

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。

26 Graphical model structure learning

26.1 Introduction

We have seen how graphical models can be used to express conditional independence assumptions between variables. In this chapter, we discuss how to learn the structure of the graphical model itself. That is, we want to compute $p(G|D)$, where G is the graph structure, represented as an $V \times V$ adjacency matrix.

As we discussed in Section 1.3.3, there are two main applications of structure learning: knowledge discovery and density estimation. The former just requires a graph topology, whereas the latter requires a fully specified model. The main obstacle in structure learning is that the number of possible graphs is exponential in the number of nodes: a simple upper bound is $O(2^V(V-1)/2)$. Thus the full posterior $p(G|D)$ is prohibitively large: even if we could afford to compute it, we could not even store it. So we will seek appropriate summaries of the posterior. These summary statistics depend on our task.

If our goal is knowledge discovery, we may want to compute posterior edge marginals, $p(G_{st} = 1|D)$; we can then plot the corresponding graph, where the thickness of each edge represents our confidence in its presence. By setting a threshold, we can generate a sparse graph, which can be useful for visualization purposes (see Figure 1.11).

If our goal is density estimation, we may want to compute the MAP graph, $\hat{G} \in \operatorname{argmax}_G p(G|D)$. In most cases, finding the globally optimal graph will take exponential time, so we will use discrete optimization methods such as heuristic search. However, in the case of trees, we can find the globally optimal graph structure quite efficiently using exact methods, as we discuss in Section 26.3.

If density estimation is our only goal, it is worth considering whether it would be more appropriate to learn a latent variable model, which can capture correlation between the visible variables via a set of latent common causes (see Chapters 12 and 27). Such models are often easier to learn and, perhaps more importantly, they can be applied (for prediction purposes) much more efficiently, since they do not require performing inference

in a learned graph with potentially high treewidth. The downside with such models is that the latent factors are often unidentifiable, and hence hard to interpret. Of course, we can combine graphical model structure learning and latent variable learning, as we will show later in this chapter.

In some cases, we don't just want to model the observed correlation between variables; instead, we want to model the causal structure behind the data, so we can predict the effects of manipulating variables. This is a much more challenging task, which we briefly discuss in Section 26.6.

已思考 22s

下面先给出逐字逐句的完整翻译（无遗漏），然后是通俗详尽的解释，最后是要点小结。

翻译

26 图模型结构学习

26.1 引言

我们已经看到，图模型可以用来表达变量之间的条件独立假设。在本章中，我们讨论如何学习图模型本身的结构。也就是说，我们想要计算 $p(G | D)$ ，其中 G 是图结构，用一个 $V \times V$ 的邻接矩阵来表示。

正如我们在第 1.3.3 节所讨论的，结构学习主要有两类应用：知识发现 (knowledge discovery) 和密度估计 (density estimation)。前者只需要图的拓扑结构，而后者需要一个完全指定的模型。结构学习的主要障碍在于：可能图的数量随节点数呈指数级增长；一个简单的上界是 $O(2^{V(V-1)/2})$ 。因此完整的后验 $p(G | D)$ 大得令人望而却步：即使我们有能力把它算出来，也无法把它存储下来。所以我们将寻求对该后验进行合适的概括 (summary)。这些概括性的统计量取决于我们的任务。

如果我们的目标是知识发现，我们可能想计算后验边（缘）概率，即 $p(G_{st} = 1 | D)$ ；然后我们可以画出相应的图，其中每条边的粗细表示我们对其存在与否的置信度。通过设置一个阈值，我们可以生成一个稀疏图，这在可视化方面很有用（见图 1.11）。

如果我们的目标是密度估计，我们可能想计算 MAP 图， $\hat{G} \in \arg \max_G p(G | D)$ 。在多数情况下，找到全局最优的图需要指数时间，因此我们会使用离散优化方法，比如启发式搜索。然而，在树这一特殊情形下，我们可以用精确方法相当高效地找到全局最优的图结构，我们将在第 26.3 节讨论。

如果密度估计是我们唯一的目标，那么值得考虑学习一个潜变量模型是否更为合适；这类模型可以通过一组潜在的共同原因来刻画可见变量之间的相关性（见第 12 章和第 27 章）。这类模型通常更容易学习，而且或许更重要的是，它们在应用（用于预测）时要高效得多，因为它们不需要在一个可能具有很高树宽的已学习图上执行推断。这类模型的缺点在于潜在因子往往不可识别，因此难以解释。当然，我们也可以把图模型结构学习和潜变量学习结合起来，正如我们将在本章后面展示的那样。

在某些情况下，我们不仅想建模变量之间的观察到的相关性；相反，我们想建模数据背后

的因果结构，以便预测操纵变量所产生的效果。这是一个更具挑战性的任务，我们将在第26.6 节中简要讨论。

通俗详解

1) 我们在学什么？

- 数据里有很多变量（比如基因、症状、股票），它们彼此有关联。
- “图模型”用“点（变量）+边（变量间关系）”来描述这些联系，还能表达“条件独立”（某些变量在知道第三个变量后就互相独立）。
- “结构学习”就是：从数据中学出“哪些点之间应该连边”，也就是学出那张图本身。

2) 为什么要学结构？两种主要用途

- **知识发现**：想知道“谁和谁相关、结构大概什么样”。只要图的形状（拓扑）即可，用来看做科学发现、可视化展示。
- **密度估计**：想得到一个能很好逼近“数据分布 $p(x)$ ”的完整模型，方便预测、生成、缺失值填补等。这就不仅要图的形状，还要边上、节点上的参数。

3) 为什么完整求解很难？（组合爆炸）

- V 个节点的无向图，一共有 $\frac{V(V-1)}{2}$ 条可能的边；每条边要么在、要么不在 → 总可能数大约是 $2^{V(V-1)/2}$ 。
- 这增长太快了：比如 $V = 20$ ，可能图数量就是 2^{190} （约 10^{57} 量级），几乎不可能把每个图的后验概率都算完并存下来。
- 所以我们不追求“把整个后验分布 $p(G | D)$ 都握在手里”，而是**挑有用的摘要**。

