩阵 X ”(能吃原始含噪数据)。
- **输出两型**: 扁平(快、要选 K 、常受初始影响) vs 层次(慢些、常不需 K 、结果更直观有用)。
- **距离选择很关键**:
 - 连续数值: 欧氏(放大差异)、曼哈顿(更稳健);
 - 向量/时间序列: 相关 \leftrightarrow 平方距离等价(在标准化后);
 - 有序: 先数值化再算距;
 - 分类: 汉明距离(同 0 异 1 的和)。
- **评估聚类的三大外部指标**:
 1. **纯度**: 易懂, 但偏爱“多簇”。
 2. **Rand/ARI**: 把样本对当作二分类, ARI校正“撞大运”。
 3. **MI/NMI**: 信息论视角, NMI标准化后更可比, 另有 VI 变体。
- **本质困难**: 无监督 \rightarrow 没有标准答案; 似然不等于“好分组”。选择指标要贴合应用目标。

小贴士: 实际工作里, 常常会“多指标一起看 + 画图观察(比如 t-SNE/UMAP 可视化) + 结合业务语义”来判断聚类是否“好用”。

这一节主要在讲什么? 我不太懂

已思考 5s

没关系, 我用大白话抓住这节的“主线”和“重点”。

一句话概括

这节在讲: **怎么把相似的东西自动分组(聚类)**, 分组前需要什么样的输入、会得到什么样的输出、相似/不相似怎么量、以及分完组后怎么判断“分得好不好”。

核心内容(不带公式版)

1. **两种输入方式(分组前你手里拥有什么)**
- **相似度/距离表**: 每两样东西之间有多像(或多不像)的分数表。
 - **特征表**: 每一样东西的属性, 比如身高体重、颜色、词频等。
(前者容易融入“行业里的相似度规则”, 后者可以直接吃原始数据。)

2. 两种结果形态（分组会长什么样）

- **扁平分组**: 一次性分成 K 个不重叠的小组。快，但要先定 K 。
- **层次分组**: 像家谱，从大类逐步细分成小类。慢一些，但更直观，通常不用先定 K 。

3. 相似/不相似怎么量

- 连续数值: 用**欧氏距离** (差大更敏感) 或**曼哈顿距离** (更稳健)。
- 向量/时间序列: 用**相关性** (形状像就算相似)。
- 有序值 (低/中/高): 先映射成数字再算距离。
- 分类值 (红/绿/蓝): 一样记 0, 不一样记 1, 逐维相加 (**汉明距离**)。

4. 分完了，怎么评估好不好

- **内部看**: 看簇内是否紧、簇间是否远 (但可能不贴任务)。
- **外部对照** (有“真标签”时最常用):
 - **纯度 (Purity)**: 每个组里占最多的标签比例。容易被“多分几个组”弄得看起来很高。
 - **Rand 指数 / 调整后 Rand (ARI)**: 把“任何一对样本”当成一次判断 (该不该被分到一起)，看判对比例；ARI会扣掉“随机碰巧”的水分。
 - **互信息 / 归一化互信息 (NMI)**: 用信息论衡量你的分组与真标签的一致性，NMI标准化到 0–1，更好比较。

5. 为什么评估难

聚类是**无监督**的——很多时候没有“标准答案”，所以“好不好”必须结合业务目的或外部信息来判定。

你可以这样理解

想像给一堆歌分歌单：

- 有的人手上有“任意两首歌的相似度” (相似度/距离表)；
- 有的人只有“每首歌的属性” (节奏、情绪、歌手国籍等)。

不管怎样，你要决定是一次性分成 K 个歌单 (扁平)，还是先分成流派→再分年代→再分情绪 (层次)。

分的时候要先定“相似是什么意思” (不同数据类型用不同的“距离”)。

分完后，如果你手里还有“官方标签” (比如已知流派)，就用 Purity、ARI、NMI 这类指标衡量你分得像不像“官方答案”。

记住这几点就够了

- **这节=聚类入门地图**: 输入 (相似度表or特征表) →输出 (扁平or层次) →相似度怎么量→结果怎么评。

- 扁平快但要先定 K；层次直观通常不需要 K。
- 距离要跟数据类型匹配：数值用欧氏/曼哈顿，时间序列看相关性，有序先编码，分类看汉明。
- 评估三件套：Purity（易高估）、ARI（更公平）、NMI（信息论视角，0–1好比）。
- 无监督难在“无标准答案”：最好结合外部标签或业务目标一起评估。

如果你愿意，我可以用一个小玩具数据，现场算一遍 Purity/ARI/NMI，帮你把感觉建立起来。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

25.2 Dirichlet process mixture models

The simplest approach to (flat) clustering is to use a finite mixture model, as we discussed in Section 11.2.3. This is sometimes called model-based clustering, since we define a probabilistic model of the data, and optimize a well-defined objective (the likelihood or posterior), as opposed to just using some heuristic algorithm.

The principle problem with finite mixture models is how to choose the number of components K. We discussed several techniques in Section 11.5. However, in many cases, there is no welldefined number of clusters. Even in the simple 2d height-weight data (Figure 1.8), it is not clear if the “correct” value of K should be 2, 3, or 4. It would be much better if we did not have to choose K at all.

In this section, we discuss infinite mixture models, in which we do not impose any a priori bound on K. To do this, we will use a non-parametric prior based on the Dirichlet process (DP). This allows the number of clusters to grow as the amount of data increases. It will also prove useful later when we discuss hierarchical clustering.

The topic of non-parametric Bayes is currently very active, and we do not have space to go into details (see (Hjort et al. 2010) for a recent book on the topic). Instead we just give a brief review of the DP and its application to mixture modeling, based on the presentation in (Sudderth 2006, sec 2.2).

25.2.1 From finite to infinite mixture models

Consider a finite mixture model, as shown in Figure

25.2(a). The usual representation is as follows:

$$p(x_i|z_i = k, \theta) = p(x_i|\theta_k) \quad (25.13)$$

$$p(z_i = k|\pi) = \pi_k \quad (25.14)$$

$$p(\pi|\alpha) = \text{Dir}(\pi|\alpha/K) \quad (25.15)$$

The form of $p(\theta_k|\lambda)$ is chosen to be conjugate to $p(x_i|\theta_k)$.

We can write $p(x_i|\theta_k)$ as $x_i \sim F(\theta_k)$, where F is the observation distribution. Similarly, we can write $\theta_k \sim H(\lambda)$, where H is the prior.

An equivalent representation for this model is shown in Figure 25.2(b). Here θ_i is the parameter used to generate observation x_i ; these parameters are sampled from distribution G , which has the form

$$G(\theta) = \sum_{k=1}^K \pi_k \delta_{\theta_k}(\theta) \quad (25.16)$$

where $\pi \sim \text{Dir}(\alpha K)$, and $\theta_k \sim H$. Thus we see that G is a finite mixture of delta functions, centered on the cluster parameters θ_k . The probability that θ_i is equal to θ_k is exactly π_k , the prior probability for that cluster.

If we sample from this model, we will always (with probability one) get exactly K clusters, with data points scattered around the cluster centers. We would like a more flexible model, that can generate a variable number of clusters. Furthermore, the more data we generate, the more likely we should be to see a new cluster. The way to do this is to replace the discrete distribution G with a random probability measure. Below we will show that the Dirichlet process, denoted $G \sim DP(\alpha, H)$, is one way to do this.

Before we go into the details, we show some samples from this non-parametric model in Figure 25.3. We see that it has the desired properties of generating a variable number of clusters, with more clusters as the amount of data increases. The resulting samples look much more like real data than samples from a finite mixture model.

Of course, working with an “infinite” model sounds scary. Fortunately, as we show below, even though this model is potentially infinite, we can perform inference using an amount of computation that is not only tractable, but is often much less than that required to fit a set of finite mixture models for different K . The intuitive reason is that we can get evidence that certain values of K are appropriate (have high posterior support) long before we

have been able to estimate the parameters, so we can focus our computational efforts on models of appropriate complexity. Thus going to the infinite limit can sometimes be faster. This is especially true when we have multiple model selection problems to solve.

25.2.2 The Dirichlet process

Recall from Chapter 15 that a Gaussian process is a distribution over functions of the form $f : X \rightarrow \mathbb{R}$. It is defined implicitly by the requirement that $p(f(x_1), \dots, f(x_N))$ be jointly Gaussian, for any set of points $x_i \in X$. The parameters of this Gaussian can be computed using a mean function $\mu()$ and covariance (kernel) function $K()$. We write $f \sim GP(\mu(), K())$. Furthermore, the GP is consistently defined, so that $p(f(x_1))$ can be derived from $p(f(x_1), f(x_2)), \dots$, etc.

A Dirichlet process is a distribution over probability measures $G : \Theta \rightarrow \mathbb{R}_+$, where we require $G(\Theta) \geq 0$ and $\int G(\theta) d\theta = 1$. The DP is defined implicitly by the requirement that $(G(T_1), \dots, G(T_K))$ has a joint Dirichlet distribution $\text{Dir}(\alpha H(T_1), \dots, \alpha H(T_K))$ (25.17)

for any finite partition (T_1, \dots, T_K) of Θ . If this is the case, we write $G \sim DP(\alpha, H)$, where α is called the concentration parameter and H is called the base measure.

An example of a DP is shown in Figure 25.4, where the base measure is a 2d Gaussian. The distribution over all the cells, $p(G(T_1), \dots, G(T_K))$, is Dirichlet, so the marginals in each cell are beta distributed:

$$\text{Beta}(\alpha H(T_i), \alpha) \quad (25.18)$$

The DP is consistently defined in the sense that if T_1 and T_2 form a partition of Θ , then $G(T_1) + G(T_2)$ and $G(\Theta \setminus (T_1 \cup T_2))$ both follow the same beta distribution.

Recall that if $\pi \sim \text{Dir}(\alpha)$, and $z | \pi \sim \text{Cat}(\pi)$, then we can integrate out π to get the predictive distribution for the Dirichlet-multinoulli model:

$$z \sim \text{Cat}(\alpha_1/\alpha_0, \dots, \alpha_K/\alpha_0) \quad (25.19)$$

where $\alpha_0 = k \alpha$. In other words, $p(z = k | \alpha) = \alpha_k / \alpha_0$. Also, the updated posterior for π given one observation is given by

$$\pi | z \sim \text{Dir}(\alpha_1 + I(z=1), \dots, \alpha_K + I(z=K)) \quad (25.20)$$

The DP generalizes this to arbitrary partitions. If $G \sim DP(\alpha, H)$, then $p(\theta \in T_i) = H(T_i)$ and the posterior is $p(G(T_1), \dots, G(T_K) | \theta, \alpha, H) = \text{Dir}(\alpha H(T_1) + I(\theta \in T_1), \dots, \alpha H(T_K) + I(\theta \in T_K))$ (25.21)

This holds for any set of partitions. Hence if we observe multiple samples $\theta_i \sim G$, the new posterior is given by
 $G|\theta_1, \dots, \theta_N, \alpha, H \sim DP(\alpha + N, 1/\alpha + N) \prod_{i=1}^N \delta_{\theta_i}$ (25.22)

Thus we see that the DP effectively defines a conjugate prior for arbitrary measurable spaces. The concentration parameter α is like the effective sample size of the base measure H .

25.2.2.1 Stick breaking construction of the DP

Our discussion so far has been very abstract. We now give a constructive definition for the DP, known as the stick-breaking construction.

Let $\pi = \{\pi_k\}_{k=1}^\infty$ be an infinite sequence of mixture weights derived from the following process:

$$\beta_k \sim Beta(1, \alpha) \quad (25.23)$$

$$\pi_k = \beta_k \prod_{l=1}^{k-1} (1 - \beta_l) = \beta_k \left(1 - \prod_{l=1}^{k-1} \pi_l\right) \quad (25.24)$$

This is often denoted by

$$\pi \sim GEM(\alpha) \quad (25.25)$$

where GEM stands for Griffiths, Engen and McCloskey (this term is due to (Ewens 1990)). Some samples from this process are shown in Figure 25.5. One can show that this process will terminate with probability 1, although the number of elements it generates increases with α .

Furthermore, the size of the π_k components decreases on average.

Now define

$$G(\theta) = \sum_{k=1}^\infty \pi_k \delta_{\theta_k}(\theta) \quad (25.26)$$

where $\pi \sim GEM(\alpha)$ and $\theta_k \sim H$. Then one can show that $G \sim DP(\alpha, H)$.

As a consequence of this construction, we see that samples from a DP are discrete with probability one. In other words, if you keep sampling it, you will get more and more repetitions of previously generated values. So if we sample $\theta_i \sim G$, we will see repeated values; let us number the unique values $\theta_1, \theta_2, \dots$. Data sampled from θ_i will therefore cluster around the θ_k . This is evident in Figure 25.3, where most data comes from the Gaussians with large π_k values, represented by ellipses with thick borders. This is our first indication that the DP might be useful for clustering.

25.2.2.2 The Chinese restaurant process (CRP)

Working with infinite dimensional sticks is problematic. However, we can exploit the clustering property to draw

samples form a GP, as we now show.

The key result is this: If $\theta_i \sim G$ are N observations from $G \sim DP(\alpha, H)$, taking on K distinct values θ_k , then the predictive distribution of the next observation is given by

$$p(\theta_{N+1} = \theta | \theta_1:N, \alpha, H) = \frac{1}{\alpha + N} \alpha H(\theta) + \sum_{k=1}^K N_k \delta_{\theta_k}(\theta) \quad (25.27)$$

where N_k is the number of previous observations equal to θ_k . This is called the Polya urn or Blackwell-MacQueen sampling scheme. This provides a constructive way to sample from a DP.

It is much more convenient to work with discrete variables z_i which specify which value of θ_k to use. That is, we define $\theta_i = \theta_{z_i}$. Based on the above expression, we have

$$p(z_{N+1} = z | z_1:N, \alpha) = \frac{1}{\alpha + N} \alpha I(z = k*) + \sum_{k=1}^K N_k I(z = k) \quad (25.28)$$

where k^* represents a new cluster index that has not yet been used. This is called the Chinese restaurant process or CRP, based on the seemingly infinite supply of tables at certain Chinese restaurants. The analogy is as follows: The tables are like clusters, and the customers are like observations. When a person enters the restaurant, he may choose to join an existing table with probability proportional to the number of people already sitting at this table (the N_k); otherwise, with a probability that diminishes as more people enter the room (due to the $1/(\alpha+N)$ term), he may choose to sit at a new table k^* . The result is a distribution over partitions of the integers, which is like a distribution of customers to tables.

The fact that currently occupied tables are more likely to get new customers is sometimes called the rich get richer phenomenon. Indeed, one can derive an expression for the distribution of cluster sizes induced by this prior process; it is basically a power law. The number of occupied tables K almost surely approaches $\alpha \log(N)$ as $N \rightarrow \infty$, showing that the model complexity will indeed grow logarithmically with dataset size. More flexible priors over cluster sizes can also be defined, such as the two-parameter Pitman-Yor process.

25.2.3 Applying Dirichlet processes to mixture modeling

The DP is not particularly useful as a model for data directly, since data vectors rarely repeat exactly. However, it is useful as a prior for the parameters of a stochastic data generating mechanism, such as a mixture model. To

create such a model, we follow exactly the same setup as Section 11.2, but we define $G \sim DP(\alpha, H)$. Equivalently, we can write the model as follows:

$$\pi \sim GEM(\alpha) \quad (25.29)$$

$$z_i \sim \pi \quad (25.30)$$

$$\theta_k \sim H(\lambda) \quad (25.31)$$

$$x_i \sim F(\theta_{z_i}) \quad (25.32)$$

This is illustrated in Figure 25.6. We see that G is now a random draw of an unbounded number of parameters θ_k from the base distribution H , each with weight π_k . Each data point x_i is generated by sampling its own “private” parameter θ_i from G . As we get more and more data, it becomes increasingly likely that θ_i will be equal to one of the θ_k ’s we have seen before, and thus x_i will be generated close to an existing datapoint.

25.2.4 Fitting a DP mixture model

The simplest way to fit a DPMM is to modify the collapsed Gibbs sampler of Section 24.2.4. From Equation 24.23 we have

$$p(z_i = k | z_{-i}, x, \alpha, \lambda) \propto p(z_i = k | z_{-i}, \alpha) p(x_i | x_{-i}, z_i = k, z_{-i}, \lambda) \quad (25.33)$$

By exchangeability, we can assume that z_i is the last customer to enter the restaurant. Hence the first term is given by

$$p(z_i | z_{-i}, \alpha) = \frac{1}{\alpha + N - 1} \alpha I(z_i = k^*) + \sum_{k=1}^K \frac{N_k}{\alpha + N - 1} I(z_i = k) \quad (25.34)$$

where K is the number of clusters used by z_{-i} , and k^* is a new cluster. Another way to write this is as follows:

$$p(z_i = k | z_{-i}, \alpha) = \begin{cases} \frac{N_k}{\alpha + N - 1} & \text{if } k \text{ has been seen before} \\ \frac{\alpha}{\alpha + N - 1} & \text{if } k \text{ is a new cluster} \end{cases} \quad (25.35)$$

Interestingly, this is equivalent to Equation 24.26, which has the form $p(z_i = k | z_{-i}, \alpha) = \frac{N_k}{\alpha + N - 1} \frac{\alpha}{\alpha + N - 1}$, in the $K \rightarrow \infty$ limit (Rasmussen 2000; Neal 2000).

To compute the second term, $p(x_i | x_{-i}, z_i = k, z_{-i}, \lambda)$, let us partition the data x_{-i} into clusters based on z_{-i} . Let $x_{-i,c} = \{x_j : z_j = c, j = i\}$ be the data assigned to cluster c . If $z_i = k$, then x_i is conditionally independent of all the data points except those assigned to cluster k . Hence we have

$$p(x_i | x_{-i}, z_i = k, \lambda) = p(x_i | x_{-i}, k, \lambda) = p(x_i, x_{-i}, k | \lambda) p(x_{-i}, k | \lambda) \quad (25.36)$$

where

$$p(x_i, x_{-i}, k | \lambda) = p(x_i | \theta_k) \prod_{j \neq i, z_j=k} p(x_j | \theta_k) \int H(\theta_k | \lambda) d\theta_k \quad (25.37)$$

is the marginal likelihood of all the data assigned to cluster k , including i , and $p(x-i, k|\lambda)$ is an analogous expression excluding i . Thus we see that the term $p(x_i|x-i, z-i, z_i = k, \lambda)$ is the posterior predictive distribution for cluster k evaluated at x_i .

If $z_i = k_*$, corresponding to a new cluster, we have

$$p(x_i|x-i, z-i, z_i = k_*, \lambda) = p(x_i|\lambda) = p(x_i|\theta)H(\theta|\lambda)d\theta \quad (25.38)$$
which is just the prior predictive distribution for a new cluster evaluated at x_i .

This is very similar to collapsed Gibbs for finite mixtures except that we have to consider the case $z_i = k_*$.

An example of this procedure in action is shown in Figure 25.7. The sample clusterings, and the induced posterior over K , seems reasonable. The method tends to rapidly discover a good clustering. By contrast, Gibbs sampling (and EM) for a finite mixture model often gets stuck in poor local optima (not shown). This is because the DPMM is able to create extra redundant clusters early on, and to use them to escape local optima. Figure 25.8 shows that most of the time, the DPMM converges more rapidly than a finite mixture model.

A variety of other fitting methods have been proposed. (Daume 2007a) shows how one can use A star search and beam search to quickly find an approximate MAP estimate. (Mansinghka et al. 2007) discusses how to fit a DPMM online using particle filtering, which is a like a stochastic version of beam search. This can be more efficient than Gibbs sampling, particularly for large datasets. (Kurihara et al. 2006) develops a variational approximation that is even faster (see also (Zobay 2009)). Extensions to the case of non-conjugate priors are discussed in (Neal 2000).

Another important issue is how to set the hyper-parameters. For the DP, the value of α does not have much impact on predictive accuracy, but it does affect the number of clusters. One approach is to put a $Ga(a, b)$ prior for α , and then to from its posterior, $p(\alpha|K, N, a, b)$, using auxiliary variable methods (Escobar and West 1995).

Alternatively, one can use empirical Bayes (McAuliffe et al. 2006). Similarly, for the base distribution, we can either sample the hyper-parameters λ (Rasmussen 2000) or use empirical Bayes (McAuliffe et al. 2006).

已思考 22s

下面先给出**完整中文翻译**（逐段对应原文，保留公式与编号；对个别明显的拼写/排版小错误做了不改含义的微调并在行文中自然更正），随后是**通俗解释**，最后是**要点总结**。

译文

25.2 狄利克雷过程混合模型 (Dirichlet process mixture models)

用来做（扁平）聚类的最简单方法是使用**有限混合模型**，正如我们在第 11.2.3 节讨论的那样。这有时被称为**基于模型的聚类** (model-based clustering)，因为我们为数据定义了一个概率模型，并优化一个定义良好的目标（似然或后验），而不是仅仅使用某些启发式算法。

使用有限混合模型的首要问题是**如何选择成分个数 K** 。我们在第 11.5 节讨论了几种技术。然而，在许多情况下，并不存在一个定义良好的簇数。即使在简单的二维身高一体重数据（图 1.8）中，也不清楚“正确”的 K 应该是 2、3 还是 4。**如果我们根本不必选择 K 就更好了。**

在本节中，我们讨论**无限混合模型**，其中我们不对 K 预先施加任何上界。为此，我们将使用基于**狄利克雷过程 (DP) 的非参数先验**。这使得簇的数量可以随着数据量的增加而增长。在讨论层次聚类时这也非常有用。

非参数贝叶斯是当前非常活跃的主题，我们没有空间展开细节（见 Hjort 等, 2010, 一本较新的著作）。相反，我们仅基于 (Sudderth 2006, 第 2.2 节) 的表达，给出对 DP 及其在混合建模中应用的简要回顾。

25.2.1 从有限混合到无限混合模型 (From finite to infinite mixture models)

考虑图 25.2(a) 所示的一个有限混合模型。其通常表示如下：

$$p(x_i \mid z_i = k, \theta) = p(x_i \mid \theta_k) \quad (25.13)$$

$$p(z_i = k \mid \pi) = \pi_k \quad (25.14)$$

$$p(\pi \mid \alpha) = \text{Dir}(\pi \mid (\alpha/K)\mathbf{1}_K) \quad (25.15)$$

$p(\theta_k \mid \lambda)$ 的形式选择为与 $p(x_i \mid \theta_k)$ 共轭。我们可以写作 $x_i \sim F(\theta_{z_i})$ ，其中 F 为观测分布。类似地， $\theta_k \sim H(\lambda)$ ，其中 H 为先验。

该模型的等价表示如图 25.2(b)。此处 θ_i 是用来生成观测 x_i 的参数；这些参数从分布 G 中采样，其形式为

$$G(\theta) = \sum_{k=1}^K \pi_k \delta_{\theta_k}(\theta) \quad (25.16)$$

其中 $\pi \sim \text{Dir}(\frac{\alpha}{K} \mathbf{1})$, 且 $\theta_k \sim H$ 。因此我们看到, G 是一个**有限个以簇参数 θ_k 为中心的**狄拉克点质量 (delta) **的混合**。参数 θ_i 等于 θ_k 的概率正是 π_k , 即该簇的先验概率。

如果我们从该模型采样, 几乎必然 (概率为 1) 得到**恰好 K 个簇**, 数据点分散在若干簇中心周围。我们希望一个更灵活的模型, **能产生可变数量的簇**。并且, 生成的数据越多, **越有可能看到一个新簇**。实现这一点的方法是将离散分布 G 换成一个**随机概率测度**。下面我们将展示, **狄利克雷过程** (记作 $G \sim \text{DP}(\alpha, H)$) 就是一种实现方式。

在深入细节之前, 我们先看图 25.3 中来自这个非参数模型的若干样本。可以看到, 它确实具有“生成可变簇数”的理想性质, 并且随着数据量的增加, 簇的数量也更多。相比有限混合模型的样本, **结果看起来更像真实数据**。

当然, 与“无限”模型打交道听起来吓人。幸运的是, 正如下文所示, **尽管模型潜在是无限的**, 我们仍能用可处理的计算量进行推断, 而且往往**比为了不同 K 去拟合一堆有限混合模型所需的计算还更少**。直观原因是: 在尚未能把参数估好之前, 我们就能**较早得到“某些 K 值更合适 (有较高后验支持)”的证据**, 于是可以把计算集中在合适复杂度的模型上。因此, **取无限极限有时反而更快**。当我们需要解决多个模型选择问题时尤其如此。

25.2.2 狄利克雷过程 (The Dirichlet process)

回顾第 15 章: **高斯过程 (GP)** **是定义在函数 $f : X \rightarrow \mathbb{R}$ 上的分布**。它以这样的方式被隐式定义: 对任意点集 $x_i \in X$, $p(f(x_1), \dots, f(x_N))$ 均为联合高斯。该高斯的参数可由均值函数 $\mu(\cdot)$ 与协方差 (核) 函数 $K(\cdot)$ 计算。我们记 $f \sim \text{GP}(\mu(\cdot), K(\cdot))$ 。此外, GP 是一致定义的, 即 $p(f(x_1))$ 可由 $p(f(x_1), f(x_2))$ 推导, 等等。

狄利克雷过程 (DP) **是定义在概率测度上的分布**: $G : \Theta \rightarrow \mathbb{R}_+$, 其中要求 $G(\theta) \geq 0$ 且 $\int_{\Theta} G(\theta) d\theta = 1$ 。DP 以如下方式被隐式定义: 对 Θ 的任意有限划分 (T_1, \dots, T_K) , 随机向量

$$(G(T_1), \dots, G(T_K))$$

服从一个**Dirichlet 分布**

$$\text{Dir}(\alpha H(T_1), \dots, \alpha H(T_K)) \quad (25.17)$$

若满足此条件, 我们记 $G \sim \text{DP}(\alpha, H)$, 其中 α 称为**集中参数** (concentration parameter), H 称为**基测度** (base measure)。

一个 DP 的例子如图 25.4, 其中基测度是二维高斯。对所有小格的分布 $p(G(T_1), \dots, G(T_K))$ 是 Dirichlet, 因此每个小格的边缘分布是 beta 分布:

$$\text{Beta} \left(\alpha H(T_i), \alpha \sum_{j \neq i} H(T_j) \right) \quad (25.18)$$

DP 以一致方式定义：如果 T_1 与 T_2 构成了 \tilde{T}_1 的一个划分，那么 $G(T_1) + G(T_2)$ 与 $G(\tilde{T}_1)$ 都服从相同的 beta 分布。

回忆：若 $\pi \sim \text{Dir}(\alpha)$ ，且 $z \mid \pi \sim \text{Cat}(\pi)$ ，则我们可以积分消去 π ，得到 Dirichlet-多项式模型（又称 Dirichlet-multinoulli）的预测分布：

$$z \sim \text{Cat} \left(\frac{\alpha_1}{\alpha_0}, \dots, \frac{\alpha_K}{\alpha_0} \right), \quad \alpha_0 = \sum_k \alpha_k \quad (25.19)$$

换言之， $p(z = k \mid \alpha) = \alpha_k / \alpha_0$ 。同时，给出一个观测后的 π 的后验更新为

$$\pi \mid z \sim \text{Dir}(\alpha_1 + \mathbf{I}(z=1), \dots, \alpha_K + \mathbf{I}(z=K)) \quad (25.20)$$

DP 将此推广到任意划分。若 $G \sim \text{DP}(\alpha, H)$ ，则 $p(\theta \in T_i) = H(T_i)$ ，并且后验为

$$p(G(T_1), \dots, G(T_K) \mid \theta, \alpha, H) = \text{Dir}(\alpha H(T_1) + \mathbf{I}(\theta \in T_1), \dots, \alpha H(T_K) + \mathbf{I}(\theta \notin T_K))$$

这对任何划分都成立。因此，若我们观测到多次样本 $\theta_i \sim G$ ，新的后验为

$$G \mid \theta_1, \dots, \theta_N, \alpha, H \sim \text{DP}(\alpha + N, \frac{1}{\alpha + N} (\alpha H + \sum_{i=1}^N \delta_{\theta_i})) \quad (25.22)$$

于是我们看到，DP 实质上为任意可测空间定义了一个共轭先验。集中参数 α 类似于基测度 H 的“有效样本量”。

25.2.2.1 DP 的“折棒” (stick breaking) 构造

到目前为止讨论很抽象。下面给出 DP 的一个构造性定义，称为折棒构造。

令 $\pi = \{\pi_k\}_{k=1}^\infty$ 为由以下过程得到的一串无限混合权重：

$$\beta_k \sim \text{Beta}(1, \alpha) \quad (25.23)$$

$$\pi_k = \beta_k \prod_{l=1}^{k-1} (1 - \beta_l) = \beta_k \left(1 - \sum_{l=1}^{k-1} \pi_l \right) \quad (25.24)$$

通常记作

$$\pi \sim \text{GEM}(\alpha) \quad (25.25)$$

其中 GEM 是 Griffiths、Engen 与 McCloskey 的缩写（术语来自 Ewens, 1990）。该过程中抽样的一些样例见图 25.5。可以证明，这个过程以概率 1 终止（即剩余质量最终可忽

略) , 尽管它生成的分量数量会随着 α 增大而增多。此外, π_k 的大小平均上递减。

现在定义

$$G(\theta) = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}(\theta) \quad (25.26)$$

其中 $\pi \sim \text{GEM}(\alpha)$, $\theta_k \sim H$ 。则可以证明 $G \sim \text{DP}(\alpha, H)$ 。

由该构造可知, DP 的样本几乎必然是离散的。换句话说, 如果你不断从中采样, 会一次次得到已出现过的取值。因此若我们采样 $\theta_i \sim G$, 将会看到重复取值; 把唯一值记为 $\theta_1, \theta_2, \dots$ 。由 θ_i 采样的数据因此会围绕这些 θ_k 形成簇。这在图 25.3 中很明显: 大部分数据来自权重较大的高斯 (π_k 大), 它们以边框较粗的椭圆表示。这是 DP 可能用于聚类的第一个信号。

25.2.2.2 中华餐馆过程 (CRP)

处理“无限维的折棒”不太方便。然而, 我们可以利用其聚类性来采样 (注: 原文一处笔误写作“from a GP”, 此处应为 from a DP) , 如下所示。

关键结果是: 若 $\theta_i \sim G$ 是来自 $G \sim \text{DP}(\alpha, H)$ 的 N 个观测, 且它们取了 K 个不同的值 θ_k , 则下一个观测的预测分布为

$$p(\theta_{N+1} = \theta \mid \theta_{1:N}, \alpha, H) = \frac{1}{\alpha + N} (\alpha H(\theta) + \sum_{k=1}^K N_k \delta_{\theta_k}(\theta)) \quad (25.27)$$

其中 N_k 是先前观测中等于 θ_k 的数量。这称为 Polya 抽签 (urn) 或 Blackwell–MacQueen 采样方案, 它提供了从 DP 采样的构造性方法。

使用离散变量 z_i 来指明所用的 θ_k 会更方便。即定义 $\theta_i = \theta_{z_i}$ 。基于上式, 有

$$p(z_{N+1} = z \mid z_{1:N}, \alpha) = \frac{1}{\alpha + N} (\alpha \mathbf{I}(z = k^*) + \sum_{k=1}^K N_k \mathbf{I}(z = k)) \quad (25.28)$$

其中 k^* 表示一个尚未使用过的新簇索引。这被称为中华餐馆过程 (CRP) , 基于某些中餐馆“似乎有无限多的桌子”的比喻。类比如下: 桌子=簇, 顾客=观测。当一个人进店时, 他会以与当前桌上人数 N_k 成比例的概率加入一个已有桌; 否则, 他也可能以一个随着顾客变多而递减的概率 (因为分母有 $1/(\alpha + N)$) 选择坐到新桌 k^* 。其结果是一个对整数划分的分布, 好比把顾客分配到桌子上。

“富者愈富”现象: 已经有人坐的桌子更容易吸引新顾客。的确, 可以推出该先验过程诱导的簇大小分布的公式; 它基本是一个幂律。占用桌子的数量 K 以几乎处处的方式收敛到

$\alpha \log N$ (当 $N \rightarrow \infty$)，这表明模型复杂度确实会随数据规模按对数增长。亦可定义更灵活的簇大小先验，例如两参数 Pitman–Yor 过程。

25.2.3 将狄利克雷过程用于混合建模 (Applying Dirichlet processes to mixture modeling)

DP 本身并不特别适合直接作为数据的模型，因为数据向量很少完全重复。然而，它非常适合作为随机数据生成机制（如混合模型）中参数的先验。要构造这样的模型，我们沿用第 11.2 节的设定，但令 $G \sim DP(\alpha, H)$ 。等价地，可以这样写模型：

$$\pi \sim GEM(\alpha) \quad (25.29)$$

$$z_i \sim \pi \quad (25.30)$$

$$\theta_k \sim H(\lambda) \quad (25.31)$$

$$x_i \sim F(\theta_{z_i}) \quad (25.32)$$

如图 25.6 所示。我们看到， G 现在是从基分布 H 随机抽出无界多个参数 θ_k 的一个样本，每个带有权重 π_k 。每个数据点 x_i 通过从 G 中采样其“私有”的参数 θ_i 而生成。随着数据越来越多， θ_i 等于先前已见过的某个 θ_k 的概率会越来越大，因此 x_i 将会在已存在的数据点附近被生成。

25.2.4 拟合 DPMM (Fitting a DP mixture model)

拟合 DPMM 的最简单方法是修改第 24.2.4 节的折叠 Gibbs 采样器 (collapsed Gibbs)。由式 (24.23) 我们有

$$p(z_i = k | z_{-i}, \mathbf{x}, \alpha, \lambda) \propto p(z_i = k | z_{-i}, \alpha) \times p(x_i | \mathbf{x}_{-i}, z_i = k, z_{-i}, \lambda) \quad (25.33)$$

由于可交换性，我们可假设 z_i 是最后一个进入餐馆的顾客。因此第一项为

$$p(z_i = k | z_{-i}, \alpha) = \frac{1}{\alpha + N - 1} (\alpha \mathbf{I}(z_i = k^*) + \sum_{k=1}^K N_{k,-i} \mathbf{I}(z_i = k)) \quad (25.34)$$

其中 K 是 z_{-i} 使用的簇数， k^* 是一个新簇。等价地可写成如下分段形式：

$$p(z_i = k | z_{-i}, \alpha) = \begin{cases} \frac{N_{k,-i}}{\alpha + N - 1}, & \text{若 } k \text{ 先前出现过} \\ \frac{\alpha}{\alpha + N - 1}, & \text{若 } k \text{ 是新簇} \end{cases} \quad (25.35)$$

有趣的是，这与式 (24.26) 等价，而后者的形式是

$p(z_i = k | z_{-i}, \alpha) = (N_{k,-i} + \alpha/K)/(\alpha + N - 1)$ ，在 $K \rightarrow \infty$ 极限下成立 (Rasmussen 2000; Neal 2000)。

要计算第二项 $p(x_i \mid \mathbf{x}_{-i}, z_i = k, z_{-i}, \lambda)$, 我们按 z_{-i} 将 \mathbf{x}_{-i} 分簇。令 $\mathbf{x}_{-i,c} = \{x_j : z_j = c, j \neq i\}$ 为分到簇 c 的数据。若 $z_i = k$, 则在给定其他信息时, x_i 与除簇 k 中的点以外的所有点**条件独立**。因此

$$p(x_i \mid \mathbf{x}_{-i}, z_{-i}, z_i = k, \lambda) = p(x_i \mid \mathbf{x}_{-i,k}, \lambda) = \frac{p(x_i, \mathbf{x}_{-i,k} \mid \lambda)}{p(\mathbf{x}_{-i,k} \mid \lambda)} \quad (25.36)$$

其中

$$p(x_i, \mathbf{x}_{-i,k} \mid \lambda) = \int_{j \neq i: z_j=k} (p(x_i \mid \theta_k) \prod_{j \neq i: z_j=k} p(x_j \mid \theta_k)) H(\theta_k \mid \lambda) d\theta_k \quad (25.37)$$

是将 i 包含在内时簇 k 的边际似然; 而 $p(\mathbf{x}_{-i,k} \mid \lambda)$ 是不含 i 的对应表达。由此可见, $p(x_i \mid \mathbf{x}_{-i}, z_{-i}, z_i = k, \lambda)$ 就是簇 k 的**后验预测分布**在 x_i 处的评估值。

如果 $z_i = k^*$ (对应新簇), 则

$$p(x_i \mid \mathbf{x}_{-i}, z_{-i}, z_i = k^*, \lambda) = p(x_i \mid \lambda) = \int p(x_i \mid \theta) H(\theta \mid \lambda) d\theta \quad (25.38)$$

这就是**新簇的先验预测分布**在 x_i 处的评估值。

这与有限混合的折叠 Gibbs **非常相似**, 只是我们需要额外考虑 $z_i = k^*$ 的情形。

该过程的一个实际例子见图 25.7。采样得到的聚类以及诱导出的关于 K 的后验看起来是合理的。该方法倾向于**快速地发现一个好的聚类**。相比之下, 有限混合模型的 Gibbs (以及 EM) 常常卡在较差的**局部最优** (图中未示)。原因在于: **DPMM 能在早期创建一些冗余的簇**, 并用它们**跳出局部最优**。图 25.8 显示, 大多数时候, DPMM 收敛更快于有限混合模型。

还有许多其他拟合方法被提出。(Daumé, 2007a) 展示了如何使用 A* 搜索与**束搜索 (beam search)** **快速找到近似 MAP 估计**。(Mansinghka 等, 2007) 讨论了如何用**粒子滤波** (类似随机版束搜索) **在线**拟合 DPMM; 对大数据集这可能比 Gibbs 更高效。

(Kurihara 等, 2006) 发展了一个**变分近似**, 更快 (另见 Zobay, 2009)。**非共轭先验**情形的扩展见 Neal (2000)。

另一个重要问题是**如何设定超参数**。对 DP 而言, α 的取值对预测精度影响不算大, 但**会影响簇数**。一种方法是给 α 放一个 $Ga(a, b)$ 先验, 然后用**辅助变量法** (Escobar 和 West, 1995) 从其后验 $p(\alpha \mid K, N, a, b)$ 进行采样。或者可以用**经验贝叶斯** (McAuliffe 等, 2006)。类似地, 对基分布, 也可以对超参数 λ 做采样 (Rasmussen 2000), 或者使用**经验贝叶斯** (McAuliffe 等, 2006)。