4) 任务不同，摘要也不同

- **做知识发现**：
 - 关注每一条边存在的后验概率 $p(G_{st} = 1 | D)$ 。
 - 画图时让边的粗细反映置信度；再设个阈值，把“置信度低的边”砍掉，得到稀疏、更易读的网络。
- **做密度估计**：
 - 找一个“最可能的图”——MAP 图 $\hat{G} = \arg \max_G p(G | D)$ 。
 - 直接穷举找全局最优通常要**指数时间**，现实中常用**启发式搜索**（像爬山法、模拟退火、贪心添加/删除边等）。
 - 但有个**重要特例**：如果我们限制结构是树（每两个点之间只有一条路径），就能用**精确且高效**的方法找到全局最优（经典做法是把“边的权重”设为互信息，然后求**最大生成树**，也称 Chow-Liu 方法）。

5) 只为密度估计时，何必学复杂结构？看看潜变量模型

- **潜变量模型**（如因子分析、混合模型、隐马尔可夫模型等）通过“看不见的共同原因”来解释可见变量的相关性。
- 优点：
 - **更容易学**（参数更规整；优化更顺滑）。
 - **用起来更快**：预测时不用在高树宽（图的“复杂度”指标，越大推断越慢）的图上做复杂推断。
- 缺点：
 - **可解释性差**：潜因子常常**不可识别**（不同的潜因子组合可能产生一样的观测分布），所以很难给出“这个因子究竟代表什么”的直观解释。
 - 折中方案：**把结构学习和潜变量一起用**（既有可解释的边，又允许潜在共同原因）。

6) 相关 vs 因果：更难的一关

- 上面大多在学**相关结构**（谁跟谁一起变化）。
 - 如果你想知道**干预**会怎样（“把变量 X 强行设成 1，会导致 Y 变吗？”），那要学的是**因果结构**，难度更高，需要不同的假设与方法（如干预数据、工具变量、因果图等）。本章 26.6 节会简单提到。
-

要点小结

- **目标**：从数据中学习图模型的**结构** G ，形式上在估计后验 $p(G | D)$ 。
- **两大应用**：
 1. **知识发现**：只要图拓扑；看**边的后验概率**并画出“粗细=置信度”的图，阈值化得到稀疏网络。
 2. **密度估计**：要一个完整模型；常求 MAP 图，一般用启发式；**树结构**是少有的可**精确高效**求全局最优的特例（如 Chow-Liu）。
- **难点**：可能图数量随节点数**指数爆炸**（上界 $O(2^{V(V-1)/2})$ ），无法完全计算与存储整个后验。
- **替代/补充**：若只为密度估计，可考虑**潜变量模型**（更易学、预测更快），但**可解释性**差；也可与结构学习**结合**。
- **更高目标**：学**因果结构**以预测**干预效果**，比相关结构学习**更具挑战**。

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。

Since computing the MAP graph or the exact posterior edge marginals is in general computationally intractable (Chickering 1996), in this section we discuss some “quick and dirty” methods for learning graph structures which can be used to visualize one’s data. The resulting models do not constitute consistent joint probability distributions, so they cannot be used for prediction, and they cannot even be formally evaluated in terms of goodness of fit. Nevertheless, these methods are a useful ad hoc tool to have in one’s data visualization toolbox, in view of their speed and simplicity.

26.2.1 Relevance networks

A relevance network is a way of visualizing the pairwise mutual information between multiple random variables: we simply choose a threshold and draw an edge from node i to node j if $I(X_i; X_j)$ is above this threshold. In the Gaussian case, $I(X_i; X_j) = -\frac{1}{2} \log(1 - \rho_{ij}^2)$, where ρ_{ij} is the correlation coefficient (see Exercise 2.13), so we are essentially visualizing Σ ; this is known as the covariance graph (Section 19.4.4.1).

This method is quite popular in systems biology (Margolin et al. 2006), where it is used to visualize the interaction between genes. The trouble with biological examples is that they are hard for non-biologists to understand. So let us instead illustrate the idea using natural language text. Figure 26.1 gives an example, where we visualize the MI between words in the newsgroup dataset from Figure 1.2. The results seem intuitively reasonable.

However, relevance networks suffer from a major problem: the graphs are usually very dense, since most variables are dependent on most other variables, even after thresholding the MIs. For example, suppose X_1 directly influences X_2 which directly influences X_3 (e.g., these form components of a signalling cascade, $X_1 \rightarrow X_2 \rightarrow X_3$). Then X_1 has non-zero MI with X_3 (and vice versa), so there will be a $1 \rightarrow 3$ edge in the relevance network. Indeed, most pairs will be connected.

A better approach is to use graphical models, which represent conditional independence, rather than dependence. In the above example, X_1 is conditionally independent of X_3 given X_2 , so there will not be a $1 \rightarrow 3$ edge. Consequently graphical models are usually much sparser than relevance networks, and hence are a more useful way of visualizing interactions between multiple variables.

26.2.2 Dependency networks

A simple and efficient way to learn a graphical model structure is to independently fit D sparse full-conditional distributions $p(x_t|x - t)$; this is called a dependency network (Heckerman et al. 2000). The chosen variables constitute the inputs to the node, i.e., its Markov blanket. We can then visualize the resulting sparse graph. The advantage over relevance networks is that redundant variables will not be selected as inputs.

We can use any kind of sparse regression or classification method to fit each CPD. (Heckerman et al. 2000) uses classification/ regression trees, (Meinshausen and Bühlmann 2006) use l1regularized linear regression, (Wainwright et al. 2006) use l1-regularized logistic regression (see depnetFit for some code), (Dobra 2009) uses Bayesian variable selection, etc. (Meinshausen and Bühlmann 2006) discuss theoretical conditions under which ;1-regularized linear regression can recover the true graph structure, assuming the data was generated from a sparse Gaussian graphical model.

Figure 26.2 shows a dependency network that was learned from the 20-newsgroup data using l1 regularized logistic regression, where the penalty parameter λ was chosen by BIC. Many of the words present in these estimated Markov blankets represent fairly natural associations (aids:disease, baseball:fans, bible:god, bmw:car, cancer:patients, etc.). However, some of the estimated statistical dependencies seem less intuitive, such as baseball:windows and bmw:christian. We can gain more insight if we look not only at the sparsity pattern, but also the values of the regression weights. For example, here are the incoming weights for the first 5 words:

- aids: children (0.53), disease (0.84), fact (0.47), health (0.77), president (0.50), research (0.53)
- baseball: christian (-0.98), drive (-0.49), games (0.81), god (-0.46), government (-0.69), hit (0.62), memory (-1.29), players (1.16), season (0.31), software (-0.68), windows (-1.45)
- bible: car (-0.72), card (-0.88), christian (0.49), fact (0.21), god (1.01), jesus (0.68), orbit (0.83), program (-0.56), religion (0.24), version (0.49)
- bmw: car (0.60), christian (-11.54), engine (0.69), god (-0.74), government (-1.01), help (-0.50), windows (-1.43)
- cancer: disease (0.62), medicine (0.58), patients (0.90), research (0.49), studies (0.70)

Words in italic red have negative weights, which represents a dissociative relationship. For example, the model reflects that baseball:windows is an unlikely combination. It turns out that most of the weights are negative (1173 negative, 286 positive, 8541 zero) in this model.

In addition to visualizing the data, a dependency network can be used for inference. However, the only algorithm we can use is Gibbs sampling, where we repeatedly sample the nodes with missing values from their full conditionals. Unfortunately, a product of full conditionals does not, in general, constitute a representation of any valid joint distribution (Heckerman et al. 2000), so the output of the Gibbs sampler may not be meaningful. Nevertheless, the method can sometimes give reasonable results if there is not much missing data, and it is a useful method for data imputation (Gelman and Raghunathan 2001). In addition, the method can be used as an initialization technique for more complex structure learning methods that we discuss below.

已思考 8s

下面先给出完整、无遗漏的翻译，然后是通俗详尽的解释，最后是要点小结。

翻译

26.2 为知识发现而进行的结构学习

由于计算 MAP 图或精确的后验边缘概率在一般情况下都是计算上不可行的 (Chickering, 1996)，本节我们讨论一些“快而简” (quick and dirty) 的方法来学习图结构，这些方法可用于对数据进行可视化。由此得到的模型并不构成一致的联合概率分布，所以既不能用于预测，甚至也不能在拟合优度的意义上被正式评估。尽管如此，鉴于它们的速度和简单性，这些方法仍是数据可视化工具箱中有用的临时性 (ad hoc) 工具。

26.2.1 关联网络 (Relevance networks)

关联网络是一种可视化多个随机变量之间两两互信息的方法：我们只需选择一个阈值，当 $I(X_i; X_j)$ 高于该阈值时，就从结点 i 画一条边到结点 j 。在高斯情形下， $I(X_i; X_j) = -\frac{1}{2} \log(1 - \rho_{ij}^2)$ ，其中 ρ_{ij} 是相关系数 (见习题 2.13)，因此我们本质上是在可视化 Σ ；这被称为协方差图 (第 19.4.4.1 节)。

这种方法在系统生物学中相当流行 (Margolin 等, 2006)，用来可视化基因之间的相互作用。生物学的例子的问题在于，对非生物学家来说难以理解。因此，我们改用自然语言文本来说明这个想法。图 26.1 给出了一个例子：我们对图 1.2 中的新sgroup 数据集里的词之间的互信息进行可视化。结果看起来在直觉上是合理的。

然而，关联网络存在一个主要问题：图通常非常稠密，因为即使在对互信息进行阈值处理

之后，大多数变量仍然与大多数其他变量相依。举例来说，假设 X_1 直接影响 X_2 ，而 X_2 直接影响 X_3 （例如它们构成信号级联的组成部分， $X_1 - X_2 - X_3$ ）。那么 X_1 与 X_3 （反之亦然）具有非零互信息，因此在关联网络中会出现 1 – 3 的边。事实上，大多数成对都会被连接起来。

一个更好的方法是使用图模型，它表示的是**条件独立**而不是（无条件的）相依。在上述例子中，给定 X_2 时， X_1 与 X_3 条件独立，因此不会有 1 – 3 的边。于是，图模型通常比关联网络稀疏得多，因此在可视化多个变量之间的相互作用时更为有用。

26.2.2 依赖网络 (Dependency networks)

一种简单而高效地学习图模型结构的方法是**独立地**拟合 D 个稀疏的**全条件分布** $p(x_t | x_{-t})$ ；这称为**依赖网络** (Heckerman 等, 2000)。被选中的变量构成该结点的输入，也就是它的**马尔可夫毯**。然后我们可以对所得的稀疏图进行可视化。与关联网络相比，其优势在于**冗余变量**不会被选为输入。

为了拟合每个 CPD (条件概率分布)，我们可以使用任何类型的**稀疏回归或分类方法**。

(Heckerman 等, 2000) 使用**分类/回归树**，(Meinshausen 和 Bühlmann, 2006) 使用 l_1 正则化的线性回归，(Wainwright 等, 2006) 使用 l_1 正则化的逻辑回归（参见 `depnetFit` 获取一些代码），(Dobra, 2009) 使用**贝叶斯变量选择**，等等。

(Meinshausen 和 Bühlmann, 2006) 讨论了在何种理论条件下， l_1 正则化线性回归能够恢复真实的图结构，前提是假定数据来自一个**稀疏高斯图模型**。

图 26.2 显示了从 20-newsgroup 数据使用 l_1 正则化逻辑回归学习到的依赖网络，其中惩罚参数 λ 由 BIC 选择。许多出现在这些估计的马尔可夫毯中的词代表了相当自然的关联 (aids:disease, baseball:fans, bible:god, bmw:car, cancer:patients, 等等)。然而，一些估计出的统计依赖看起来不那么直观，比如 baseball:windows 和 bmw:christian。若不仅观察稀疏模式，还查看**回归权重的数值**，我们可以获得更多洞见。例如，以下是前 5 个词条的入边权重：