通俗解释（尽量不靠公式，把直觉说清）

这节解决的问题：有限混合模型要你事先告诉它“要分成 K 组”。可很多真实数据没有一个固定的 K 。怎么办？——用狄利克雷过程混合模型 (DPMM)，让模型自己随着数据多起来动态长出合适数量的簇。

1) 从“要先定 K ”到“不用定 K ”

- **有限混合：**像办活动要先准备 K 张桌子（簇），每张桌子的位置（参数 θ_k ）和大小（权重 π_k ）都要定。问题是，我到底该摆几张桌子？
- **无限混合 / DP：**假装有**无限多张桌子**（潜在簇），但**并不会真的用完**。随着人（数据）越来越多，可能**开新桌**，也可能**坐旧桌**。这样，“用多少桌子（簇）”是由数据顺势决定的，而不是事先拍脑袋定。

2) DP 是什么——两个比喻

- **折棒 (Stick-Breaking) :** 把一根 1 米长的棒（总概率 1）拿来分。第一次掰一段 β_1 ($Beta(1, \alpha)$)，这段长度是第一簇的权重 π_1 ；剩下的再掰 β_2 的比例，得到第二簇的 π_2 ；以此类推。 α 越大，掰得越碎，簇越多越均匀； α 越小，前几段更大，往后很快变得很小。
- **中华餐馆 (CRP) :** 客人（数据点）进店。已有桌子越热闹 (N_k 越大)，越吸引新客人加入（富者愈富）；同时也有一定概率**开新桌**（概率与 α 成比例）。结果就是：簇数会随数据量 N 以大约 $\alpha \log N$ 的速度增加，而不是固定死的。

小结：这两个视角是等价的：折棒告诉你“权重如何产生”；餐馆告诉你“分配规则是什么”。

3) 把 DP 嵌进混合模型：DPMM

- 标准混合是：“先选一个簇 z_i （按权重 π 抽），再从该簇的分布 $F(\theta_{z_i})$ 里生成数据 x_i 。”
- DPMM 的不同是：**簇参数集合** $\{\theta_k\}$ 不是固定数量的，而是从 DP 中来，即“有无穷多潜在簇，每个簇一个参数 θ_k 和一个权重 π_k ”。
- 随着数据渐多，“选择新簇”的机会始终存在，但会按 CRP 的规则逐渐变小；更多时候是**复用**已有簇，因此数据会**自然地簇成团**。

4) 怎么训练（拟合）：折叠 Gibbs 的直觉

- 对每个点 x_i ，我们问两个问题：
 - A. **先验上**，它更可能去**已有簇**（按 $N_{k,-i}$ 比例）还是**新簇**（按 α 比例）？
 - B. **似然上**，它去某个簇后，这个簇对它的**解释力（后验预测）**有多强？

- 把 A 和 B 的信息相乘（式 25.33），就是把 x_i 重新分配到各个（含“新簇”）选项的**对权重**。对每个点轮流更新，如此循环，就能得到一个“自适应长出新簇/合并旧簇”的聚类。
- 经验观察：DPMM 经常**更快**找到好解，因为它可以**先多开几桌**跳出局部最优，再慢慢把多余的桌子“冷落”下去；而有限混合的 EM/Gibbs 容易被“固定的 K ”与“糟糕的初始化”困住。

5) α 和超参数怎么设

- α 主要影响簇的个数：大→簇多，小→簇少；对**预测精度**的影响通常不大。
 - 可以给 α 一个 Gamma 先验 $Ga(a, b)$ ，用**辅助变量抽样更新**；或者用**经验贝叶斯**（先用数据估一估）。
 - 基分布 H 的超参 λ 也可以抽样或用经验贝叶斯估计。
 - 若 H 与 F 是**共轭的**（比如正态-正态-逆伽马等），推断会**简洁高效**；非共轭时可以用 Neal (2000) 的方法、变分、粒子滤波等。
-

要点总结

- **目标：**避免手动选 K 。让模型（DPMM）**自适应地**决定簇数，并随数据量按 $\alpha \log N$ 的速度增长。
- **核心工具：****狄利克雷过程**（DP）
 - **折棒：** π_k 由反复“掰”出，前大后小； α 控制“碎不碎、多少簇”。
 - **餐馆：**已有簇越大越易吸引新点；也有概率开新簇（富者愈富）。
- **DPMM 结构：** $\pi \sim GEM(\alpha)$, $z_i \sim \pi$, $\theta_k \sim H$, $x_i \sim F(\theta_{z_i})$ 。**簇的数量不设上限**。
- **推断直觉：**对每个点，权衡“先验上坐旧桌还是开新桌”（CRP）与“该簇对它解释得好不好”（后验预测）；折叠 Gibbs 只多了“新簇”这个选项。
- **优势：**常比“多跑若干个 K 的有限混合”更省事；也较不易陷入局部最优。
- **α 的作用：**主要控制**簇数倾向**，对预测影响小；可用 Gamma 先验或经验贝叶斯估计。
- **工程选项：**除了 Gibbs，还有 A* / beam 近似 MAP、**粒子滤波在线方法**、**变分方法**，规模大时更高效；非共轭可用 Neal 等扩展招式。

一句话：DPMM = “**不用拍脑袋定 K 的混合模型**”。它让“有多少簇、每个簇多大”顺着数据自然长出来。需要我用一个小玩具数据做一遍 CRP 的抽样演示，帮你把感觉彻底建立起来吗？

这一节主要在讲什么？我不太懂

已思考 10s

没事~我用大白话把 25.2 的主线拎出来，最后给你一张“口袋卡片”。

一句话版

这节讲的是：**不用事先选簇数 K** 的聚类办法——**狄利克雷过程混合模型（DPMM）**。它用“**狄利克雷过程（DP）**”这类非参数先验，让模型**自己**根据数据多少**长出合适数量的簇**，并教你它的直觉（两个比喻：**折棒和中华餐馆**）、生成过程、怎么做推断（主要是**折叠 Gibbs**），以及超参数 α 怎么影响簇的个数。

核心脉络（通俗解释）

1. 为什么要 DPMM？

传统“有限混合模型”做聚类得**先选 K** （分几组），但现实里 K 常常不明确。DPMM 直接绕过这一步：**簇的数量不设上限**，由数据决定。

2. DP 的两种直觉

- **折棒（Stick-Breaking）**：把一根长度为 1 的“概率棒”不断掰断，掰出的每段就是一个簇的权重 π_k 。 α 控制“掰得碎不碎”： α 大→段数多、权重更平均； α 小→前几段大、后面很快变小。
- **中华餐馆过程（CRP）**：客人（数据点）进店，要么坐**已有的热桌**（已有的人越多越容易吸引你，叫“富者愈富”），要么**开新桌**（概率与 α 成比例）。**数据越多，桌子数（簇数）大致按 $\alpha \log N$ 增长**。

3. DPMM 的生成故事（怎么“造数据”）

- 先用 DP 产生无限多组“簇参数” θ_k 和权重 π_k （你可以想成“无限多张潜在桌子”）；
- 对每个数据点：先抽一个簇标号 z_i （选桌），再按这个簇的分布 $F(\theta_{z_i})$ 生成 x_i 。

4. 怎么训练（推断）

用**折叠 Gibbs**做迭代更新：对每个点 x_i

- **先验部分（CRP）**：更偏向去“人多的旧簇”（按当前该簇人数 $N_{k,-i}$ ），但也有一定概率去**新簇**（权重与 α 有关）；
- **数据适配度**：看这个簇对 x_i 的**解释力**（后验预测概率）。

把两者相乘得到“把 x_i 放到每个簇（含新簇）”的相对概率。循环更新，模型会**自适应地开/并簇**。经验上，DPMM常比“试好多个 K ”的有限混合更快找到好解，因为它能**先多开几桌**来摆脱局部最优，再自然收敛。

5. α 和超参数

- α **主要影响簇数倾向**（大→簇多，小→簇少），对预测精度影响通常不大；常给它放个 Gamma 先验或用经验贝叶斯来估。
- 基分布 H 的超参也可采样或经验估计。非共轭时可用 Neal 的招式、变分、粒子滤波等近似法。

何时用、该注意什么

- **适用：**你不想拍脑袋设定 K ，希望簇数随数据自然增长；数据规模可能逐步增加（在线场景）。
- **注意：** α 不是“越大越好”或“越小越好”，而是“你希望簇更细/更粗”的**先验开关**；另外选对 H （比如高斯簇就用高斯先验的共轭配置）能让推断更稳更快。

要点速记（口袋卡片）

- DPMM = **不用事先定 K 的混合模型。**
- DP **两面**：折棒给出权重；CRP给出“坐哪桌”的规则（富者愈富 + 可能开新桌）。
- **推断逻辑**：旧簇人数（先验偏好） \times 簇对样本的解释力（似然/后验预测）。
- α 控簇数倾向，预测影响小；可采样或经验估计。
- **常更稳更快**：能先“多开几桌”跳出局部最优，再慢慢收敛。

如果你愿意，我也可以用一个极小的玩具数据，演示“CRP 抽座位 + 折叠 Gibbs”全过程，帮助把直觉彻底落地。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

25.3 Affinity propagation

Mixture models, whether finite or infinite, require access to the raw $N \times D$ data matrix, and need to specify a generative model of the data. An alternative approach takes as input an $N \times N$ similarity matrix, and then tries to identify exemplars, which will act as cluster centers. The K-medoids or K-centers algorithm (Section 14.4.2) is one approach, but it can suffer from local minima. Here we describe an alternative approach called affinity propagation (Frey and Dueck 2007) that works substantially better in practice.

The idea is that each data point must choose another data

point as its exemplar or centroid; some data points will choose themselves as centroids, and this will automatically determine the number of clusters. More precisely, let $c_i \in \{1, \dots, N\}$ represent the centroid for datapoint i .

The goal is to maximize the following function

$$S(c) = \sum_{i=1}^N s(i, c_i) + \sum_{k=1}^K \delta_k(c) \quad (25.39)$$

The first term measures the similarity of each point to its centroid. The second term is a penalty term that is $-\infty$ if some data point i has chosen k as its exemplar (i.e., $c_i = k$), but k has not chosen itself as an exemplar (i.e., we do not have $c_k = k$).

More formally, $\delta_k(c) = -\infty$ if $c_k = k$ but $\exists i : c_i = k \neq c_k$ otherwise (25.40)

The objective function can be represented as a factor graph. We can either use N nodes, each with N possible values, as shown in Figure 25.9, or we can use N^2 binary nodes (see (Givoni and Frey 2009) for the details). We will assume the former representation.

We can find a strong local maximum of the objective by using max-product loopy belief propagation (Section 22.2). Referring to the model in Figure 25.9, each variable node c_i sends a message to each factor node δ_k . It turns out that this vector of N numbers can be reduced to a scalar message, denote $r_i \rightarrow k$, known as the responsibility. This is a measure of how much i thinks k would make a good exemplar, compared to all the other exemplars i has looked at. In addition, each factor node δ_k sends a message to each variable node c_i . Again this can be reduced to a scalar message, $a_i \leftarrow k$, known as the availability. This is a measure of how strongly k believes it should be an exemplar for i , based on all the other data points k has looked at.

As usual with loopy BP, the method might oscillate, and convergence is not guaranteed.

However, by using damping, the method is very reliable in practice. If the graph is densely connected, message passing takes $O(N^2)$ time, but with sparse similarity matrices, it only takes $O(E)$ time, where E is the number of edges or non-zero entries in S .

The number of clusters can be controlled by scaling the diagonal terms $S(i, i)$, which reflect how much each data point wants to be an exemplar. Figure 25.10 gives a

simple example of some 2d data, where the negative Euclidean distance was used to measured similarity. The $S(i, i)$ values were set to be the median of all the pairwise similarities. The result is 3 clusters. Many other results are reported in (Frey and Dueck 2007), who show that the method significantly outperforms K-medoids.

已思考 17s

下面先给出**完整中文翻译**（逐段对应原文，保留公式与编号；对明显的排版/符号小错误做了不改含义的更正，并在相应处标注），随后是**通俗解释**，最后是**要点总结**。

译文

25.3 亲和传播 (Affinity propagation)

无论是有限还是**无限**的混合模型，都需要访问原始的 $N \times D$ 数据矩阵，并需要为数据指定一个生成式模型。另一种做法是把一个 $N \times N$ 的**相似度矩阵**作为输入，然后尝试识别**代表点** (exemplar)，它们将充当簇中心。第 14.4.2 节的 K-medoids 或 K-centers 算法就是一种方法，但它可能会遭遇**局部极小值**。这里我们描述一种称为**亲和传播** (affinity propagation, Frey 和 Dueck, 2007) 的方法，实践中表现要好得多。

其想法是：**每个数据点必须选择另一个数据点作为它的代表点/质心**；有些数据点会**选择自己作为质心**，而这将**自动决定簇的数量**。更准确地，让 $c_i \in \{1, \dots, N\}$ 表示样本 i 的质心。

目标是最大化下面的函数

$$S(c) = \sum_{i=1}^N s(i, c_i) + \sum_{k=1}^N \delta_k(c) \quad (25.39)$$

第一项度量了**每个点与其质心的相似度**。第二项是一个惩罚项：当存在某个数据点 i 选择了 k 作为它的代表点（即 $c_i = k$ ），但 k **没有选择自己作为代表点**（即**没有** $c_k = k$ ）时，该惩罚项为 $-\infty$ 。

更正式地（原文此处有显然的号误；应为“ $c_k \neq k$ ”而非“ $c_k = k$ ”）：

$$\delta_k(c) = \begin{cases} -\infty, & \text{若 } c_k \neq k \text{ 且 } \exists i : c_i = k \\ 0, & \text{否则} \end{cases} \quad (25.40)$$

该目标函数可以表示成一个**因子图**。我们可以用 N 个取值域为 N 的节点来表示（如图 25.9），也可以用 N^2 个二值节点来表示（细节见 Givoni 和 Frey, 2009）。我们假设采用

前一种表示。

我们可以通过max-product 的**有环信念传播**（第 22.2 节）来找到该目标的一个很强的**局部极大值**。参照图 25.9 的模型，每个变量节点 c_i 向每个因子节点 δ_k 发送信息。结果发现，这个由 N 个数字构成的向量可以压缩为一个**标量信息**，记作 $r_{i \rightarrow k}$ ，称为**责任 (responsibility)**。它衡量“样本 i 认为 k 相对于它考察过的**其他代表点**来说，作为代表点有多合适”。此外，每个因子节点 δ_k 也向每个变量节点 c_i 发送信息，同样可以压缩为一个**标量信息**，记作 $a_{i \leftarrow k}$ ，称为**可用度 (availability)**。它衡量“基于 k 考察过的**所有数据点**， k 觉得自己作为 i 的代表点的可信程度”。

和一般的有环 BP 一样，该方法可能会**振荡**，且**不保证收敛**。不过，通过使用**阻尼 (damping)**，该方法在实践中非常可靠。如果图是**稠密连接的**，消息传递的时间是 $O(N^2)$ ；但如果相似度矩阵是**稀疏的**，时间复杂度仅为 $O(E)$ ，其中 E 是 S 中边或非零项的数量。

簇的数量可以通过**缩放对角项** $S(i, i)$ 来控制，它们反映了每个数据点“想成为代表点”的程度。图 25.10 给出了一个 2D 数据的简单例子，此处用**负欧氏距离**来度量相似度。 $S(i, i)$ 的取值被设置为**所有两两相似度的中位数**。结果得到 3 个簇。Frey 和 Dueck (2007) 报告了更多结果，并显示该方法**明显优于 K-medoids**。

通俗解释（尽量不靠公式，把直觉说清）

这节解决的问题：

不想写生成式模型、也不想动原始 $N \times D$ 特征？那就直接拿到“一对一的相似度表” S 来自动挑“代表点”当中心，并且**不用事先定 K**。这就是**亲和传播 (AP)**。

1) AP 的目标在干嘛？

- 每个点 i 都要选一个“代表点” c_i （可以是别人，也可以是自己）。
- 总得分 = “我和我代表点的相似度”之和 + 一个**逻辑惩罚**：不允许出现“大家都把 k 当代表点，但 k 自己却没站出来”（这会被罚到 $-\infty$ ）。
- 这就逼着**真正的代表点必须自荐**（自己选自己），由此**自动决定有多少个代表点**（也就是多少个簇）。

2) 两种信息：责任 r 与可用度 a (像拉锯)

- **责任 $r_{i \rightarrow k}$** ：从“样本 i ”看，“让 k 当我代表点”究竟比“让别的候选当代表点”好多少？
- **可用度 $a_{i \leftarrow k}$** ：从“候选中心 k ”看，“我 (k) 有多大把握/资格去当 i 的代表点”？（要参考其他样本的意见，因为一个中心不能“同时被无限多样本强占”，会相互挤压。）

- 算法在图里来回传这两类消息（有环 BP 的 max-product 版本），逐步达成一致：哪些点该自荐当中心、哪些点归属到哪个中心。

3) 为什么会振荡？阻尼怎么帮忙？

- 有环网络里大家互相影响，容易出现“你推我搡”的来回反复（振荡）。
- **阻尼**就是“别一下把新消息全信了”，而是用“新旧消息的加权平均”来更新，通常就稳了。

4) 算力与稀疏

- 如果每对点都有相似度（图是稠密的），消息数是 $O(N^2)$ 。
- 若相似度图是稀疏的（只连近邻），只在有边的地方传消息，复杂度降到 $O(E)$ 。

5) 怎么“调 K”

- $S(i, i)$ 是**自荐分**：数值越大，点 i 越“想”当中心。
- 把所有 $S(i, i)$ 整体抬高，就会多出一些中心（ K 变大）；整体压低，中心就会少。
- 常用做法：把 $S(i, i)$ 设为“所有两两相似度的**中位数**”，在很多数据上会得到一个合理的簇数（文中的示例得到 3 簇）。

6) 为什么比 K-medoids 更稳？

- K-medoids 要先定 K ，且容易卡在局部最小。
- AP 通过全图的消息传递“全局协调”谁当中心、谁跟谁，更不容易被差的初始化束缚，实践里常更好用。

要点总结

- **亲和传播 (AP)**：直接用相似度矩阵做聚类，**自动挑代表点当中心，无需预设 K**。
- **目标函数**：最大化“样本对代表点的相似度总和”，并用 $-\infty$ 惩罚强制“**被别人选为中心的人必须自荐**”。
- **两类消息**：
 - **责任** $r_{i \rightarrow k}$ ：样本 i 觉得“让 k 做我中心”的**相对好处**。
 - **可用度** $a_{i \leftarrow k}$ ：候选中心 k 觉得“我去当 i 的中心”的**可信度**。
 二者迭代更新（有环 BP 的 max-product），配合**阻尼**以防振荡。
- **复杂度**：稠密图 $O(N^2)$ ；稀疏图 $O(E)$ 。
- **调簇数的旋钮**：调节对角项 $S(i, i)$ （自荐分）。常把它们设为两两相似度的**中位数**能给出合适的 K 。
- **效果**：实践中常显著优于 K-medoids，更不易陷入局部极小。