- **aids**: children (0.53), disease (0.84), fact (0.47), health (0.77), president (0.50), research (0.53)
- **baseball**: christian (-0.98), drive (-0.49), games (0.81), god (-0.46), government (-0.69), hit (0.62), memory (-1.29), players (1.16), season (0.31), software (-0.68), windows (-1.45)
- **bible**: car (-0.72), card (-0.88), christian (0.49), fact (0.21), god (1.01), jesus (0.68), orbit (0.83), program (-0.56), religion (0.24), version (0.49)
- **bmw**: car (0.60), christian (-11.54), engine (0.69), god (-0.74), government (-1.01), help (-0.50), windows (-1.43)
- **cancer**: disease (0.62), medicine (0.58), patients (0.90), research (0.49), studies (0.70)

斜体红色的词具有**负权重**，这表示一种**相互排斥** (dissociative) 关系。例如，模型反映出 baseball:windows 是一个不太可能的组合。事实证明，在该模型中大多数权重都是**负的** (1173 个负值，286 个正值，8541 个为零)。

除了用于数据可视化之外，依赖网络还可以用于**推断**。不过，我们唯一能用的算法是**Gibbs 采样**：反复从各结点的全条件分布中对缺失值的结点进行采样。不幸的是，一组全条件分

布的乘积一般并不构成任何有效的联合分布的表示 (Heckerman 等, 2000) , 因此 Gibbs 采样器的输出可能没有意义。尽管如此, 如果缺失数据不多, 该方法有时会给出合理结果, 并且它是一个有用的数据插补方法 (Gelman 和 Raghunathan, 2001) 。此外, 该方法还可以作为我们下面将要讨论的更复杂结构学习方法的初始化技术。

通俗详解

1) 本节想解决什么问题?

- 我们只想快速画出“变量之间怎么连线”来做可视化/探索, 而不是构建一个严格可用于预测的完整概率模型。
 - 因为严格的方法 (求 MAP 图或精确边缘概率) 太难算, 所以退而求其次, 用一些近似但很快的方法。
-

2) 关联网络 (relevance network) 到底做了啥?

- 先计算每一对变量之间的互信息 $I(X_i; X_j)$ (衡量“知道一个对另一个的信息增益”)。
 - 选择阈值: 大于阈值就画边。
 - 在高斯数据里, 互信息和相关系数 ρ 有公式: $I = -\frac{1}{2} \log(1 - \rho^2)$ 。所以它其实是在反映协方差矩阵 Σ 的结构 (哪对变量相关)。
 - **优点:** 简单、直观、很快; 常用于生物网络或文本词关系的快速可视化。
 - **致命缺点:** 太稠密。
 - 例: 链式影响 $X_1 \rightarrow X_2 \rightarrow X_3$ 。虽然 X_1 不直接连 X_3 , 但它俩仍有正的互信息 (通过 X_2 传递), 所以也会被连边。
 - 结果: 几乎对对都连上, 图太乱, 不利于解读。
-

3) 为什么图模型更适合可视化? (条件独立让图更稀疏)

- 图模型强调的是**条件独立**: 如果给定中间变量 X_2 后, X_1 和 X_3 就互不影响 (统计上独立), 那么在图里不连边。
 - 这会去掉传递性的“假连线”, 因此图往往更稀疏, 更容易读懂, 更能体现**直接关系**。
-

4) 依赖网络 (dependency network) 是什么? 为什么快?

- 思路: 对每个变量 X_t , 单独拟合它对“其他所有变量 X_{-t} ”的**条件分布** $p(x_t | x_{-t})$ 。
- 实现上, 就像给每个变量做一次预测模型, 输入是其余变量, 输出是该变量。
- 为了让图稀疏, 用**稀疏学习** (比如 l_1 正则) :

- 把系数压到 0，只留下少数真正有用的“邻居”——这组被选中的输入变量就构成该结点的**马尔可夫毯**。
- 可选工具箱：
 - **分类/回归树** (Heckerman 等, 2000)
 - l_1 **线性回归** (Meinshausen & Bühlmann, 2006)
 - l_1 **逻辑回归** (Wainwright 等, 2006; 教材里提到 *depnetFit* 代码)
 - **贝叶斯变量选择** (Dobra, 2009)
 - 这些方法都在不同假设下推动稀疏选择，有些还有理论保证（如：在**稀疏高斯图模型**成立时， l_1 线性回归可恢复真结构的条件）。
- λ (正则强度) 如何选？例子中用 **BIC** (在“拟合好”与“够简单”之间折中)。

读权重怎么读？

- **正权重**：条件在其他变量后，两词/两变量**同现的倾向更高** (正关联)。
- **负权重**：**互相排斥** (出现一个往往意味着另一个不出现)。例：baseball/windows 权重负，说明它们一起出现不常见。
- 案例里统计：绝大多数权重是**负的** (1173 负、286 正、8541 为 0)。这常见于“多类别词汇彼此竞争出现”的文本场景。

相比关联网络的优势

- 依赖网络通过**特征选择/稀疏化**避免了许多**冗余变量**被连上，图更干净、更具可解释性。
-

5) 用依赖网络做推断：可行但要小心

- 想填补缺失值或推断未观测结点，可用**Gibbs 采样**：
 1. 轮流对每个缺失结点，从已学得的 $p(x_t | x_{-t})$ 里抽样；
 2. 重复多轮，得到样本。
 - **风险**：把每个结点的“条件分布”简单相乘，通常**不对应**任何真正的**联合分布** (称“条件不一致”)。
 - 直观讲：你分别学到的一组条件分布，彼此可能**对不上**，因此 Gibbs 采样收敛到的“分布”可能**没意义**。
 - **实践经验**：
 - 如果**缺失很少**，当作**插补**工具往往还**挺好用** (Gelman & Raghunathan, 2001)。
 - 还能作为更复杂结构学习算法的**初始化** (先给个不错的稀疏结构起点，再精修)。
-