小提示：如果你愿意，我可以拿一小块二维数据，给你直观展示 r / a 的迭代过程，以及调 $S(i, i)$ 如何改变簇数。

这一节主要在讲什么？我不太懂

已思考 6s

没事～我用大白话把 25.3 的主线拎出来，读完就懂它在干嘛。

一句话版

亲和传播 (Affinity Propagation, AP) 是一种只需要“相似度矩阵”的聚类方法：它让每个点在所有点里“票选代表点 (exemplar)”，通过两种消息来回传递，**自动挑出中心并决定簇数**，不必事先指定 K 。

用生活比喻讲原理

- **场景：**班里同学要推“组长”。每个人都可以把任何同学（包括自己）当候选。
- **规则：**如果很多人把某个同学 k 当组长，那 k 必须自己也愿意当（自荐），否则这票作废（原文里的 $-\infty$ 惩罚就是强制这条规则）。
- **两种消息来回传：**
 - **责任 r (我→你)：** 我（样本 i ）觉得“你（候选 k ）当我组长”比其他人当组长好多少？
 - **可用度 a (你→我)：** 我（候选 k ）基于大家的选择，觉得“我有多合适/有余力接收你 i ”？
- 这两类消息在全体同学之间**迭代更新**（像不断拉锯、商量），最后稳定：一些人**自荐成功**成了中心（代表点），其他人各自归属最近的中心，**簇数也就自然定了**。

为什么它好用

- **不需要生成式模型**、不看原始特征 X ，只要**相似度矩阵 S** （比如用“负欧氏距离”当相似度）。
 - **不用先定 K** ：谁自荐成功、多少人自荐成功，全靠数据“协商”出来。
 - **调簇数很直观**：调对角元素 $S(i, i)$ （叫“偏好/自荐分”）
 - $S(i, i)$ 抬高 → 更多人愿意当中心 → 簇变多；
 - $S(i, i)$ 压低 → 更少人当中心 → 簇变少。
- 常见做法：把 $S(i, i)$ 设成**所有两两相似度的中位数**，常能得到合理的簇数。

- **收敛稳定性**: 有时会**振荡**, 但用**阻尼 (damping) **就常能稳住。
- **效率**: 相似度是稠密图时消息传递约 $O(N^2)$; 若你有**稀疏相似图**, 只在有边处传, 成本降到 $O(E)$ 。

何时用它

- 你有或更容易得到**相似度**而不是原始特征;
- 不想/不便写数据的**生成模型**;
- 不想先定 K , 希望**自动决定簇数**;
- K-medoids/K-centers 老卡在局部最优时, AP 往往表现更好。

要点清单

- **目标**: 让每个点选代表点, 且**被选为代表的人必须自荐**。
- **核心机制**: 责任 r (点→候选的相对好处) 与可用度 a (候选→点的接纳程度) 反复消息传递。
- **不需 K** : 簇数由谁自荐成功自动决定。
- **调参旋钮**: 对角偏好 $S(i, i)$ 决定“谁更可能当中心”。
- **计算**: 稠密 $O(N^2)$, 稀疏 $O(E)$; 用阻尼抑制振荡。
- **效果**: 实践里**常优于 K-medoids**, 更不依赖初始化。

如果你想, 我可以用一小撮 2D 点, 现场演示调 $S(i, i)$ 如何把 2、3、4 个簇“拔出来”, 更直观。

翻译并解释以下内容, 翻译不能有遗漏, 解释请通俗易懂且详尽, 因为我的数学基础不太好, 最后总结要点。

25.4 Spectral clustering

An alternative view of clustering is in terms of graph cuts. The idea is we create a weighted undirected graph W from the similarity matrix S , typically by using the nearest neighbors of each point; this ensures the graph is sparse, which speeds computation. If we want to find a partition into K clusters, say A_1, \dots, A_K , one natural criterion is to minimize

$$\text{cut}(A_1, \dots, A_K) = \sum_{k=1}^K W(A_k, \bar{A}_k) \quad (25.41)$$

where $\bar{A}_k = V \setminus A_k$ is the complement of A_k , and $W(A, B) = \sum_{i \in A, j \in B} w_{ij}$. For $K = 2$ this problem is easy to solve.

Unfortunately the optimal solution often just partitions off a single data point from the rest. To ensure the sets are reasonably large, we can define the normalized cut to be

$$Ncut(A_1, \dots, A_K) = \frac{1}{2} \sum_{k=1}^K \frac{\text{cut}(A_k, \bar{A}_k)}{\text{vol}(A_k)} \quad (25.42)$$

where $\text{vol}(A) = \sum_{i \in A} d_i$, and $d_i = \sum_{j=1}^N w_{ij}$ is the weighted degree of node i . This splits the graph into K clusters such that nodes within each cluster are similar to each other, but are different to nodes in other clusters.

We can formulate the Ncut problem in terms of searching for binary vectors $c_i \in \{0, 1\}^N$, where $c_{ik} = 1$ if point i belongs to cluster k , that minimize the objective.

Unfortunately this is NP-hard (Wagner and Wagner 1993). Affinity propagation is one way to solve the problem. Another is to relax the constraints that c_i be binary, and allow them to be real-valued. The result turns into an eigenvector problem known as spectral clustering (see e.g., (Shi and Malik 2000)). In general, the technique of performing eigenanalysis of graphs is called spectral graph theory (Chung 1997).

Going into the details would take us too far afield, but below we give a very brief summary, based on (von Luxburg 2007), since we will encounter some of these ideas later on.

25.4.1 Graph Laplacian

Let W be a symmetric weight matrix for a graph, where $w_{ij} = w_{ji} \geq 0$. Let $D = \text{diag}(d_i)$ be a diagonal matrix containing the weighted degree of each node. We define the graph Laplacian as follows:

$$L = D - W \quad (25.43)$$

This matrix has various important properties. Because each row sums to zero, we have that $\mathbf{1}$ is an eigenvector with eigenvalue 0. Furthermore, the matrix is symmetric and positive semi-definite. To see this, note that

$$\mathbf{f}^T L \mathbf{f} = \mathbf{f}^T D \mathbf{f} - \mathbf{f}^T W \mathbf{f} = \sum_i d_i f_i^2 - \sum_{i,j} w_{ij} f_i f_j \quad (25.44)$$

$$= \frac{1}{2} \sum_i (d_i f_i^2 - 2 \sum_j w_{ij} f_i f_j) = \frac{1}{2} \sum_i \sum_j (w_{ij} f_i^2 - w_{ij} f_i f_j) = \frac{1}{2} \sum_i \sum_j w_{ij} (f_i - f_j)^2 \quad (25.45)$$

Hence $\mathbf{f}^T L \mathbf{f} \geq 0$ for all $\mathbf{f} \in \mathbb{R}^N$. Consequently we see that L has N non-negative, real-valued eigenvalues, $0 \leq \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_N$.

To get some intuition as to why L might be useful for graph-based clustering, we note the following result.

Theorem 25.4.1. The set of eigenvectors of L with eigenvalue 0 is spanned by the indicator vectors $\mathbf{1}, \mathbf{A}_1, \dots,$

$\mathbf{1A}\mathbf{K}$, where $\mathbf{A}\mathbf{k}$ are the K connected components of the graph.

Proof. Let us start with the case $K = 1$. If \mathbf{f} is an eigenvector with eigenvalue 0, then $0 = \sum_{j \neq i} w_{ij}(f_i - f_j)$. If two nodes are connected, so $w_{ij} > 0$, we must have that $f_i = f_j$. Hence \mathbf{f} is constant for all vertices which are connected by a path in the graph. Now suppose $K > 1$. In this case, \mathbf{L} will be block diagonal. A similar argument to the above shows that we will have K indicator functions, which "select out" the connected components.

This suggests the following algorithm. Compute the first K eigenvectors \mathbf{u}_k of \mathbf{L} . Let $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_K]$ be an $N \times K$ matrix with the eigenvectors in its columns. Let $\mathbf{y}_i \in \mathbb{R}^K$ be the i 'th row of \mathbf{U} . Since these \mathbf{y}_i will be piecewise constant, we can apply K-means clustering to them to recover the connected components. Now assign point i to cluster k iff row i of \mathbf{Y} was assigned to cluster k .

In reality, we do not expect a graph derived from a real similarity matrix to have isolated connected components — that would be too easy. But it is reasonable to suppose the graph is a small "perturbation" from such an ideal. In this case, one can use results from perturbation theory to show that the eigenvectors of the perturbed Laplacian will be close to these ideal indicator functions (Ng et al. 2001). Note that this approach is related to kernel PCA (Section 14.4.4). In particular, KPCA uses the largest eigenvectors of \mathbf{W} ; these are equivalent to the smallest eigenvectors of $\mathbf{I} - \mathbf{W}$. This is similar to the above method, which computes the smallest eigenvectors of $\mathbf{L} = \mathbf{D} - \mathbf{W}$. See (Bengio et al. 2004) for details. In practice, spectral clustering gives much better results than KPCA.

25.4.2 Normalized graph Laplacian

In practice, it is important to normalize the graph Laplacian, to account for the fact that some nodes are more highly connected than others. There are two common ways to do this. One method, used in e.g., (Shi and Malik 2000; Meila 2001), creates a stochastic matrix where each row sums to one:

$$\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W} \quad (25.46)$$

The eigenvalues and eigenvectors of \mathbf{L} and \mathbf{L}_{rw} are closely related to each other (see (von Luxburg 2007) for details). Furthermore, one can show that for \mathbf{L}_{rw} , the eigenspace of 0 is again spanned by the indicator vectors $\mathbf{1A}\mathbf{k}$. This

suggests the following algorithm: find the smallest K eigenvectors of L_{rw} , create U, cluster the rows of U using K-means, then infer the partitioning of the original points (Shi and Malik 2000). (Note that the eigenvectors/ values of L_{rw} are equivalent to the generalized eigenvectors/ values of L, which solve $Lu = \lambda DU$.)

Another method, used in e.g., (Ng et al. 2001), creates a symmetric matrix

$$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2} \quad (25.47)$$

This time the eigenspace of 0 is spanned by $D^{1/2} 1_A k$.

This suggests the following algorithm: find the smallest K eigenvectors of L_{sym} , create U, normalize each row to unit norm by creating $t_{ij} = u_{ij} / \|u\|_2$ ($k \leq ik$), cluster the rows of T using K-means, then infer the partitioning of the original points (Ng et al. 2001).

There is an interesting connection between Ncuts and random walks on a graph (Meila 2001). First note that $P = D^{-1} W = I - L_{rw}$ is a stochastic matrix, where $p_{ij} = w_{ij}/d_i$ can be interpreted as the probability of going from i to j. If the graph is connected and non-bipartite, it possesses a unique stationary distribution $\pi = (\pi_1, \dots, \pi_N)$, where $\pi_i = d_i/\text{vol}(V)$. Furthermore, one can show that

$$\text{Ncut}(A, A) = p(A|A) + p(A|A) \quad (25.48)$$

This means that we are looking for a cut such that a random walk rarely makes transitions from A to A or vice versa.

25.4.3 Example

Figure 25.11 illustrates the method in action. In Figure 25.11(a), we see that K-means does a poor job of clustering, since it implicitly assumes each cluster corresponds to a spherical Gaussian. Next we try spectral clustering. We define a similarity matrix using the Gaussian kernel. We compute the first two eigenvectors of the Laplacian. From this we can infer the clustering in Figure 25.11(b).

Since the method is based on finding the smallest K eigenvectors of a sparse matrix, it takes $O(N^3)$ time.

However, a variety of methods can be used to scale it up for large datasets (see e.g., (Yan et al. 2009)).

已思考 32s

下面先给出**完整中文翻译**（逐段对应原文，保留公式与编号；对个别明显的排版小错误做了不改含义的微调并在行文中自然更正），随后是**通俗解释**，最后是**要点总结**。

译文

25.4 谱聚类 (Spectral clustering)

一种关于聚类的替代视角是图割 (graph cuts)。这个想法是：我们用相似度矩阵 S 构造一个带权无向图 W ，通常通过为每个点只保留其若干近邻来建图；这能让图变稀疏，从而加速计算。如果我们想把图划分成 K 个簇，记为 A_1, \dots, A_K ，一个自然的准则是最小化

$$\text{cut}(A_1, \dots, A_K) = \frac{1}{2} \sum_{k=1}^K W(A_k, \bar{A}_k) \quad (25.41)$$

其中 $\bar{A}_k = V \setminus A_k$ 表示 A_k 的补集，且 $W(A, B) = \sum_{i \in A, j \in B} w_{ij}$ 。当 $K = 2$ 时，这个问题容易求解。不幸的是，最优解常常只是把单个数据点从其余点中切出去。为了保证各集合合理够大，我们可以定义**归一化割 (normalized cut) **为

$$\text{Ncut}(A_1, \dots, A_K) = \frac{1}{2} \sum_{k=1}^K \frac{\text{cut}(A_k, \bar{A}_k)}{\text{vol}(A_k)} \quad (25.42)$$

其中 $\text{vol}(A) = \sum_{i \in A} d_i$ ，而 $d_i = \sum_{j=1}^N w_{ij}$ 是结点 i 的带权度数。这会把图分成 K 个簇，使得簇内结点彼此相似、而与其他簇中的结点不同。

我们可以把 Ncut 问题表述为：在二值向量 $c_i \in \{0, 1\}^N$ （若样本 i 属于簇 k ，则 $c_{ik} = 1$ ）上做搜索以最小化目标函数。不幸的是，这是个 NP-hard 问题 (Wagner and Wagner 1993)。亲和传播是一种可用的解法。另一种办法是放松“ c_i 必须是二值”的约束，允许它们取实数。这样问题会转化为一个特征向量 (特征值) 问题，称为谱聚类 (见如 Shi and Malik 2000)。更一般地，对图做特征分解的技术称为谱图理论 (Chung 1997)。

深入细节会让我们跑题太远，但我们下面基于 (von Luxburg 2007) 给一个非常简短的总结，因为后文我们还会遇到这些思想。

25.4.1 图拉普拉斯算子 (Graph Laplacian)

令 W 为一个对称的权重矩阵，满足 $w_{ij} = w_{ji} \geq 0$ 。令 $D = \text{diag}(d_i)$ 是包含每个结点带权度数的对角矩阵。我们定义图拉普拉斯矩阵

$$L = D - W \quad (25.43)$$

这个矩阵有一些重要性质。因为每一行求和为 0，所以常数向量 **1** 是 L 的一个特征向量，其特征值为 0。此外， L 是对称且半正定的。为说明这一点，注意到

$$\mathbf{f}^\top L \mathbf{f} = \mathbf{f}^\top D \mathbf{f} - \mathbf{f}^\top W \mathbf{f} = \sum_i d_i f_i^2 - \sum_{i,j} f_i f_j w_{ij} \quad (25.44)$$

$$= \frac{1}{2} (\sum_i d_i f_i^2 - 2 \sum_{i,j} f_i f_j w_{ij} + \sum_j d_j f_j^2) = \frac{1}{2} \sum_{i,j} w_{ij} (f_i - f_j)^2 \quad (25.45)$$

因此对一切 $\mathbf{f} \in \mathbb{R}^N$, $\mathbf{f}^\top L \mathbf{f} \geq 0$ 。由此可见, L 有 N 个**非负的实**特征值, 记作 $0 \leq \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_N$ 。

为了直观理解为什么 L 对基于图的聚类有用, 注意如下结果。

定理 25.4.1. 具有特征值 0 的所有特征向量的集合, 由指标向量 $\mathbf{1}_{A_1}, \dots, \mathbf{1}_{A_K}$ 张成, 其中 A_k 是图的 K 个**连通分量**。

证明. 先看 $K = 1$ 的情形。若 \mathbf{f} 是特征值为 0 的特征向量, 则 $0 = \sum_{i,j} w_{ij} (f_i - f_j)^2$ 。若两个结点相连 ($w_{ij} > 0$), 则必有 $f_i = f_j$ 。因此在图的每个连通分量上, \mathbf{f} 是常数。若 $K > 1$, 此时 L 呈**块对角**结构。类似推理可知会得到 K 个**指标函数**, 它们“挑选出”各个连通分量。

这启发了如下算法: 计算 L 的**最小的** K 个特征向量 u_k 。令 $U = [u_1, \dots, u_K]$ 为把这些特征向量按列拼成的 $N \times K$ 矩阵。令 $y_i \in \mathbb{R}^K$ 是 U 的第 i 行。由于这些 y_i 在理想情况下是**分段常数的**, 我们可以对它们做 K-means 来找回连通分量。最终把点 i 分到簇 k , 当且仅当 U 的第 i 行被 K-means 分到簇 k 。

现实里, 由真实相似度矩阵导出的图**通常不会**真的有互相孤立的连通分量——那就太容易了。但合理的假设是: 该图是某个“理想图”的一个**小扰动**。在这种情况下, 可用**扰动理论**结果证明: 扰动后拉普拉斯的特征向量会**接近**这些理想的指标函数 (Ng et al. 2001)。

注意该方法与**核 PCA** (第 14.4.4 节) 有关。特别地, KPCA 使用的是 W 的**最大**特征向量; 它们等价于 $I - W$ 的**最小**特征向量。这与上面的做法类似, 上面是计算 $L = D - W$ 的**最小**特征向量。细节见 (Bengio et al. 2004)。在实践中, **谱聚类**通常比 KPCA 给出更好的聚类结果。

25.4.2 归一化图拉普拉斯 (Normalized graph Laplacian)

实际中, 考虑到有些结点比另一些**更高连接度**, 对图拉普拉斯做**归一化**很重要。常见有两种方式。

一种方式 (如 Shi and Malik 2000; Meilă 2001) 构造一个**行随机矩阵** (每行求和为 1) :

$$L_{rw} = D^{-1}L = I - D^{-1}W \quad (25.46)$$

L 与 L_{rw} 的特征值/向量**密切相关** (详见 von Luxburg 2007)。此外可证明: 对 L_{rw} , 特征值 0 的特征空间仍由指标向量 $\mathbf{1}_{A_k}$ 张成。这提示如下算法: 求 L_{rw} 的最小 K 个特征向

量，构成 U ，对其行向量做 K-means，再把结果映射回原始点的划分（Shi and Malik 2000）。（注意： L_{rw} 的特征对等价于 L 的广义特征对，即解 $Lu = \lambda Du$ 。）

另一种方式（如 Ng et al. 2001）构造一个对称矩阵

$$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2} \quad (25.47)$$

此时，特征值 0 的特征空间由 $D^{1/2} \mathbf{1}_{A_k}$ 张成。这提示如下算法：求 L_{sym} 的最小 K 个特征向量，构成 U ；把每一行单位化，令 $t_{ij} = u_{ij} / \sqrt{\sum_k u_{ik}^2}$ 得到矩阵 T ；对 T 的行做 K-means，再把结果映射回原始点的划分（Ng et al. 2001）。

关于 Ncut 与图上的随机游走之间存在一个有趣联系（Meilă 2001）。首先注意 $P = D^{-1} W = I - L_{rw}$ 是一个随机矩阵，其中 $p_{ij} = w_{ij}/d_i$ 可解释为从 i 走到 j 的概率。若图连通且非二部，它具有唯一的平稳分布 $\pi = (\pi_1, \dots, \pi_N)$ ，其中 $\pi_i = d_i/\text{vol}(V)$ 。此外可证明

$$\text{Ncut}(A, \bar{A}) = p(\bar{A} | A) + p(A | \bar{A}) \quad (25.48)$$

这意味着：我们在寻找一种割，使得随机游走很少从 A 跳到 \bar{A} 或反之。

25.4.3 示例 (Example)

图 25.11 展示了该方法的运行。图 25.11(a) 中，K-means 聚类效果很差，因为它隐含假设每个簇都对应一个球形高斯。接着我们尝试谱聚类：用高斯核定义相似度矩阵，计算拉普拉斯的前两个（最小的）特征向量，由此得到图 25.11(b) 的聚类。

由于该方法的核心是求一个稀疏矩阵的最小 K 个特征向量，复杂度为 $O(N^3)$ 。不过已有多种方法可将其扩展到大数据集（见如 Yan et al. 2009）。

通俗解释（尽量不靠公式，把直觉说清）

它在干嘛？

把所有样本当成图里的结点，相似的两点之间放一条粗（权重大）的边，不相似就放细或不放边；然后把图切成几块，让块内边“又粗又多”，块间边“又细又少”。这就是图割。直接最小化“割边总量”会出现“把孤零零的一点切出去”的偏差，所以用归一化割（Ncut），在“割边”之外再惩罚小块，逼着每块都“像样”。

为什么要拉普拉斯和特征向量？

如果图真的分成了几块互不相连的子图，那么 $L = D - W$ 的最小几个特征向量正好就是“哪块是哪块”的指示灯（每块对应一个 0 特征值的特征向量）。真实数据不会那么完美，但如果只是“稍微扰动”，这些特征向量仍然近似那几个指示灯。于是我们：

1. 求出最小的 K 个特征向量，把它们按列拼成 U ；
2. 看 U 的每一行（对应一个样本）在这 K 个特征向量里的“坐标”；
3. 对这些坐标做 K-means，就把样本分到了 K 块。
(为公平起见，还常对 L 做归一化，以免“度数很大”的点主导结果；这就有 L_{rw} 与 L_{sym} 两种常见做法。)

和随机游走的关系？

把 W 变成转移概率 $P = D^{-1} W$ ，就是在图上“随机走”。好的切分应该让“游走”很少从一个块走到另一个块。事实证明，Ncut 就是在度量“跨块跳转的概率有多大”。

为什么比直接用 K-means 强？

K-means 假设簇是圆的（或各向同性高斯）。但很多数据簇是月牙、环形等奇形怪状。谱聚类先把数据放进“图”的空间里，再用特征向量把“形状”摊平，能把非凸形状分得更好。

两种归一化怎么选？

- L_{rw} ：行随机视角，求最小 K 个特征向量后直接对行做 K-means。
 - L_{sym} ：对称视角，求最小 K 个特征向量后先把每行单位化再做 K-means（这能避免尺度差异的干扰）。
- 两者在实践里都常用。

复杂度&扩展

直接算特征分解约 $O(N^3)$ ；但实际会用稀疏图（k 近邻）、稀疏特征求解器、Nystrom 近似等把成本大大降下来。

要点总结

- **谱聚类=图割视角的聚类**：用相似度建稀疏带权图，寻找归一化割小的划分。
- **图拉普拉斯 $L = D - W$ 的最小 K 个特征向量** 在理想情况下等于连通分量的指示函数；现实中它们的近似仍可把数据分成 K 簇。
- **算法骨架**：建图 → （可选）归一化 L_{rw} 或 L_{sym} → 取最小 K 个特征向量 → 对行做 K-means → 得到分簇。
- **Ncut 与随机游走**：好切分让“跨块”跳转概率小； $\text{Ncut}(A, \bar{A}) = p(\bar{A} | A) + p(A | \bar{A})$ 。
- **优于 K-means 的典型场景**：簇是非球形/非凸的流形状结构。
- **工程要点**：用近邻图保持稀疏、选合适的核（如高斯核）、选择 K 可配合**谱间隙（eigengap）**等启发；大规模用稀疏解法/近似。

一句话：谱聚类把“相似度”变成“图”，再用**拉普拉斯的特征向量**把“形状信息”提炼出来，最后用 K-means 在这个新空间里做一次**干净利落**的分簇。

这一节主要在讲什么？我不太懂

已思考 6s

没关系～我用大白话把 25.4 「谱聚类」的主线捋清楚，读完就知道它在干嘛。

一句话版

谱聚类把聚类问题变成**在相似度图上找“好切法”**：先把样本当成图的节点、用相似度当边，再用**拉普拉斯矩阵的前 K 个特征向量**把数据映射到一个新空间，最后在这个新空间里用 K-means 分组。它特别擅长处理**非球形/弯曲**的簇。

直观图景

1. **建图**：样本=节点，相似就连粗边（权重大），不相似就细边或不连；常用 **k 近邻 + 高斯核**，图更稀疏、计算更快。
2. **什么是“好切法”**：直接把“边权和”切小容易把孤零零的点切出去 → **用归一化割 (Ncut)**：既要少切跨簇的边，还要**惩罚太小的块**，逼出像样的簇。
3. **为何看特征向量**：如果图真的分成互不相连的 K 块，拉普拉斯 $L = D - W$ 的**最小 K 个特征向量**就像 K 个“指示灯”，分别亮在各块上。真实数据只是“近似如此”，这些特征向量仍**近似指示**每个簇。
4. **实际做法（骨架）**：
 - 选归一化方式： $L_{rw} = I - D^{-1}W$ 或 $L_{sym} = I - D^{-1/2}WD^{-1/2}$ ；
 - 取**最小的 K 个特征向量**组成矩阵 U （用 L_{sym} 时把每行单位化）；
 - 对 U 的行做 K-means，得到最终簇。
5. **随机游走理解**：把边权正则化成转移概率 $P = D^{-1}W$ 。好的切分让随机游走**很少**从一个簇走到另一个簇（这正是 Ncut 在度量的东西）。

什么时候比 K-means 更香

- 簇是**长条、月牙、环形、流形等非凸形状**；
- 特征尺度差异大、节点度数差别明显（用归一化能中和影响）；
- 只要有相似度就能开干，不一定原始特征模型。

小心事项

- 朴素特征分解是 $O(N^3)$, 实际要靠稀疏图、稀疏特征求解器或 Nyström/近似方法 扩展到大规模;
- K 可用**谱间隙 (eigengap) **等启发式挑选;
- 相似度/近邻参数 (如 σ 、 k) 会显著影响效果。

口袋清单 (记住这 5 步)

- 由相似度建稀疏图 W (kNN + 高斯核)。
- 选 L_{rw} 或 L_{sym} 并构造 L 。
- 求 L 的最小 K 个特征向量 → 矩阵 U (L_{sym} 时行单位化)。
- 对 U 的行做 K-means。
- 把簇标签还原到原始样本。

一句话收尾：谱聚类=“图割思想 + 拉普拉斯特特征向量 + K-means”，专治 K-means 搞不定的弯弯绕绕的簇。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

25.5 Hierarchical clustering

Mixture models, whether finite or infinite, produce a “flat” clustering. Often we want to learn a hierarchical clustering, where clusters can be nested inside each other. There are two main approaches to hierarchical clustering: bottom-up or agglomerative clustering, and top-down or divisive clustering. Both methods take as input a dissimilarity matrix between the objects. In the bottom-up approach, the most similar groups are merged at each step. In the top-down approach, groups are split using various different criteria. We give the details below.

Note that agglomerative and divisive clustering are both just heuristics, which do not optimize any well-defined objective function. Thus it is hard to assess the quality of the clustering they produce in any formal sense.

Furthermore, they will always produce a clustering of the input data, even if the data has no structure at all (e.g., it is random noise). Later in this section we will discuss a

probabilistic version of hierarchical clustering that solves both these problems.

25.5.1 Agglomerative clustering

Agglomerative clustering starts with N groups, each initially containing one object, and then at each step it merges the two most similar groups until there is a single group, containing all the data. See Algorithm 11 for the pseudocode. Since picking the two most similar clusters to merge takes $O(N^2)$ time, and there are $O(N)$ steps in the algorithm, the total running time is $O(N^3)$. However, by using a priority queue, this can be reduced to $O(N^2 \log N)$ (see e.g., (Manning et al. 2008, ch. 17) for details). For large N , a common heuristic is to first run K-means, which takes $O(KND)$ time, and then apply hierarchical clustering to the estimated cluster centers.

The merging process can be represented by a binary tree, called a dendrogram, as shown in Figure 25.12(b). The initial groups (objects) are at the leaves (at the bottom of the figure), and every time two groups are merged, we join them in the tree. The height of the branches represents the dissimilarity between the groups that are being joined. The root of the tree (which is at the top) represents a group containing all the data. If we cut the tree at any given height, we induce a clustering of a given size. For example, if we cut the tree in Figure 25.12(b) at height 2, we get the clustering $\{\{4, 5\}, \{1, 3\}, \{2\}\}$. We discuss the issue of how to choose the height/ number of clusters below.

A more complex example is shown in Figure 25.13(a), where we show some gene expression data. If we cut the tree in Figure 25.13(a) at a certain height, we get the 16 clusters shown in Figure 25.13(b).

There are actually three variants of agglomerative clustering, depending on how we define the dissimilarity between groups of objects. These can give quite different results, as shown in Figure 25.15. We give the details below.

25.5.1.1 Single link

In single link clustering, also called nearest neighbor clustering, the distance between two groups G and H is defined as the distance between the two closest members of each group:

$$d_{SL}(G, H) = \min_{i \in G, j \in H} d_{i,j} \quad (25.49)$$

See Figure 25.14(a).

The tree built using single link clustering is a minimum spanning tree of the data, which is a tree that connects all the objects in a way that minimizes the sum of the edge weights (distances). To see this, note that when we merge two clusters, we connect together the two closest members of the clusters; this adds an edge between the corresponding nodes, and this is guaranteed to be the "lightest weight" edge joining these two clusters. And once two clusters have been merged, they will never be considered again, so we cannot create cycles. As a consequence of this, we can actually implement single link clustering in $O(N^2)$ time, whereas the other variants take $O(N^3)$ time.

25.5.1.2 Complete link

In complete link clustering, also called furthest neighbor clustering, the distance between two groups is defined as the distance between the two most distant pairs:

$$d_{CL}(G,H) = \max_{i \in G, j \in H} d_{i,j} \quad (25.50)$$

See Figure 25.14(b).

Single linkage only requires that a single pair of objects be close for the two groups to be considered close together, regardless of the similarity of the other members of the group. Thus clusters can be formed that violate the compactness property, which says that all the observations within a group should be similar to each other. In particular if we define the diameter of a group as the largest dissimilarity of its members, $d_G = \max_{i \in G, j \in G} d_{i,j}$, then we can see that single linkage can produce clusters with large diameters. Complete linkage represents the opposite extreme: two groups are considered close only if all of the observations in their union are relatively similar. This will tend to produce clusterings with small diameter, i.e., compact clusters.

25.5.1.3 Average link

In practice, the preferred method is average link clustering, which measures the average distance between all pairs:

$$d_{avg}(G,H) = \frac{1}{n_G n_H} \sum_{i \in G, j \in H} d_{i,j} \quad (25.51)$$

where n_G and n_H are the number of elements in groups G and H. See Figure 25.14(c).

Average link clustering represents a compromise between single and complete link clustering. It tends to produce

relatively compact clusters that are relatively far apart. However, since it involves averaging of the $d_{i,i}$'s, any change to the measurement scale can change the result. In contrast, single linkage and complete linkage are invariant to monotonic transformations of $d_{i,i}$, since they leave the relative ordering the same.

25.5.2 Divisive clustering

Divisive clustering starts with all the data in a single cluster, and then recursively divides each cluster into two daughter clusters, in a top-down fashion. Since there are $2N - 1 - 1$ ways to split a group of N items into 2 groups, it is hard to compute the optimal split, so various heuristics are used. One approach is pick the cluster with the largest diameter, and split it in two using the K-means or K-medoids algorithm with $K = 2$. This is called the bisecting K-means algorithm (Steinbach et al. 2000). We can repeat this until we have any desired number of clusters. This can be used as an alternative to regular K-means, but it also induces a hierarchical clustering.

Another method is to build a minimum spanning tree from the dissimilarity graph, and then to make new clusters by breaking the link corresponding to the largest dissimilarity. (This actually gives the same results as single link agglomerative clustering.)

Yet another method, called dissimilarity analysis (Macnaughton-Smith et al. 1964), is as follows. We start with a single cluster containing all the data, $G = \{1, \dots, N\}$. We then measure the average dissimilarity of $i \in G$ to all the other $i \in G$:

$$dG_i = \frac{1}{n} \sum_{j \neq i} d_{i,j} \quad (25.52)$$

We remove the most dissimilar object and put it in its own cluster H :

$$i^* = \operatorname{argmax}_{i \in G} dG_i, G = G \setminus \{i^*\}, H = \{i^*\} \quad (25.53)$$

We now continue to move objects from G to H until some stopping criterion is met. Specifically, we pick a point i^* to move that maximizes the average dissimilarity to each $i \in G$ but minimizes the average dissimilarity to each $i \in H$:

$$dH_i = \frac{1}{n} \sum_{j \in H} d_{i,j}, i^* = \operatorname{argmax}_{i \in G} dG_i - dH_i \quad (25.54)$$

We continue to do this until $dG_i - dH_i$ is negative. The final result is that we have split G into two daughter clusters, G and H . We can then recursively call the algorithm on G and/or H , or on any other node in the tree.

For example, we might choose to split the node G whose average dissimilarity is highest, or whose maximum dissimilarity (i.e., diameter) is highest. We continue the process until the average dissimilarity within each cluster is below some threshold, and/or all clusters are singletons. Divisive clustering is less popular than agglomerative clustering, but it has two advantages. First, it can be faster, since if we only split for a constant number of levels, it takes just $O(N)$ time. Second, the splitting decisions are made in the context of seeing all the data, whereas bottom-up methods make myopic merge decisions.

25.5.3 Choosing the number of clusters

It is difficult to choose the “right” number of clusters, since a hierarchical clustering algorithm will always create a hierarchy, even if the data is completely random. But, as with choosing K for K-means, there is the hope that there will be a visible “gap” in the lengths of the links in the dendrogram (represent the dissimilarity between merged groups) between natural clusters and unnatural clusters. Of course, on real data, this gap might be hard to detect. In Section 25.5.4, we will present a Bayesian approach to hierarchical clustering that nicely solves this problem.

25.5.4 Bayesian hierarchical clustering

There are several ways to make probabilistic models which produce results similar to hierarchical clustering, e.g., (Williams 2000; Neal 2003b; Castro et al. 2004; Lau and Green 2006). Here we present one particular approach called Bayesian hierarchical clustering (Heller and Ghahramani 2005). Algorithmically it is very similar to standard bottom-up agglomerative clustering, and takes comparable time, whereas several of the other techniques referenced above are much slower. However, it uses Bayesian hypothesis tests to decide which clusters to merge (if any), rather than computing the similarity between groups of points in some ad-hoc way. These hypothesis tests are closely related to the calculations required to do inference in a Dirichlet process mixture model, as we will see. Furthermore, the input to the model is a data matrix, not a dissimilarity matrix.

25.5.4.1 The algorithm

Let $D = \{x_1, \dots, x_N\}$ represent all the data, and let D_i be the set of datapoints at the leaves of the subtree T_i . At each step, we compare two trees T_i and T_j to see if they should

be merged into a new tree. Define D_{ij} as their merged data, and let $M_{ij} = 1$ if they should be merged, and $M_{ij} = 0$ otherwise.

The probability of a merge is given by

$$r_{ij} = p(D_{ij}|M_{ij} = 1)p(M_{ij} = 1) p(D_{ij}|T_{ij}) \quad (25.55)$$

$$p(D_{ij}|T_{ij}) = p(D_{ij}|M_{ij} = 1)p(M_{ij} = 1) + p(D_{ij}|M_{ij} = 0)p(M_{ij} = 0) \quad (25.56)$$

Here $p(M_{ij} = 1)$ is the prior probability of a merge, which can be computed using a bottom-up algorithm described below. We now turn to the likelihood terms. If $M_{ij} = 1$, the data in D_{ij} is assumed to come from the same model, and hence

$$p(D_{ij}|M_{ij} = 1) = \prod_{x_n \in D_{ij}} p(x_n|\theta) \int p(\theta|\lambda) d\theta \quad (25.57)$$

If $M_{ij} = 0$, the data in D_{ij} is assumed to have been generated by each tree independently, so

$$p(D_{ij}|M_{ij} = 0) = p(D_i|T_i)p(D_j|T_j) \quad (25.58)$$

These two terms will have already been computed by the bottom-up process. Consequently we have all the quantities we need to decide which trees to merge. See Algorithm 9 for the pseudocode, assuming $p(M_{ij})$ is uniform. When finished, we can cut the tree at points where $r_{ij} < 0.5$.

25.5.4.2 The connection with Dirichlet process mixture models

In this section, we will establish the connection between BHC and DPMMs. This will in turn give us an algorithm to compute the prior probabilities $p(M_{ij} = 1)$.

Note that the marginal likelihood of a DPMM, summing over all $2N - 1$ partitions, is given by

$$p(D_k) = \sum_{v \in V} p(v)p(D_v) \quad (25.59)$$

$$p(v) = \alpha^{m_v} m_v \prod_{l=1}^L \frac{\Gamma(n_{vl})}{\Gamma(n_{vl} + \alpha)} \frac{\Gamma(\alpha)}{\Gamma(n_k + \alpha)} \quad (25.60)$$

$$p(D_v) = \prod_{l=1}^L p(D_{vl}) \quad (25.61)$$

where V is the set of all possible partitions of D_k , $p(v)$ is the probability of partition v , m_v is the number of clusters in partition v , n_{vl} is the number of points in cluster l of partition v , D_{vl} are the points in cluster l of partition v , and n_k are the number of points in D_k .

One can show (Heller and Ghahramani 2005) that $p(D_k|T_k)$ computed by the BHC algorithm is similar to $p(D_k)$ given above, except for the fact that it only sums over partitions which are consistent with tree T_k . (The number of tree-consistent partitions is exponential in the number of data points for balanced binary trees, but this is obviously a

subset of all possible partitions.) In this way, we can use the BHC algorithm to compute a lower bound on the marginal likelihood of the data from a DPMM.

Furthermore, we can interpret the algorithm as greedily searching through the exponentially large space of tree-consistent partitions to find the best ones of a given size at each step.

We are now in a position to compute $\pi_k = p(M_k = 1)$, for each node k with children i and j . This is equal to the probability of cluster D_k coming from the DPMM, relative to all other partitions of D_k consistent with the current tree. This can be computed as follows: initialize $d_i = \alpha$ and $\pi_i = 1$ for each leaf i ; then as we build the tree, for each internal node k , compute $d_k = \alpha\Gamma(n_k) + d_{ij}$, and $\pi_k = \alpha\Gamma(n_k) / d_k$, where i and j are k 's left and right children.

25.5.4.3 Learning the hyper-parameters

The model has two free-parameters: α and λ , where λ are the hyper-parameters for the prior on the parameters θ . In (Heller and Ghahramani 2005), they show how one can back-propagate gradients of the form $\partial p(D_k|T_k) / \partial \lambda$ through the tree, and thus perform an empirical Bayes estimate of the hyper-parameters.

25.5.4.4 Experimental results

(Heller and Ghahramani 2005) compared BHC with traditional agglomerative clustering algorithms on various data sets in terms of purity scores. The results are shown in Table 25.1. We see that BHC did much better than the other methods on all datasets except the forensic glass one.