要点小结

- **目标**: 为知识发现/可视化快速学结构, 而非严谨预测。
- **关联网络**: 基于两两互信息阈值连边; **简单快但易稠密** (传递相依导致“到处连”)。
- **图模型视角**: 表达**条件独立**, 天然更稀疏、更能体现直接联系。
- **依赖网络**: 为每个变量单独拟合稀疏的 $p(x_t | x_{-t})$, 选出的输入即**马尔可夫毯**; 常用 l_1 或树、贝叶斯选择; λ 可用 **BIC** 选。
- **权重解读**: 正=同现倾向, 负=相互排斥, 0=无关; 文本示例里多为**负权重**。
- **推断注意**: 这套条件分布**未必对应合法联合分布**; Gibbs 采样结果可能**不可靠**。但**少量缺失的数据插补**、以及作为**复杂方法初始化**, 都很实用。

这一节主要在讲什么? 我不太懂

已思考 5s

没关系~这节其实就讲一件事: **当你只是想“看懂数据里谁跟谁有关系”时, 有一些又快又粗的画图方法**, 它们能帮你把变量之间的关系画成网络图, 方便探索; 但这些方法**不适合做严肃预测**。

可以把它分成两招:

1. 关联网络 (relevance networks)

- 做法: 看每一对变量“黏不黏”(用两两互信息衡量), 超过阈值就连线。
- 直觉: 谁常一起变化, 就把谁连上。
- 问题: 容易**太密**。哪怕 A 只通过 B 影响到 C, A 和 C 也会被连上, 最后几乎“满天是线”, 不利读图。

2. 依赖网络 (dependency networks)

- 做法: 对**每个变量单独训练一个“小模型”**, 用“其他所有变量”来预测它(选用稀疏方法, 让大多数不重要的连线变成零)。被留下来的那些“真有用的”变量, 就是它的**马尔可夫毯**, 把这些连起来就得到了较**稀疏**的图。
- 优点: 比关联网络更能**去掉冗余**, 图更干净、可解释性更好。
- 但要注意: 把这些“小模型”的条件分布拼在一起, **不一定构成一个真正合法的整体概率模型**。所以它更像是**可视化/填补少量缺失值**的工具, **不是拿来做严肃推断/预测的最终模型**。

小比喻:

- 关联网络像是“看谁老在同一个群里说话就拉条线”, 结果朋友的朋友也被你连上, 圈子特别密。

- 依赖网络像是“对每个人，只挑真正影响 TA 发言的人连线”（用稀疏方法筛选），图就清爽多了。但这套连线规则拼成的“全场规则书”并不一定自洽，所以别把它当成完整定律。

什么时候用？

- 想**快速探索和画图**、找“可能的直接关系” → 先试依赖网络。
- 想**非常粗略看整体黏性** → 关联网络也行，但要有“会很密”的心理预期。
- 想**做预测/生成** → 这两种都不够，要用真正的一体化概率模型（或改用潜变量模型等）。

一个常见误区

- 用依赖网络做 Gibbs 采样“推断”。书里提醒：因为整体不一定合法，采出来的结果可能没意义；只在**缺失很少、做数据插补或当更复杂方法的初始化**时还算实用。
-

要点小结

- 本节核心：**为“知识发现”提供快而简的画图方法**，不是为预测。
- **关联网络**：看两两关联就连线→快，但**很稠密**。
- **依赖网络**：给每个变量做稀疏的条件模型→图更**稀疏**、**可解释**；但**不保证**是合法的整体概率模型。
- 可用于**可视化、少量缺失值插补、复杂方法的初始化**；不适合当最终预测模型。

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。

26.3 Learning tree structures

For the rest of this chapter, we focus on learning fully specified joint probability models, which can be used for density estimation, prediction and knowledge discovery. Since the problem of structure learning for general graphs is NP-hard (Chickering 1996), we start by considering the special case of trees. Trees are special because we can learn their structure efficiently, as we discuss below, and because, once we have learned the tree, we can use them for efficient exact inference, as discussed in Section 20.2.

26.3.1 Directed or undirected tree?

Before continuing, we need to discuss the issue of whether we should use directed or undirected trees. A directed tree, with a single root node r , defines a joint

distribution as follows:

$$p(x|T) = t \in V p(x_t|x_{pa(t)}) \quad (26.1)$$

where we define $pa(r) = \emptyset$. For example, in Figure 26.3(b-c), we have

$$p(x_1, x_2, x_3, x_4|T) = p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_2) \quad (26.2)$$

$$= p(x_2)p(x_1|x_2)p(x_3|x_2)p(x_4|x_2) \quad (26.3)$$

We see that the choice of root does not matter: both of these models are equivalent.

To make the model more symmetric, it is preferable to use an undirected tree. This can be represented as follows:

$$p(x|T) = t \in V p(x_t) \quad (s, t) \in E \quad p(x_s, x_t) \quad p(x_s)p(x_t) \quad (26.4)$$

where $p(x_s, x_t)$ is an edge marginal and $p(x_t)$ is a node marginal. For example, in Figure 26.3(a) we have

$$p(x_1, x_2, x_3, x_4|T) = p(x_1)p(x_2)p(x_3)p(x_4)$$

$$p(x_1, x_2)p(x_2, x_3)p(x_2, x_4) \quad p(x_1)p(x_2)p(x_2)p(x_3)p(x_2)p(x_4)$$

$$(26.5)$$

To see the equivalence with the directed representation,

let us cancel terms to get

$$p(x_1, x_2, x_3, x_4|T) = p(x_1, x_2) \quad p(x_2, x_3) \quad p(x_2) \quad p(x_2, x_4) \quad p(x_2)$$

$$(26.6)$$

$$= p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_2) \quad (26.7)$$

$$= p(x_2)p(x_1|x_2)p(x_3|x_2)p(x_4|x_2) \quad (26.8)$$

where $p(x_t|x_s) = p(x_s, x_t)/p(x_s)$.

Thus a tree can be represented as either an undirected or directed graph: the number of parameters is the same, and hence the complexity of learning is the same. And of course, inference is the same in both representations, too.

The undirected representation, which is symmetric, is useful for structure learning, but the directed representation is more convenient for parameter learning.