Figure 25.16 visualizes the tree structure estimated by BHC and agglomerative hierarchical clustering (AHC) on the newsgroup data (using a beta-Bernoulli model). The BHC tree is clearly superior (look at the colors at the leaves, which represent class labels). Figure 25.17 is a zoom-in on the top few nodes of these two trees. BHC splits off clusters concerning sports from clusters concerning cars and space. AHC keeps sports and cars merged together. Although sports and cars both fall under the same “rec” newsgroup heading (as opposed to space, that comes under the “sci” newsgroup heading), the BHC clustering still seems more reasonable, and this is borne out by the quantitative purity scores.

BHC has also been applied to gene expression data, with good results (Savage et al. 2009).

已思考 30s

下面先给出**完整中文翻译**（逐段对应原文，保留公式与编号；对个别明显的排版/索引符号小误写做了不改含义的微调，例如用 i' 表示“另一个索引”），随后是**通俗解释**，最后是**要点总结**。

译文

25.5 层次聚类 (Hierarchical clustering)

无论是有限还是无限的混合模型，产生的都是“扁平”的聚类。很多时候我们想学习**层次聚类**，其中簇可以彼此嵌套。

层次聚类有两种主要方法：**自底向上**（凝聚式，agglomerative clustering）与**自顶向下**（分裂式，divisive clustering）。两种方法都把对象之间的**不相似度矩阵**作为输入。在自底向下（应为“自底向上”）的方法里，每一步把**最相似的组合并**。在自顶向下的方法里，根据各种不同的准则把组**拆分**。我们在下文给出细节。

需要注意的是，凝聚式与分裂式聚类都只是**启发式**，它们并不优化任何**定义良好的目标函数**。因此，从严格意义上很难评估它们产生的聚类质量。此外，它们**总会**对输入数据产生一个聚类，即使数据**毫无结构**（例如就是随机噪声）。在本节后面，我们会讨论一种**概率版**的层次聚类，它能解决这两个问题。

25.5.1 凝聚式聚类 (Agglomerative clustering)

凝聚式聚类从 N 个组开始，每个组一开始只包含一个对象；然后每一步都把**最相似的两组合并**，直到只剩一个包含所有数据的组。伪代码见算法 11。由于挑选“要合并的两簇”需要 $O(N^2)$ 时间，而算法有 $O(N)$ 步，所以**总时间是 $O(N^3)$** 。不过使用**优先队列**可以把它降到 $O(N^2 \log N)$ （细节见如 Manning 等，2008，第 17 章）。当 N 很大时，一个常见启发是先跑 K-means（花费 $O(KND)$ ），然后只对**估计的簇中心**做层次聚类。

合并过程可以用一棵**二叉树**表示，称为**系统树/树状图** (dendrogram)，如图 25.12(b)。初始的各组（对象）位于**叶子**（图的底部），每当两组被合并，我们就在树上把它们连起来。树枝的高度表示被合并的两组之间的**不相似度**。树的**根**（顶部）代表包含所有数据的那一组。如果我们在任意高度把树**切开**，就得到对应规模的一个聚类。例如，把图 25.12(b) 的树在高度 2 处切开，我们得到聚类 $\{\{\{4, 5\}, \{1, 3\}\}, \{2\}\}$ 。关于如何选择切割高度/簇数，我们稍后讨论。

一个更复杂的例子在图 25.13(a) 中 (基因表达数据)。把树在某个高度切开, 就得到图 25.13(b) 所示的 16 个簇。

实际上, 凝聚式聚类有三种**变体**, 取决于我们如何定义“**组与组之间的不相似度**”。三种方法可能给出相当不同的结果, 如图 25.15 所示。细节如下。

25.5.1.1 单链接 (Single link)

在**单链接聚类** (又称**最近邻聚类**) 中, 组 G 与 H 的距离定义为两个组里**最近那一对成员**之间的距离:

$$d_{\text{SL}}(G, H) = \min_{i \in G, i' \in H} d_{i, i'} \quad (25.49)$$

见图 25.14(a)。

用单链接聚类建立的树是数据的一个**最小生成树**: 一棵把所有对象连起来且**边权 (距离) 和最小**的树。理由是: 当我们合并两簇时, 是把这两簇里**最近**的两个成员连起来; 这在对应结点之间加了一条边, 而且它保证是连接这两簇的“**最轻**”的边。而且一旦两簇被合并, 它们不会再被单独考虑, 因此不会形成环。由此, 单链接聚类实际上可以在 $O(N^2)$ 时间内实现, 而**其他变体要** $O(N^3)$ 。

25.5.1.2 全链接 (Complete link)

在**全链接聚类** (又称**最远邻聚类**) 中, 两个组的距离定义为两组中**最远那一对成员之间的距离**:

$$d_{\text{CL}}(G, H) = \max_{i \in G, i' \in H} d_{i, i'} \quad (25.50)$$

见图 25.14(b)。

单链接只要求**有一对**对象很近, 就把两个组视为近, 这与**组内紧凑性** (组内所有观测彼此应相似) 不一致。特别地, 如果我们把组的“**直径**”定义为其成员之间最大的**不相似度**,

$$d_G = \max_{i \in G, i' \in G} d_{i, i'},$$

就能看到单链接会产生**直径很大的簇**。全链接代表了另一个极端: 只有当**并集中所有观测**彼此都相对相似时, 两个组才被视为近。这倾向于产生**直径小**、也就是**紧凑的簇**。

25.5.1.3 平均链接 (Average link)

在实践中, 更常用的是**平均链接聚类**, 它度量的是**所有配对**的平均距离:

$$d_{\text{avg}}(G, H) = \frac{1}{n_G n_H} \sum_{i \in G} \sum_{i' \in H} d_{i, i'} \quad (25.51)$$

其中 n_G 与 n_H 是组 G 、 H 的元素个数。见图 25.14(c)。

平均链接是**单链接**与**全链接**之间的折中。它倾向于产生**相对紧凑且彼此相对较远的簇**。不过，由于它对 $d_{i,i'}$ 取**平均**，**测量尺度**的改变会**改变结果**。相对地，单链接与全链接对 $d_{i,i'}$ 的**单调变换不变**（因为它们只依赖次序）。

25.5.2 分裂式聚类 (Divisive clustering)

分裂式聚类从“所有数据在一个簇里”开始，然后**递归地**把每个簇**一分为二**，自顶向下地进行。由于把 N 个元素分成 2 组的方式有 $2^{N-1} - 1$ 种，想算最优切分很难，所以使用各种启发式。一种做法是选**直径最大的簇**，用 K-means 或 K-medoids 在 $K = 2$ 下把它一分为二。这叫**二分 K-means** (Steinbach 等, 2000)。重复此过程直到得到想要的簇数。它既可以作为常规 K-means 的替代，也会同时产生一个**层次**。

另一种方法是先从不相似度图构建**最小生成树**，然后把**最大不相似度**对应的那条边**断开来**产生新簇。（这实际上与**单链接凝聚式**得到相同结果。）

还有一种叫**不相似度分析** (Macnaughton-Smith 等, 1964) 的方法如下。我们从包含所有数据的单个簇开始， $G = \{1, \dots, N\}$ 。然后计算 $i \in G$ 对其余 $i' \in G$ 的**平均不相似度**：

$$d_G^i = \frac{1}{n_G} \sum_{i' \in G} d_{i,i'} \quad (25.52)$$

把**最不相似**的对象移走，单独成簇 H ：

$$i^{*} = \arg \max_{i \in G} d_G^i, \quad G \leftarrow G \setminus \{i^{*}\}, \quad H \leftarrow \{i^{*}\} \quad (25.53)$$

然后我们继续把对象从 G 移到 H ，直到满足某个停止准则。具体地，选择要移动的点 i^{*} 使得它对 G 的**平均不相似度最大**，但对 H 的**平均不相似度最小**：

$$d_H^i = \frac{1}{n_H} \sum_{i' \in H} d_{i,i'}, \quad i^{*} = \arg \max_{i \in G} (d_G^i - d_H^i) \quad (25.54)$$

当 $d_G^i - d_H^i$ 为**负**时停止。最终我们把 G 分成两个子簇 G 与 H 。然后可在 G 与/或 H 上递归调用算法，或在树的任何其他结点上调用。例如，可以选择切分**平均不相似度最高的**结点 G ，或**最大不相似度（直径）最高的**结点。持续这个过程，直到每个簇的**平均不相似度**低于某阈值，和/或所有簇都是**单元素**。

分裂式聚类**不如**凝聚式常用，但它有两个优点。第一，它可能**更快**：如果我们只切分固定层数，复杂度仅 $O(N)$ 。第二，它的切分决策是在**看到了全部数据**的背景下做出的，而自底向上的方法做的是**目光短浅**的合并决策。

25.5.3 选择簇的个数 (Choosing the number of clusters)

选择“合适”的簇数很难，因为**层次聚类算法**无论如何都会给出一棵层次树，即使数据完全随机。但就像给 K-means 选 K 一样，我们寄希望于在树状图里**连接长度**（代表被合并组之间的不相似度）的分布中，能看到“**明显的缺口**”，把**自然的簇与不自然的簇区分开**。当然，在真实数据上，这个缺口可能很难察觉。在第 25.5.4 节，我们将给出一种**贝叶斯的层次聚类**，它很好地解决了这个问题。

25.5.4 贝叶斯层次聚类 (Bayesian hierarchical clustering, BHC)

构造能产生与层次聚类相似结果的**概率模型**有多种方式（如 Williams 2000; Neal 2003b; Castro 等 2004; Lau 与 Green 2006）。这里我们介绍一种称为 **BHC** (Heller 与 Ghahramani, 2005) 的方法。从算法上，它与标准的自底向上凝聚式聚类非常相似，时间也可比；而前述的若干其他技术要慢很多。不过，BHC 是用**贝叶斯假设检验**来决定“是否合并某两簇”，而不是用随意定义的“组间相似度”。这些假设检验与在**狄利克雷过程混合模型 (DPMM) 中进行推断所需的计算密切相关**（如下会看到）。此外，BHC 的输入是**数据矩阵，而不是不相似度矩阵**。

25.5.4.1 算法 (The algorithm)

令 $D = \{x_1, \dots, x_N\}$ 表示所有数据，令 D_i 是子树 T_i 的**叶子**上的数据集合。每一步，我们拿两棵树 T_i 与 T_j 比较，判断是否应把它们合并成一棵新树。记它们合并后的数据为 D_{ij} ，并令

$M_{ij} = 1$ 表示“应合并”， $M_{ij} = 0$ 表示“**不合并**”。

合并的概率为

$$r_{ij} = \frac{p(D_{ij} \mid M_{ij} = 1)p(M_{ij} = 1)}{p(D_{ij} \mid T_{ij})} \quad (25.55)$$

$$p(D_{ij} \mid T_{ij}) = p(D_{ij} \mid M_{ij} = 1)p(M_{ij} = 1) + p(D_{ij} \mid M_{ij} = 0)p(M_{ij} = 0) \quad (25.56)$$

其中 $p(M_{ij} = 1)$ 是“**合并**”在先验下的概率，可用下面介绍的自底向上的算法计算。下面看似然项。若 $M_{ij} = 1$ ，则 D_{ij} 中的数据被认为来自**同一个模型**，因此

$$p(D_{ij} \mid M_{ij} = 1) = \int_{x_n \in D_{ij}} [\prod p(x_n \mid \theta)] p(\theta \mid \lambda) d\theta \quad (25.57)$$

若 $M_{ij} = 0$ ，则 D_{ij} 的数据被认为由两棵树**各自独立地**产生，因此

$$p(D_{ij} \mid M_{ij} = 0) = p(D_i \mid T_i)p(D_j \mid T_j) \quad (25.58)$$

这两项在自底向上的过程中已被计算出来。因此我们有了决定合并哪两棵树所需的全部量。若假设 $p(M_{ij})$ 为均匀分布，伪代码见算法 9。完成后，我们可以在所有 $r_{ij} < 0.5$ 的地方**剪断树**。

25.5.4.2 与狄利克雷过程混合模型的联系 (The connection with DPMMs)

本节我们建立 BHC 与 DPMM 的联系；这反过来会给出一个计算先验概率 $p(M_{ij} = 1)$ 的算法。

注意，DPMM 的**边际似然**（对所有可能的划分求和）为

$$p(D_k) = \sum_{v \in V} p(v) p(D_v) \quad (25.59)$$

$$p(v) = \frac{\alpha^{m_v} \prod_{l=1}^{m_v} \Gamma(n_v^l)}{\Gamma(n_k + \alpha) \Gamma(\alpha)} \quad (25.60)$$

$$p(D_v) = \prod_{l=1}^{m_v} p(D_v^l) \quad (25.61)$$

其中 V 是 D_k 的**所有可能划分**的集合； $p(v)$ 是划分 v 的先验概率； m_v 是划分 v 中的簇数； n_v^l 是划分 v 的第 l 个簇里的点数； D_v^l 是该簇的点集； n_k 是 D_k 的点数。

可以证明 (Heller 与 Ghahramani, 2005)，BHC 算法计算的 $p(D_k | T_k)$ 与上式的 $p(D_k)$ 相似，只是它只对**与树 T_k 一致**的划分求和。（对平衡二叉树而言，与树一致的划分数量是指数级，但显然它只是“所有可能划分”的一个子集。）这样，我们可以用 BHC 算法来计算 DPMM 数据边际似然的一个**下界**。此外，可以把该算法解释为：在指数大的“树一致划分”空间里做**贪心搜索**，每一步找到给定规模下最好的那些划分。

现在我们可以为每个具有子结点 i, j 的结点 k 计算

$\pi_k = p(M_k = 1)$ 。它等于“在所有与当前树一致的 D_k 的划分中， D_k 作为一个簇来自 DPMM 的概率”。具体可按下述递推计算：对每个**叶子** i 初始化 $d_i = \alpha$ 、 $\pi_i = 1$ ；随后在建树过程中，对每个**内部结点** k ，令

$$d_k = \alpha \Gamma(n_k) + d_i d_j, \quad \pi_k = \frac{\alpha \Gamma(n_k)}{d_k},$$

其中 i, j 是 k 的左右孩子。

25.5.4.3 学习超参数 (Learning the hyper-parameters)

该模型有两个自由参数： α 与 λ (λ 是参数 θ 的先验的超参数)。在 Heller 与 Ghahramani (2005) 中，作者展示了如何把 $\frac{\partial p(D_k | T_k)}{\partial \lambda}$ 这样的梯度沿树**反向传播**，从而进行**经验贝叶斯**估计这些超参数。

25.5.4.4 实验结果 (Experimental results)

Heller 与 Ghahramani (2005) 用**纯度分数**在多个数据集上比较了 BHC 与传统凝聚式聚类算法。结果见表 25.1。我们看到，除“forensic glass (法证玻璃)”数据外，BHC 在**所有**数据集上都明显更好。

图 25.16 可视化了在 newsgroup 数据上（使用 beta-Bernoulli 模型）BHC 与凝聚式层次聚类（AHC）估计出的树结构。BHC 的树显然更优（看叶子处的颜色——表示类别标签）。

图 25.17 放大了两棵树的顶部若干结点：BHC 把“体育”从“汽车与太空”中分离出来；而 AHC 把“体育”和“汽车”合在一起。尽管“体育”和“汽车”都属于同一“rec”新闻组（而“太空”属于“sci”新闻组），BHC 的聚类看起来依然更合理，这一点也被定量的纯度分数所支持。

BHC 也被用于基因表达数据，表现良好（Savage 等，2009）。

通俗解释（用直觉把路子说清）

1) 为什么要“层次”

扁平聚类只告诉你“分成几组”。但现实常是“组中有子组、子组中还有子子组”。**层次聚类**就是把这种“从粗到细”的结构做出来，一般画成一棵**树状图**（dendrogram）：越往上越粗，往下越细。

2) 两条路：自底向上 vs 自顶向下

- **自底向上（凝聚式）**：一开始每个样本单独成组；每次把最像的两组合并；一直到只剩一个大组。画出来就是把树枝往上连。
 - 三种“组距”定义：
 - **单链接**：只看最近那一对 → 容易产生“细长链条”、不紧凑；但能在 $O(N^2)$ 时间搞定（最小生成树）。
 - **全链接**：看最远那一对 → 更偏好紧凑簇（直径小）。
 - **平均链接**：看所有配对的平均 → 折中，常用；但对尺度变化敏感。
- **自顶向下（分裂式）**：先把所有样本放在一个组里；每次选一个组一分为二（比如选直径最大的组，用 K-means ($K=2$) 或 K-medoids 去切）；不断重复直到达到想要的簇数或组内差异够小。
 - 还可以：先建**最小生成树**，每次砍掉最大那条边；或用“不相似度分析”规则逐个把“更像 H、不像 G”的点移过去，直到没必要再移。

小结：两路都没有严格的目标函数保证；它们总会给你一棵树，哪怕数据是纯噪声。

3) 怎么定“在哪儿切树 (K) ”？

看树枝的“高度差”找“缺口”——理论上有用，实际常常看不太出来。

4) 贝叶斯层次聚类 (BHC) 救场

- 思路：与其拍脑袋“哪两簇像就并谁”，不如做一个**合并 vs 不合并的贝叶斯假设检验**。

- **若并**: 把两簇的数据视作来自同一分布, 算一个 (积分过 θ 的) **边际似然**。
 - **若不并**: 就把这两棵子树各自的**边际似然**相乘。
 - 乘以“**合并的先验概率**”, 按式子算出“**合并的后验概率** r_{ij} ”。
 - 算法像**凝聚式**一样自底向上, 但每一步是挑 r_{ij} 大的去并; 最后你甚至可以把所有 $r_{ij} < 0.5$ 的地方剪断, 得到稳妥的层次。
 - 与 DPMM 的联系: BHC 其实现的是在“与当前树一致的划分集合”里**近似最大化**边际似然 → 相当于 DPMM 边际似然的一个**下界**; 还能据此计算“**合并的先验概率**”。
 - **优点**: 有**概率意义**、更**稳健** (不会“无中生有”地硬凑层次), 在许多数据集上**纯度**明显赢传统层次法。
-

要点总结

- **层次聚类 = 分组的“树”**: 簇中有子簇, 切树的高度决定“分几组”。
- **凝聚式 (自底向上)** : 每次**合并**最像的两组; 复杂度可到 $O(N^2 \log N)$ 。
 - **单链接**: 看最近一对 → 可能“拉长串”, 不紧凑; 但等价于**最小生成树** ($O(N^2)$)。
 - **全链接**: 看最远一对 → 倾向**紧凑小直径簇**。
 - **平均链接**: 看平均 → 折中、常用; 对**尺度敏感**。
- **分裂式 (自顶向下)** : 每次把一个组**一分为二** (如二分 K-means); 可更快 (分固定层时 $O(N)$), 也更“全局看局势”。
- **挑簇数难**: 树总会长出来, 随机数据也不例外; 靠“树枝长度缺口”不稳定。
- **BHC (贝叶斯层次)** : 用**贝叶斯检验**决定是否合并; 与 DPMM 有紧密联系, 能给出**合并先验与边际似然下界**; 实践中**纯度**常优于传统层次法。
- **工程提示**: 大 N 时先用 K-means 预聚类再做层次; 需要可解释的层级时优先考虑层次法; 若希望有“统计学依据/不乱长层次”, 用 BHC。

一句话: 传统层次法给你一棵“看起来像样”的树; BHC 则给你一棵“在统计上站得住脚”的树。

这一节主要在讲什么? 我不太懂

已思考 11s

没问题! 我用大白话把 25.5 「层次聚类」的主线捋一遍, 读完就知道它在讲啥。

一句话版

这节讲：**怎样把聚类做成一棵“从粗到细”的树。**有两条路——**自底向上合并**和**自顶向下拆分**——都用“不相似度矩阵”当输入；最后得到一张**树状图（dendrogram）**，在不同高度“切树”就得到不同簇数。后面还给了一个**有概率学依据的版本：贝叶斯层次聚类（BHC）**。

核心内容（通俗版）

1. 两种套路

- **凝聚式（自底向上）**：一开始每个点自己一个组；每步把**最像的两组合**在一起，直到全并成一个组。产物就是树状图：树枝越高，表示合并那俩组越“远”。
- **分裂式（自顶向下）**：一开始所有点在一个组；每步把一个组**一分为二**（常用二分 K-means/K-medoids）。递归下去，也得到一棵树。

2. “两组有多远”的三种定义（凝聚式里最关键）

- **单链接（single link）**：看**最近那一对**成员的距离 → 容易拉成“长链”，不紧凑；但和**最小生成树**等价，快。
- **全链接（complete link）**：看**最远那一对** → 偏好紧凑的小直径簇。
- **平均链接（average link）**：看**所有配对的平均** → 折中、常用；但对尺度变化敏感。

3. 分裂式常见招

- 选“直径最大”的组，用 K=2 的 K-means/K-medoids 去劈（“二分 K-means”）。
- 也可先建**最小生成树**，每次砍掉**最大的**边来分组；或用“不相似度分析”按规则把更不像的点移出去。

4. 难点：选多少簇？

- 树总会长出来（哪怕全是噪声）。常靠观察树枝“高度差”的**缺口**来挑切点，但真实数据里**不稳定**。

5. 贝叶斯层次聚类（BHC）——给树一个统计学“靠山”

- 过程像凝聚式，但合并两簇时，不是“谁近合谁”，而是做**贝叶斯假设检验**：
 - **合并假设**：两簇数据来自**同一模型** → 算合在一起的**边际似然**；
 - **不合并**：两簇各自独立 → 把各自边际似然相乘；
 - 结合**合并的先验概率**，得到**合并的后验概率** r_{ij} ，大就合。最后也可在 $r_{ij} < 0.5$ 的地方“剪树”。
- **与 DPMM 的关系**：BHC近似在“与当前树一致的划分”里最大化 DPMM 的边际似然，相当于给出一个**下界**；还能据此计算“合并先验”。
- **效果**：很多数据上 BHC 的**纯度**优于传统层次法，而且不容易对**纯噪声**硬凑结构。

何时用它

- 你需要**可解释的层级结构** (类→子类→子子类)。
- 想先看**全局轮廓**再决定切多少簇 (可在不同高度切树)。
- 数据量很大时, 可先 K-means 预聚, 再对中心做层次, 省时。

要点速记

- 层次聚类=树**: 不同切高得到不同簇数。
- 凝聚式**三种 linkage: 单 (近一对, 可能拉丝)、全 (远一对, 紧凑)、平均 (折中常用)。
- 分裂式**: 自顶向下二分, 可能更快、决策更“全局”。
- 选 K 难**: 靠“缺口”不稳。
- BHC**: 用**贝叶斯检验**决定合并, 与 DPMM 紧密相关; 常比传统层次法更靠谱、指标更好。

如果你想, 我可以拿个小例子画出树状图, 演示不同高度切树会得到怎样的分组, 对直觉很有帮助。

翻译并解释以下内容, 翻译不能有遗漏, 解释请通俗易懂且详尽, 因为我的数学基础不太好, 最后总结要点。

25.6 Clustering datapoints and features

So far, we have been concentrating on clustering datapoints. But each datapoint is often described by multiple features, and we might be interested in clustering them as well. Below we describe some methods for doing this.

25.6.1 Biclustering

Clustering the rows and columns is known as biclustering or coclustering. This is widely used in bioinformatics, where the rows often represent genes and the columns represent conditions. It can also be used for collaborative filtering, where the rows represent users and the columns represent movies.

A variety of ad hoc methods for biclustering have been proposed; see (Madeira and Oliveira 2004) for a review. Here we present a simple probabilistic generative model, based on (Kemp et al. 2006) (see also (Sheng et al. 2003) for a related approach). The idea is to associate each row

and each column with a latent indicator, $r_i \in \{1, \dots, K_r\}$, $c_j \in \{1, \dots, K_c\}$. We then assume the data are iid across samples and across features within each block:

$$p(x|r, c, \theta) = \prod_{i,j} p(x_{ij}|r_i, c_j, \theta) = p(x_{ij}|\theta_{r_i, c_j}) \quad (25.62)$$

where $\theta_{a,b}$ are the parameters for row cluster a and column cluster b . Rather than using a finite number of clusters for the rows and columns, we can use a Dirichlet process, as in the infinite relational model which we discuss in Section 27.6.1. We can fit this model using e.g., (collapsed) Gibbs sampling.

The behavior of this model is illustrated in Figure 25.18. The data has the form $X(i, j)=1$ iff animal i has feature j , where $i = 1:50$ and $j = 1:85$. The animals represent whales, bears, horses, etc. The features represent properties of the habitat (jungle, tree, coastal), or anatomical properties (has teeth, quadrupedal), or behavioral properties (swims, eats meat), etc. The model, using a Bernoulli likelihood, was fit to the data. It discovered 12 animal clusters and 33 feature clusters. For example, it discovered a bicluster that represents the fact that mammals tend to have aquatic features.

25.6.2 Multi-view clustering

The problem with biclustering is that each object (row) can only belong to one cluster. Intuitively, an object can have multiple roles, and can be assigned to different clusters depending on which subset of features you use. For example, in the animal dataset, we may want to group the animals on the basis of anatomical features (e.g., mammals are warm blooded, reptiles are not), or on the basis of behavioral features (e.g., predators vs prey).

We now present a model that can capture this phenomenon. This model was independently proposed in (Shafto et al. 2006; Mansinghka et al. 2011), who call it crosscat (for crosscategorization), and in (Guan et al. 2010; Cui et al. 2010), who call it (non-parametric) multi-clust. (See also (Rodriguez and Ghosh 2011) for a very similar model.) The idea is that we partition the columns (features) into V groups or views, so $c_j \in \{1, \dots, V\}$, where $j \in \{1, \dots, D\}$ indexes features. We will use a Dirichlet process prior for $p(c)$, which allows V to grow automatically. Then for each partition of the columns (i.e., each view), call it v , we partition the rows, again using a DP, as illustrated in Figure 25.19(a). Let $r_{iv} \in \{1, \dots, K(v)\}$ be the cluster to which

the i 'th row belongs in view v . Finally, having partitioned the rows and columns, we generate the data: we assume all the rows and columns within a block are iid. We can define the model more precisely as follows:

$$p(c, r, D) = p(c)p(r|c)p(D|r, c) \quad (25.63)$$

$$p(c) = DP(c|\alpha) \quad (25.64)$$

$$p(r|c) = V(c) \quad v=1 \quad DP(rv|\beta) \quad (25.65)$$

$$p(D|r, c, \theta) = V(c) \quad v=1 \quad j:c=v \quad | \quad K(rv) \quad k=1 \quad i:rv=k \quad p(x_{ij} | \theta_{jk})p(\theta_{jk})d\theta_{jk} \quad | \quad (25.66)$$

See Figure 25.19(b) for the DGM.

If the data is binary, and we use a $Beta(\gamma, \gamma)$ prior for θ_{jk} , the likelihood reduces to

$$p(D|r, c, \gamma) = V(c) \quad v=1 \quad j:c=v \quad K(rv) \quad k=1 \quad Beta(nj,k, v + \gamma, nj,k, v + \gamma) \quad Beta(\gamma, \gamma) \quad (25.67)$$

where $nj,k,v = i:ri, v=k \quad I(x_{ij} = 1)$ counts the number of features which are on in the j 'th column for view v and for row cluster k . Similarly, nj,k,v counts how many features are off. The model is easily extended to other kinds of data, by replacing the beta-Bernoulli with, say, the Gaussian-Gamma-Gaussian model, as discussed in (Guan et al. 2010; Mansinghka et al. 2011).

Approximate MAP estimation can be done using stochastic search (Shafto et al. 2006), and approximate inference can be done using variational Bayes (Guan et al. 2010) or Gibbs sampling (Mansinghka et al. 2011). The hyper-parameter γ for the likelihood can usually be set in a noninformative way, but results are more sensitive to the other two parameters, since α controls the number of column partitions, and β controls the number of row partitions. Hence a more robust technique is to infer the hyper-parameters using MH. This also speeds up convergence (Mansinghka et al. 2011).

Figure 25.20 illustrates the model applied to some binary data containing 22 animals and 106 features. The figures shows the (approximate) MAP partition. The first partition of the columns contains taxonomic features, such as "has bones", "is warm-blooded", "lays eggs", etc. This divides the animals into birds, reptiles/ amphibians, mammals, and invertebrates. The second partition of the columns contains features that are treated as noise, with no apparent structure (except for the single row labeled "frog"). The third partition of the columns contains ecological features like "dangerous", "carnivorous", "lives

in water", etc. This divides the animals into prey, land predators, sea predators and air predators. Thus each animal (row) can belong to a different cluster depending on what set of features are considered. Uncertainty about the partitions can be handled by sampling.

It is interesting to compare this model to a standard infinite mixture model. While the standard model can represent any density on fixed-sized vectors as $N \rightarrow \infty$, it cannot cope with $D \rightarrow \infty$, since it has no way to handle irrelevant, noisy or redundant features. By contrast, the crosscat/multi-clust system is robust to irrelevant features: it can just partition them off, and cluster the rows only using the relevant features. Note, however, that it does not need a separate "background" model, since everything is modelled using the same mechanism. This is useful, since one's person's noise is another person's signal. (Indeed, this symmetry may explain why multi-clust outperformed the sparse mixture model approach of (Law et al. 2004) in the experiments reported in (Guan et al. 2010).)

已思考 14s

下面先给出**完整中文翻译**（逐段对应原文，保留公式与编号；对明显拼写小误如 *Dirchlet*→*Dirichlet*、*quadrapedal*→*quadrupedal* 仅作不改义的自然更正），随后是**通俗解释**，最后是**要点总结**。

译文

25.6 同时聚类数据点与特征 (Clustering datapoints and features)

到目前为止，我们一直专注于对**数据点**做聚类。但每个数据点通常由多个**特征**来描述，而我们可能也希望对**特征**进行聚类。下面我们将介绍一些方法。

25.6.1 双聚类 (Biclustering)

同时对行与列做聚类被称为**双聚类** (biclustering) 或**协同聚类** (coclustering)。这在生物信息学中被广泛使用，其中行通常代表基因，列代表条件。它也可用于协同过滤，其中行代表用户，列代表电影。

人们提出了多种**特设 (ad hoc)** 的**双聚类方法**；综述见 (Madeira and Oliveira 2004)。这里我们给出一个**简单的概率生成模型**，基于 (Kemp et al. 2006) (另见 (Sheng et al. 2003) 中相关方法)。其思想是为每个行与每个列关联一个**潜在指示变量**，分别为 $r_i \in$

$\{1, \dots, K_r\}$ 、 $c_j \in \{1, \dots, K_c\}$ 。随后我们假设在每个块内，样本间与特征间都是独立同分布 (iid)：

$$p(x | r, c, \theta) = \prod_i \prod_j p(x_{ij} | r_i, c_j, \theta) = \prod_{i,j} p(x_{ij} | \theta_{r_i, c_j}) \quad (25.62)$$

其中 $\theta_{a,b}$ 是“行簇 a ”与“列簇 b ”对应的参数。与其为行、列使用**有限个簇**，我们可以像第 27.6.1 节讨论的**无限关系模型**那样，使用**Dirichlet 过程 (DP)**。该模型可用（折叠）Gibbs 采样等方法拟合。

图 25.18 展示了该模型的行为。数据形式为 $X(i, j) = 1$ 当且仅当动物 i 具有特征 j ，其中 $i = 1:50$ 、 $j = 1:85$ 。动物包括鲸、熊、马等；特征包括栖息地（丛林、树栖、海岸）、解剖属性（有牙、四足 *quadrupedal*）、行为属性（会游泳、吃肉）等。在**伯努利似然**下拟合该模型，得到 12 个动物簇与 33 个特征簇。例如，模型发现了一个**双块** (bicluster)，表达“哺乳动物往往具有水栖特征”。

25.6.2 多视角聚类 (Multi-view clustering)

双聚类的一个问题是：每个对象（行）**只能属于一个簇**。直观上，一个对象可以有**多重角色**，并且在使用**不同特征子集**时会被分到不同簇。例如，在动物数据集中，我们可能希望基于**解剖特征**来分（如哺乳类是温血，爬行类不是），也可能基于**行为特征**来分（如捕食者 vs 猎物）。

下面我们给出一个能表达该现象的模型。该模型由 (Shafto et al. 2006; Mansinghka et al. 2011) 独立提出，称为 *crosscat*（意为“交叉分类”）；也由 (Guan et al. 2010; Cui et al. 2010) 独立提出，称为（非参数）*multi-clust*。（另见 (Rodriguez and Ghosh 2011) 中非常相似的模型。）思路是：**将列（特征）划分为 V 个组/视角（views）**，令 $c_j \in \{1, \dots, V\}$ ，**其中 $j \in \{1, \dots, D\}$ 为特征索引**。我们对 $p(c)$ 使用**Dirichlet 过程**先验，使 V 能够自动增长。然后，对每个“列的划分”（即每个视角） v ，我们再次使用 DP 将**行**划分为若干簇，如图 25.19(a) 所示。记 $r_{iv} \in \{1, \dots, K^{(v)}\}$ 为第 i 行在视角 v 下所属的簇。最后，在完成“行—列”的划分后，我们生成数据：假设每个**块**中的所有单元格 iid。模型可更精确地表述为

$$p(c, r, D) = p(c)p(r | c)p(D | r, c) \quad (25.63)$$

$$p(c) = \text{DP}(c | \alpha) \quad (25.64)$$

$$p(r | c) = \prod_{v \in V(c)} \text{DP}(r_v | \beta) \quad (25.65)$$

$$p(D | r, c, \theta) = \prod_{v \in V(c)} \prod_{j: c_j=v} \left[\prod_{k=1}^{K^{(v)}} \int_{i: r_{iv}=k} \prod_{j: c_j=v} p(x_{ij} | \theta_{jk}) p(\theta_{jk}) d\theta_{jk} \right] \quad (25.66)$$

其有向图模型见图 25.19(b)。

若数据是**二值的**，并对 θ_{jk} 使用 $\text{Beta}(\gamma, \gamma)$ 先验，则似然化为

$$p(D | r, c, \gamma) = \prod_{v \in V(c)} \prod_{j: c_j=v} \prod_{k=1}^{K^{(v)}} \frac{\text{Beta}(n_{j,k,v}^{(1)} + \gamma, n_{j,k,v}^{(0)} + \gamma)}{\text{Beta}(\gamma, \gamma)} \quad (25.67)$$

其中 $n_{j,k,v}^{(1)} = \sum_{i: r_{i,v}=k} \mathbf{I}(x_{ij} = 1)$ 统计了在“视角 v 、行簇 k 、列 j ”这个块中**为 1**的个数；相应地， $n_{j,k,v}^{(0)}$ 统计**为 0**的个数。该模型可容易地扩展到其他类型数据：将 **Bernoulli** 替换为如 **Gaussian–Gamma–Gaussian** 等（见 Guan et al. 2010; Mansinghka et al. 2011）。

近似 MAP 可用随机搜索 (Shafto et al. 2006) 实现；**近似推断可用变分 Bayes** (Guan et al. 2010) 或 **Gibbs 采样** (Mansinghka et al. 2011)。似然的超参数 γ 通常可用**非信息化**方式设定，但结果对另两个参数更敏感： α 控制**列视角的数量**， β 控制**行簇的数量**。因此更稳健的做法是用 **Metropolis–Hastings (MH)** 推断这些超参数，这也能加速收敛 (Mansinghka et al. 2011)。

图 25.20 展示了将该模型应用到一个包含 22 个动物、106 个特征的**二值**数据。图中给出了（近似）MAP 划分。**第一组**特征（第一视角）包含**分类学**特征，如“有骨骼”“是温血”“会产卵”等，它把动物分成**鸟类、爬/两栖类、哺乳类、无脊椎**。**第二组**特征被模型视作**噪声**，没有明显结构（除了“青蛙”这一行单独显现）。**第三组**特征包含**生态学**属性，如“危险”“食肉”“生活在水中”等，它把动物分成**猎物、陆地捕食者、海洋捕食者、空中捕食者**。因此，同一只动物（行）会**随所考虑的特征集合不同而归入不同簇**。对划分的不确定性可以通过**采样**来处理。

把该模型与标准的**无限混合模型**（如 DPMM）对比很有趣。标准模型在**固定维度向量**的情况下，当 $N \rightarrow \infty$ 时可以表示任意密度，但当 $D \rightarrow \infty$ 时就**无能为力**，因为它无法处理**无关、噪声或冗余特征**。相反，crosscat/multi-clust 系统对无关特征很鲁棒：它会把这些特征**单独划分出去**，仅用**相关**特征对行做聚类。注意，它**不需要**单独的“背景”模型，因为一切都用同一机制建模。这很有用——“甲的噪声可能是乙的信号”。（事实上，这种对称性也许解释了 multi-clust 在 (Guan et al. 2010) 报告的实验中，**优于** (Law et al. 2004) 的**稀疏混合模型方法**。）

通俗解释（不靠公式，把直觉讲清）

这节在说啥？

之前我们只给“样本（行）”分组。现在我们还想：**给特征（列）也分组**；甚至更进一步

——按不同特征子集来给样本“多视角”分组。两步走：

1) 双聚类：把表格切成“棋盘格”小块

- 想像一个“动物 × 特征”的 0/1 表。双聚类同时把行分组、把列分组，于是表被切成很多“小方块”（块内同质、块间差异大）。
- 概率版做法：给每个行一个“行簇标签” r_i ，给每个列一个“列簇标签” c_j ，同一个“行簇 × 列簇”的交叉块，共享同一套参数（比如“这一块里 1 的概率是 p ”）。
- 不想手动定“行簇个数/列簇个数”？用 Dirichlet 过程让它自动长。
- 直觉：它能抓到“哪类动物 × 哪类特征会经常一起出现”的主题性关联（比如“哺乳动物 → 水栖特征”）。

局限：每个行在双聚类里只有一个行簇标签。但现实里同一只动物在“解剖学视角”和“生态学视角”下应有**不同的分组**。

2) 多视角聚类 (crosscat / multi-clust) : 先分“列的视角”，再分“行的簇”