26.3.2 Chow-Liu algorithm for finding the ML tree structure

Using Equation 26.4, we can write the log-likelihood for a tree as follows:

$$\log p(D|\theta, T) = \sum_{t \in V} N_{tk} \log p(x_t = k|\theta) + \sum_{s,t} N_{stjk} \log p(x_s = j, x_t = k|\theta) \quad (26.9)$$

where N_{stjk} is the number of times node s is in state j and node t is in state k , and N_{tk} is the number of times node t is in state k . We can rewrite these counts in terms of the empirical distribution: $N_{stjk} = N_{st} p_{st}(x_s = j, x_t = k)$ and $N_{tk} = N_t p_{tk}(x_t = k)$. Setting θ to the MLEs, this becomes

$$\log p(D|\theta, T) = \sum_{t \in V} k p_{tk}(x_t = k) \log p_{tk}(x_t = k) \quad (26.10)$$

$$+ \sum_{(s,t) \in E(T)} I(x_s, x_t | \theta_{st}) \quad (26.11)$$

where $I(x_s, x_t | \theta_{st}) \geq 0$ is the mutual information between

xs and xt given the empirical distribution:

$$I(xs, xt | \theta_{st}) = \sum_j \sum_k p_{emp}(xs=j, xt=k) \log p_{emp}(xs=j, xt=k) / p_{emp}(xs=j)p_{emp}(xt=k) \quad (26.12)$$

Since the first term in Equation 26.11 is independent of the topology T , we can ignore it when learning structure. Thus the tree topology that maximizes the likelihood can be found by computing the maximum weight spanning tree, where the edge weights are the pairwise mutual informations, $I(ys, yt | \theta_{st})$. This is called the Chow-Liu algorithm (Chow and Liu 1968).

There are several algorithms for finding a max spanning tree (MST). The two best known are Prim's algorithm and Kruskal's algorithm. Both can be implemented to run in $O(E \log V)$ time, where $E = V^2$ is the number of edges and V is the number of nodes. See e.g., (Sedgewick and Wayne 2011, 4.3) for details. Thus the overall running time is $O(NV^2 + V^2 \log V)$, where the first term is the cost of computing the sufficient statistics.

Figure 26.4 gives an example of the method in action, applied to the binary 20 newsgroups data shown in Figure 1.2. The tree has been arbitrarily rooted at the node representing "email". The connections that are learned seem intuitively reasonable.

26.3.3 Finding the MAP forest

Since all trees have the same number of parameters, we can safely use the maximum likelihood score as a model selection criterion without worrying about overfitting. However, sometimes we may want to fit a forest rather than a single tree, since inference in a forest is much faster than in a tree (we can run belief propagation in each tree in the forest in parallel). The MLE criterion will never choose to omit an edge. However, if we use the marginal likelihood or a penalized likelihood (such as BIC), the optimal solution may be a forest. Below we give the details for the marginal likelihood case.

In Section 26.4.2.2, we explain how to compute the marginal likelihood of any DAG using a Dirichlet prior for the CPTs. The resulting expression can be written as follows:

$$\log p(D|T) = \sum_t \sum_i N_{t,i} \log \left(\frac{p(x_{it}|x_i, pa(t)|\theta_t)}{\sum_{\theta_t} p(\theta_t)} \right) \quad (26.13)$$

where $N_{t,i}$ are the counts (sufficient statistics) for node t and its parents, and p is defined in Equation 26.28.

Now suppose we only allow DAGs with at most one

parent. Following (Heckerman et al. 1995, p227), let us associate a weight with each $s \rightarrow t$ edge, $w_{s,t} = \text{score}(t|s) - \text{score}(t|0)$, where $\text{score}(t|0)$ is the score when t has no parents. Note that the weights might be negative (unlike the MLE case, where edge weights are always non-negative because they correspond to mutual information).

Then we can rewrite the objective as follows:

$$\log p(D|T) = \sum_t w_{pa(t)}(t) + \text{score}(t|0) \quad (26.14)$$

The last term is the same for all trees T , so we can ignore it. Thus finding the most probable tree amounts to finding a maximal branching in the corresponding weighted directed graph. This can be found using the algorithm in (Gabow et al. 1984).

If the scoring function is prior and likelihood equivalent (these terms are explained in Section 26.4.2.3), we have $\text{score}(s|t) + \text{score}(t|0) = \text{score}(t|s) + \text{score}(s|0)$ (26.15)

and hence the weight matrix is symmetric. In this case, the maximal branching is the same as the maximal weight forest. We can apply a slightly modified version of the MST algorithm to find this (Edwards et al. 2010). To see this, let $G = (V, E)$ be a graph with both positive and negative edge weights. Now let G' be a graph obtained by omitting all the negative edges from G . This cannot reduce the total weight, so we can find the maximum weight forest of G by finding the MST for each connected component of G' . We can do this by running Kruskal's algorithm directly on G' : there is no need to find the connected components explicitly.

26.3.4 Mixtures of trees

A single tree is rather limited in its expressive power. Later in this chapter we discuss ways to learn more general graphs. However, the resulting graphs can be expensive to do inference in. An interesting alternative is to learn a mixture of trees (Meila and Jordan 2000), where each mixture component may have a different tree topology. This is like an unsupervised version of the TAN classifier discussed in Section 10.2.1. We can fit a mixture of trees by using EM: in the E step, we compute the responsibilities of each cluster for each data point, and in the M step, we use a weighted version of the Chow-Liu algorithm. See (Meila and Jordan 2000) for details.

In fact, it is possible to create an “infinite mixture of trees”, by integrating out over all possible trees. Remarkably, this can be done in V^3 time using the matrix tree theorem.

This allows us to perform exact Bayesian inference of posterior edge marginals etc. However, it is not tractable to use this infinite mixture for inference of hidden nodes. See (Meila and Jaakkola 2006) for details.