- 第一步：把**特征**先分成若干**视角 (views)**：例如“分类学特征”“生态学特征”“无结构的噪声特征”等；用 DP，让视角数**自动决定**。
- 第二步：在**每个视角里**，再把行（样本）分组（依然用 DP）。这就允许同一个样本在不同视角里**属于不同簇**——恰好表达“多重角色”。
- 第三步：在每个“视角 $v \times$ 行簇 $k \times$ 列 j ”的**块**里，根据数据类型（0/1、实数等）选个合适的**似然-先验**（二值用 Beta-Bernoulli，连续用高斯族），所有单元格 iid。
- 好处：
 - 抗无关特征**：把“没啥结构的列”单独成一个视角就行，不会拖垮聚类；
 - 多面刻画**：同一对象在不同视角下呈现不同的“群体关系”；
 - 超参数**： α 决定“会切出多少个列视角”， β 决定“每个视角里会有多少个行簇”， γ 只是对二值似然的平滑； α, β 更关键，常用 MH 一起抽样更稳更快。

动物案例（易懂版）：

- 视角 A（分类学）：把“有骨骼/温血/产卵”等放一起 → 动物被分成鸟、爬/两栖、哺乳、无脊椎。
- 视角 B（噪声）：大多列无结构 → 基本忽略（除了“青蛙”出挑）。
- 视角 C（生态学）：把“危险/食肉/水生/空中”等放一起 → 动物分成猎物、陆捕、海捕、空捕。

同一动物在 A 与 C 里的标签可能不同，这正是“多视角”的价值。

对比标准（无限）混合模型：

普通混合（即使无限簇）默认同一套簇标签解释所有特征；当维度很大、无关列很多时，

它容易被“噪声”稀释。crosscat先把列分视角，再在每个视角里分行，天然地“用相关列聚类，屏蔽无关列”。

要点总结

- **双聚类**：同时给行与列分组，表被切成“块”；块内共享参数，能抓“行类 × 列类”的共现模式；用 DP 可省去手动定簇数。
- **痛点**：双聚类里一个对象只有一个行簇标签，无法表达“按不同特征子集分组不同”。
- **多视角聚类 (crosscat / multi-clust)**：
 - 先用 DP 把列分成若干视角；
 - 再在每个视角里用 DP 给行分簇；
 - 每个“视角 × 行簇 × 列”的块里按数据类型选相应的似然-先验（如 Beta-Bernoulli），单元格 iid。
 - α 控列视角数量、 β 控行簇数量、 γ 为似然平滑； α, β 通常用 MH 一起推断更稳。
- **优势**：
 - 允许同一对象在不同视角有不同聚类（多重角色）；
 - 对无关/噪声特征鲁棒（把它们单独成视角）；
 - 不需要单独“背景模型”，一套机制统一处理。
- **应用图景**：基因表达（基因×条件）、协同过滤（用户×物品）、高维表格的探索性分析与可解释聚类。
- **与标准混合的区别**：标准混合（即便无限）对所有特征用同一簇标签；crosscat 先按列分视角再按行分簇，更适合 D 大、特征异质/冗余的场景。

一句话：**双聚类**把表切成“棋盘格”，发现“行类×列类”的块结构；**多视角聚类**再进一步，让“同一行在不同特征视角下各有其群体”，既能屏蔽噪声列，又能刻画对象的多面性。

这一节主要在讲什么？我不太懂

已思考 8s

没问题！我用大白话把 25.6 的主线捋清楚，读完就知道它想解决啥。

一句话版

这节课讲：**不仅给样本分组，还给特征分组**。先是**双聚类**（行和列同时分），再是更灵活的**多视角聚类**（先把特征按“视角”分组，每个视角里再给样本分组），而且用**Dirichlet 过程**让簇数/视角数自动长出来，能抗“无关特征”，也能体现同一样本的多重角色。

核心内容（通俗解释）

1. 双聚类 (biclustering / coclustering)

- 把“样本×特征”的表切成很多**小方块**：行被分成几类、列也被分成几类；每个“行类×列类”的交叉块里的数据**长得像**。
- 概率版做法：给每行一个行簇标签 r_i 、给每列一个列簇标签 c_j ，同一块共享参数；也可用 **Dirichlet 过程**，不用手定簇数。
- 适用：基因表达（基因×条件）、协同过滤（用户×电影）等，能发现“哪类样本与哪类特征**共现**”。

2. 多视角聚类 (crosscat / multi-clust)

- 痛点：双聚类里**每个样本只能有一个行簇**，可现实里同一个体在不同特征子集下**会分到不同群**。
- 方案：
 - 先把**特征**按语义切成若干**视角** (views)（比如“解剖学”“生态学”“噪声”），用 DP 自动决定视角数（参数 α 控制）。
 - 再在**每个视角里**，用 DP 给**样本**分簇（参数 β 控制）。
 - 每个“视角×样本簇×特征”构成的**块**里，数据独立同分布；二值用 Beta-Bernoulli，连续用高斯族等。
- 好处：
 - **同一样本在不同视角可进不同簇**（体现多重角色，如“分类学”分法 vs “捕食/猎物”分法）。
 - 对**无关/噪声特征鲁棒**：把它们单独成一个视角，不影响其他视角的聚类。
 - 不用单独的“背景模型”，一套机制统一处理。
- 推断：可用随机搜索（近似 MAP）、变分、Gibbs； α, β 常用 MH 一起抽，更稳更快。

3. 和标准（无限）混合模型的区别

- 标准混合模型用**一套簇标签解释所有特征**，在高维、噪声多时容易被稀释；
- 多视角模型先**按列分视角**，再**按行分簇**，天然会**屏蔽无关特征**，只用相关列做聚类。

什么时候用

- 你既关心样本如何分，也关心**哪些特征彼此相近、成主题**；

- 数据维度很高、**无关特征不少**；
- 同一对象在不同特征子集下**应当有不同分组**的场景（比如动物：按解剖 vs 按生态）。

要点速记

- **双聚类**：行、列一起分 → 表被切成“棋盘格”小块；可用 DP，发现“样本类×特征类”的共现结构。
- **多视角聚类**：先把特征分成若干**视角** (α 控视角数)，每个视角里再把样本分簇 (β 控簇数)；块内 iid，二值/连续都能适配。
- **优势**：同一样本可在不同视角进不同簇；对无关特征鲁棒；不需额外背景模型。
- **对比标准混合**：标准模型一套簇解释所有列；多视角能先把**噪声列分出去**，更适合高维异质数据。

一句话收尾：25.6 = 给“样本×特征”表做结构化分解——先学“行类×列类”的块（双聚类），再学“按不同特征视角的样本分组”（多视角），既发现主题，也抗噪声。

基于本对话里我贴过的整章原文（19章），输出：
①知识点总结（分点+关键词+用途）②子章节关联（依赖/对比/递进+关系句，附Mermaid可选）③本章总结。

已思考 7s

① 知识点总结（分点 + 关键词 + 用途）

- **聚类的两类输入**
 - **关键词**：相似度/距离矩阵 S/D ；特征矩阵 X
 - **用途**：决定可用算法族。基于相似度的算法（AP、谱聚类、层次法）不需显式生成模型；基于特征的算法（混合模型、BHC、CrossCat）可做概率推断与不确定性度量。
- **聚类的两类输出**
 - **关键词**：扁平（partitional）；层次（hierarchical、dendrogram）
 - **用途**：扁平适合一次性分组；层次可在不同“切高”得到多尺度结构，利于探索和解释。
- **(不)相似度度量**
 - **关键词**：欧氏/曼哈顿；相关系数；有序编码；汉明距离
 - **用途**：让“相似”的定义与数据类型匹配，提升算法效果与稳健性。

- 内部/外部评估指标
 - **关键词:** Purity; Rand/Adjusted Rand (ARI); 互信息/NMI
 - **用途:** 有标签时做外部对照比较聚类好坏；不同方法间可量化对比。
- 有限/无限混合模型 (DPMM)
 - **关键词:** Dirichlet 过程；折棒 (GEM)；中华餐馆过程 (CRP)；折叠 Gibbs
 - **用途:** 不用预设 K，随数据量自适应增长簇数；常较快跳出局部最优；能做贝叶斯不确定性分析。
- 亲和传播 (AP)
 - **关键词:** exemplar (代表点)、责任 r 、可用度 a 、阻尼
 - **用途:** 只需相似度矩阵，自动决定簇数；在K-medoids易陷局部时表现更稳。
- 谱聚类 (Spectral)
 - **关键词:** 图割/Ncut；图拉普拉斯 $L=D-W$ ；最小 K 个特征向量； L_{rw}/L_{sym} 归一化
 - **用途:** 把非凸/弯曲形状的簇分得更好；适合用稀疏近邻图扩展到较大规模。
- 层次聚类 (Agglomerative/Divisive)
 - **关键词:** 单/全/平均链接；最小生成树（单链接）；二分K-means
 - **用途:** 给出可解释的树状图；探索多尺度分组结构；先K-means再层次可提速。
- 贝叶斯层次聚类 (BHC)
 - **关键词:** 合并 vs 不合并的贝叶斯检验；边际似然；与DPMM的联系
 - **用途:** 用统计证据决定是否合并，减少“噪声也长树”的风险；在多数据集上 purity 优于传统层次法。
- 选择簇数 (K) /切高
 - **关键词:** 谱间隙；树枝长度“缺口”；后验/边际似然 (BHC/DPMM)
 - **用途:** 启发式+贝叶斯证据结合，避免拍脑袋定 K。
- 双聚类 (Bioclustering)
 - **关键词:** 行簇 \times 列簇的块结构；DP 扩展
 - **用途:** 发现“样本类与特征类的共现模式”（如基因 \times 条件、用户 \times 物品）。
- 多视角聚类 (CrossCat / Multi-clust)
 - **关键词:** 列先分“视角” (DP, α)；每个视角内行再分簇 (DP, β)；块内 iid (Beta–Bernoulli/高斯族)
 - **用途:** 同一样本在不同特征子集下可归不同簇；对无关特征鲁棒（单独成视角），适合高维异质数据。
- 计算与工程要点

- **关键词：**稀疏图 $O(E)$; 近似特征分解/Nyström; 阻尼; 优先队列 $O(N^2 \log N)$
- **用途：**把方法落地到大规模数据，提升收敛与稳定性。

② 子章节关联（依赖 / 对比 / 递进 + 关系句，附 Mermaid）

关系句（简要版）

- 25.1 → **全章**: 给出输入/输出范式与评估指标，是后续方法的**共同语言**。
- 25.2 vs **传统混合**: DPMM 通过 DP/CRP **避免预设 K**，与 25.5 的 BHC 在后验与边际似然思路上**互通**。
- 25.3 vs 14.4.2 (**K-medoids**) : AP 同为基于相似度的**代表点方法**，但用消息传递**更稳且自动定簇数**。
- 25.4 vs K-means: 谱聚类把数据投到拉普拉斯特征空间，再 K-means，**弥补非凸形状劣势**；亦与 KPCA 有联系。
- 25.5 (**层次**) → 25.2/25.5.4: 传统层次法是启发式；BHC引入贝叶斯证据，与 DPMM 的边际似然**建立桥梁**。
- 25.6 vs 25.2: CrossCat 在列上先 DP 分视角，再在各视角内对行做 DP 聚类，是“**DPMM 在二维表上的分解式扩展**”；比“一套簇解释所有列”的混合模型更抗噪声。

适配场景（对比）

- 相似度已知 → AP/谱/层次；原始特征且要概率解释 → DPMM/BHC；要块结构/多角色 → 双聚类/CrossCat。

```
graph TD
    A[25.1 基本概念  
输入/输出/距离/评估] --> B[25.2 DPMM  
DP/CRP/折棒]
    A -.-> C[25.3 亲和传播  
相似度+消息传递]
    A -.-> D[25.4 谱聚类  
图割/拉普拉斯/特征向量]
    A -.-> E[25.5 层次聚类  
凝聚/分裂/链接]
    E -.-> F[25.5.4 BHC  
贝叶斯检验/与DPMM联系]
    A -.-> G[25.6 双聚类/多视角  
行列同聚/列视角+行簇]
    B -. 边际似然/非参贝叶斯 .-> F
    D -. 与KPCA相关 .-> A
    C -. 对比K-medoids .-> A
    G -. 扩展DPMM到二维表 .-> B
```

③ 本章总结

本章系统地把“**如何定义相似、如何表示数据、如何评估好坏**”作为底座（25.1），在此之上铺开四条主线：

1. 概率模型线 (25.2 DPMM, 25.5.4 BHC) :

用 Dirichlet 过程与后验/边际似然把“簇数”和“合并决策”交给数据与先验，**避免拍脑袋定 K**，还能给出不确定性与统计显著性。这条线适合需要**可解释、可置信**的应用。

2. 相似度图线 (25.3 AP, 25.4 谱聚类, 25.5 传统层次) :

不写生成式模型，直接在相似度上做优化或消息传递。AP自动挑代表点并定簇数；谱聚类通过图拉普拉斯的特征向量把**非凸结构**摊平；**层次法**提供**多尺度**视角与可解释的树。它们往往**实现简单、效果直观**，对相似度设计依赖更强。

3. 层次结构与贝叶斯化 (25.5) :

传统层次法给出可解释树，但**缺目标函数**、易在噪声上“长树”。BHC用贝叶斯检验加以矫正，并与 DPMM 建立原则性联系，**把层次也做成概率的**。

4. 样本×特征的双向结构 (25.6) :

双聚类发掘“行类×列类”的块；**多视角聚类 (CrossCat)** 先按列分视角，再在各视角内按行分簇，让**同一样本在不同特征子集下有不同分组**，并天然**屏蔽无关特征**，适合高维异质表格。

整体来看，本章给了一个“聚类武器库”：

- 需要**无参自适应 K** → 用 DPMM；
- 只有相似度，希望**稳且自动挑中心** → AP；
- 簇形状**非凸/弯曲** → 谱聚类；
- 要**可解释层级** → 先**层次聚类**，若需统计保证 → BHC；
- 要同时考虑**样本与特征或面对高维噪声列** → **双聚类 / CrossCat**。

把这些方法与数据类型、业务目标和计算资源匹配，你就能为绝大多数聚类任务挑到合适的“家伙”。

请用通俗易懂的话总结本章，使我能够理解

已思考 14s

好的！我用尽量不含公式的方式，把整章“聚类”讲清楚。看完你能知道：什么是聚类、常见方法在做什么、各自适合什么场景。

这章在讲啥（一句话）

聚类就是把“相似的东西放一起”。本章给了你一套从“怎么衡量相似”到“怎么选方法/评估效果”的完整工具箱：既有**基于模型**（有概率解释）的，也有**基于相似度**（不写生成模型，直接拿相似矩阵干）的；既有**扁平分组**，也有**层次结构**；还能**同时给样本和特征分组**，甚至按不同“视角”给同一样本多种分法。

1) 基本概念 (共同语言)

- **两种输入**
 - 特征矩阵 X : 每行一个样本，每列一个特征。
 - 相似/距离矩阵 S/D : 直接给出任意两样本之间的相似程度。
 - **两种输出**
 - **扁平**: 一次性分成 K 组。
 - **层次**: 形成一棵树 (dendrogram)，在不同高度“切”得到不同簇数。
 - **相似怎么量**
 - 连续特征: 欧氏/曼哈顿、相关系数。
 - 有序/分类特征: 有序数值化、汉明距离等。
 - **怎么评估** (有标签时更好用)
 - Purity、Rand/ARI、互信息/NMI: 度量“你的分组”跟“真实标签/参考分组”有多一致。
-

2) 方法家族 (核心直觉 + 适用点)

A. 基于概率模型

- **有限混合**: 假设有 K 个“成分”(如高斯)，但需要**先选 K** 。
- **Dirichlet 过程混合 (DPMM)** : 不用先定 K ，簇数随数据自动长；直觉来自“折棒/中华餐馆”比喻；常用折叠 Gibbs 做推断。
适用: 希望有**不确定性和概率解释**，并且不想拍脑袋选 K 。

B. 只用相似度矩阵

- **亲和传播 (AP)** : 让每个点“票选代表点(中心)”，用两类消息(责任/可用度)来回传，**自动决定簇数**；可通过调对角元素控制“谁更愿当中心”。
适用: 只有相似度；K-medoids 易陷局部时想要更稳的效果。
- **谱聚类**: 把数据看成图，找“好切法”(归一化割)，求拉普拉斯矩阵的**最小 K 个特征向量**，再在这个新空间里跑 K-means。
强项: 能把**弯的/非凸**形状分好；用稀疏近邻图可扩展。
- **层次聚类 (传统)** :
 - **凝聚式** (自底向上) : 每次合并最近的两簇 (单/全/平均链接定义“簇间距离”)。
 - **分裂式** (自顶向下) : 每次把一个簇一分为二 (如二分 K-means)。**产出**: 一棵可解释的树，但本质是启发式。
 - **BHC (贝叶斯层次)** : 用**贝叶斯假设检验**决定合并与否，和 DPMM 有紧密联系，给层次结构提供统计依据。

C. 同时给“样本”和“特征”分组

- **双聚类**: 行 \times 列一起分, 表被切成“棋盘格小块”, 能发现“样本类 \times 特征类”的共现模式。
 - **多视角聚类 (CrossCat / Multi-clust)** : 先把**特征**按语义分成若干“视角”, 再在每个视角里给**样本分簇**; 同一样本在不同视角可进不同簇; 对**无关特征**很鲁棒。
-

3) 选方法的“路标” (实用)

- 不知道 K , 要概率解释 → DPMM; 要层次且有统计证据 → BHC。
- 只有相似度 → AP / 谱聚类 / 层次 (看形状和是否要层次)。
- 簇形状弯/非凸 → 谱聚类。
- 要自动挑中心、较稳 → AP。
- 需要可解释的多尺度结构 → 层次聚类 (传统或 BHC)。
- 高维表格、噪声特征多 → 多视角聚类; 要发现“样本类 \times 特征类”共现 → 双聚类。

工程提示: 相似度图尽量做稀疏 (k 近邻), 特征分解用近似/迭代方法; 层次合并用优先队列提速; AP 用阻尼防振荡; 评估尽量配外部标签或做稳定性分析。

4) 常见坑

- 相似度定义不当 → 再好的算法也徒劳; 先做**标准化/尺度处理**。
 - 启发式的层次法会“无中生有” (噪声也长树) → 结合 BHC/后验证据更稳。
 - 高维+噪声列会稀释信号 → 用**多视角**先把无关列“分出去”。
 - 只看 Purity 容易被“多簇”欺骗 → 结合 ARI/NMI 等指标。
-

5) 一页口袋卡片 (背下来就会用)

- **输入**: 特征矩阵 or 相似矩阵; **输出**: 扁平 or 层次。
- **评估**: Purity / ARI / NMI (有标签更可靠)。
- **方法** → **场景**:
 - DPMM: 无须定 K , 要不确定性;
 - AP: 相似度输入, 自动中心;
 - 谱聚类: 非凸形状;
 - 层次→BHC: 要树且要统计支撑;
 - 双/多视角: 样本 \times 特征块结构, 或高维噪声多。

- **先做对：**数据清洗与相似度设计 > 算法细节。
-

总结收尾

这章不是要你记公式，而是让你**按数据形态和目标选工具**：

- **要自适应簇数/不确定性** → 走贝叶斯线 (DPMM/BHC)；
- **只拿得到相似度或簇形状弯曲** → 走相似度图线 (AP/谱/层次)；
- **既想分样本又想分特征，还要抗噪声** → 走双聚类/多视角。

掌握这套“路标”，大多数聚类问题你都能对号入座、下手有数。