已思考 9s

下面先给出**完整、逐句且无遗漏的翻译**，接着用**通俗、细致的解释**帮你吃透要点，最后再做**要点小结**。

翻译

26.3 学习树结构

在本章的剩余部分里，我们聚焦于学习**完全指定**的联合概率模型，它们可用于**密度估计、预测以及知识发现**。

由于一般图的结构学习问题是 NP-hard 的 (Chickering 1996)，我们先从**树 (tree)** 这个**特殊情形开始**。树之所以特殊，是因为它们的结构可以高效地学到 (如下所述)；而且一旦学到了树，我们就可以像第 20.2 节讨论的那样，用它们来进行**高效且精确的推断**。

26.3.1 用有向树还是无向树？

继续之前，我们需要讨论到底应该使用**有向**还是**无向**的树。一个具有单一根结点 r 的有向树，定义的联合分布如下：

$$p(x \mid T) = \prod_{t \in V} p(x_t \mid x_{\text{pa}(t)}) \quad (26.1)$$

其中我们定义 $\text{pa}(r) = \emptyset$ 。例如，在图 26.3(b–c) 中，我们有

$$p(x_1, x_2, x_3, x_4 \mid T) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2)p(x_4 \mid x_2) \quad (26.2)$$

$$= p(x_2)p(x_1 \mid x_2)p(x_3 \mid x_2)p(x_4 \mid x_2) \quad (26.3)$$

我们看到**根的选择并不重要**：这两个模型是等价的。

为了让模型更加对称，最好使用**无向树**。它可以如下表示：

$$p(x \mid T) = \prod_{t \in V} p(x_t) \prod_{(s,t) \in E} \frac{p(x_s, x_t)}{p(x_s)p(x_t)} \quad (26.4)$$

其中 $p(x_s, x_t)$ 是**边缘 (edge marginal)**， $p(x_t)$ 是**结点边缘 (node marginal)**。例如，在图 26.3(a) 中，我们有

$$p(x_1, x_2, x_3, x_4 \mid T) = \frac{p(x_1)p(x_2)p(x_3)p(x_4)p(x_1, x_2)p(x_2, x_3)p(x_2, x_4)}{p(x_1)p(x_2)p(x_2)p(x_3)p(x_2)p(x_4)} \quad (26.5)$$

为了看出与有向表示的等价性，让我们约掉项得到

$$p(x_1, x_2, x_3, x_4 \mid T) = \frac{p(x_1, x_2)}{p(x_2)} \frac{p(x_2, x_3)}{p(x_2)} \frac{p(x_2, x_4)}{p(x_2)} \quad (26.6)$$

$$= p(x_1) p(x_2 \mid x_1) p(x_3 \mid x_2) p(x_4 \mid x_2) \quad (26.7)$$

$$= p(x_2) p(x_1 \mid x_2) p(x_3 \mid x_2) p(x_4 \mid x_2) \quad (26.8)$$

其中 $p(x_t \mid x_s) = p(x_s, x_t)/p(x_s)$ 。

因此，一个树既可以用无向图也可以用有向图来表示：参数个数相同，因此学习的复杂度也相同。当然，在这两种表示中，推断也是相同的。无向表示更对称，对结构学习有用；而有向表示在参数学习上更方便。

26.3.2 寻找极大似然 (ML) 树结构的 Chow–Liu 算法

利用式 (26.4)，我们可以把一棵树的对数似然写成：

$$\log p(D \mid \theta, T) = \sum_t \sum_k N_{tk} \log p(x_t = k \mid \theta) + \sum_{(s,t)} \sum_{j,k} N_{stjk} \log \frac{p(x_s = j, x_t = k \mid \theta)}{p(x_s = j \mid \theta) p(x_t = k \mid \theta)} \quad (26.9)$$

其中 N_{stjk} 是结点 s 取状态 j 、结点 t 取状态 k 的次数， N_{tk} 是结点 t 取状态 k 的次数。我们可用经验分布改写这些计数： $N_{stjk} = N p_{\text{emp}}(x_s = j, x_t = k)$ ， $N_{tk} = N p_{\text{emp}}(x_t = k)$ 。将 θ 设为 MLE，得到

$$\frac{1}{N} \log p(D \mid \theta, T) = \sum_{t \in V} \sum_k p_{\text{emp}}(x_t = k) \log p_{\text{emp}}(x_t = k) \quad (26.10)$$

$$+ \sum_{(s,t) \in E(T)} I(x_s, x_t \mid \hat{\theta}_{st}) \quad (26.11)$$

其中

$$I(x_s, x_t \mid \hat{\theta}_{st}) = \sum_j \sum_k p_{\text{emp}}(x_s = j, x_t = k) \log \frac{p_{\text{emp}}(x_s = j, x_t = k)}{p_{\text{emp}}(x_s = j) p_{\text{emp}}(x_t = k)} \quad (26.12)$$

是基于经验分布的两两互信息。

由于式 (26.11) 中的第一项与拓扑 T 无关，学习结构时可忽略它。因此，最大化似然的树拓扑可以通过最大权重生成树来找到，其中边权就是两两互信息 $I(x_s, x_t \mid \hat{\theta}_{st})$ 。这称为 Chow–Liu 算法 (Chow and Liu 1968)。

寻找最大生成树 (MST) 有多种算法。最著名的两个是 Prim 算法和 Kruskal 算法。二者都可以实现为 $O(E \log V)$ 的时间复杂度，其中 $E = V^2$ 是边的条数、 V 是结点数 (细节见 Sedgewick and Wayne 2011, §4.3)。因此总体运行时间为 $O(NV^2 + V^2 \log V)$ ，其中第一项是计算充分统计量的代价。

图 26.4 展示了该方法在二元的 20 newsgroups 数据 (见图 1.2) 上的一个示例。树被任意地以代表“email”的结点为根。学到的连接看上去在直觉上是合理的。

26.3.3 寻找 MAP 森林

由于所有树的参数个数相同，我们可以放心地使用极大似然分数作为模型选择准则，而无需担心过拟合。然而，有时我们可能希望拟合一片森林而不是一棵树，因为在森林中进行推断要比在树中快得多（我们可以在森林中的每棵树上并行运行信念传播）。MLE 准则永远不会选择删除一条边。然而，如果我们使用边缘似然或带惩罚的似然（例如 BIC），最优解可能是一片森林。下面我们给出边缘似然情形的细节。

在第 26.4.2.2 节中，我们解释了如何在 CPT 采用 Dirichlet 先验的情况下计算任意 DAG 的边缘似然。得到的表达式可写为：

$$\log p(D \mid T) = \sum_{t \in V} \log \int_{\theta_t=1}^N [\prod_{i=1}^N p(x_{it} \mid x_{i,\text{pa}(t)}, \theta_t)] p(\theta_t) d\theta_t = \sum_t \text{score}(N_{t,\text{pa}(t)}) \quad (26.13)$$

其中 $N_{t,\text{pa}(t)}$ 是结点 t 及其父结点的计数（充分统计量），score 的定义见式 (26.28)。

现在假设我们只允许每个结点最多一个父结点的 DAG。按照 (Heckerman 等, 1995, 第 227 页)，给每条 $s \rightarrow t$ 的边关联一个权重

$$w_{s,t} = \text{score}(t \mid s) - \text{score}(t \mid 0),$$

其中 $\text{score}(t \mid 0)$ 是当 t 没有父结点时的分数。注意这些权重可能是负的（不同于 MLE 情形，那时边权总是非负的，因为它们对应互信息）。于是我们可以把目标函数改写为

$$\log p(D \mid T) = \sum_t \text{score}(t \mid \text{pa}(t)) = \sum_t w_{\text{pa}(t),t} + \sum_t \text{score}(t \mid 0) \quad (26.14)$$

最后一项对所有树 T 都相同，因此可以忽略。于是，寻找最可能的树就等同于在相应的带权有向图中寻找一个最大分支 (maximal branching)。可用 (Gabow 等, 1984) 中的算法来求解。

如果评分函数是先验等价且似然等价的（这两个术语见第 26.4.2.3 节），那么有

$$\text{score}(s \mid t) + \text{score}(t \mid 0) = \text{score}(t \mid s) + \text{score}(s \mid 0) \quad (26.15)$$

因此权重矩阵是对称的。在这种情况下，最大分支与最大权重森林是相同的。我们可以用稍作修改的 MST 算法找到它 (Edwards 等, 2010)。为说明这一点，设 $G = (V, E)$ 是一个既有正权也有负权的图。令 G' 为从 G 中删去所有负权边后得到的图。这样做不会降低总权重，因此我们可以通过对 G' 每个连通分量求 MST 来找到 G 的最大权重森林。我们可以直接在 G' 上运行 Kruskal 算法：无需显式找到各连通分量。

26.3.4 树的混合 (mixtures of trees)

单棵树的表达能力相当有限。我们将在本章后面讨论学习更一般的图的方法。不过，那些图在做推断时可能代价很高。一个有趣的替代方案是学习一族树的混合 (Meila and Jordan 2000)，其中每个混合分量可以有不同的树拓扑。这就像第 10.2.1 节讨论的 TAN 分类器的无监督版本。我们可以用 EM 来拟合树混合：在 E 步，计算每个数据点对每个簇的责任度 (responsibility)；在 M 步，使用加权版本的 Chow-Liu 算法。细节见 (Meila and

Jordan 2000)。

事实上，通过对**所有可能的树**进行积分，可以构造一个“**无限树混合**”。令人惊讶的是，这可以利用**矩阵树定理**在 V^3 时间内完成。这使我们能够对**后验边缘（如边的后验概率）等**进行精确的贝叶斯推断。然而，用这个无限混合来对**隐藏结点**进行推断并不可行。细节见 (Meilla and Jaakkola 2006)。

通俗详解

下面把关键思想“翻译成人话”，尽量少用公式：

1) 为什么从“树”开始？

- 一般的图结构学习很难 (NP-hard)。
- 树是“最简单的有用图”：
 - 学结构有现成的高效算法；
 - 做推断也很快 (信念传播在树上是精确且线性的)。

2) 有向树 vs 无向树：其实是一回事（在树里）

- **有向树**：选个根，把每条边看成“父 → 子”，联合分布就是“每个结点在给定父结点下的条件概率”的乘积。根怎么选都行，模型等价。
- **无向树**：更“对称”。它把联合分布写成“每个点的边缘概率 × 每条边的校正项（成对边缘 / 结点边缘的乘积）”。
- 两种写法在树里能**互相还原**：参数个数一样、学习与推断的难度也一样。
 - 直觉：树里**没有环**，所以只要知道每个点的单边缘和每条边的双边缘，就能把整张分布“拼”回来。

3) Chow-Liu：学“最像数据”的树

- 想找一棵**最可能生成你数据的树**（极大似然，ML）。
- 关键结论：**最大化似然 ≡ 选一棵“总互信息”最大的生成树**
 - 把每一对变量的“亲密度”定为**互信息 (MI)**；
 - 把所有变量当作结点，把 MI 当作**边权**；
 - 求一棵**最大权重生成树 (MST)** → 这棵就是 ML 树。
- 算法层面：
 - 互信息要先算出来（数据规模 N ，变量数 V ，两两组合是 V^2 级别，因此耗时约 $O(NV^2)$ ）。
 - MST 用 Prim 或 Kruskal，在稠密图上约 $O(V^2 \log V)$ 。
 - 总体： $O(NV^2 + V^2 \log V)$ 。
- 好处：**快、简单、可解释**，而且学到的树上做推断也快。

4) 不止是树，还可以是“森林”（MAP 意义下）

- 所有树参数数目都一样，所以用 MLE 做模型选择时不担心过拟合（不会偏向更复杂的树）。
- 但有时我们想要森林（几棵树互不相连）：
 - 这样推断更快（可以并行、每棵小树各算各的）。
 - 可是纯 MLE 从不删边，它会尽量连满成一棵树。
 - 如果改用边缘似然或带惩罚的似然（BIC），最优解可能会删边，得到森林。
- 技巧（允许每个结点最多一个父亲）：
 - 给每条可能的有向边 $s \rightarrow t$ 设一个分数增益 $w_{s,t} = \text{score}(t|s) - \text{score}(t|0)$ 。
 - 目标就变成：选一组边构成“每点 ≤ 1 父”的结构，使得总增益最大。
 - 这等于在带权有向图里找最大分支（maximum branching / arborescence），可用 Gabow 算法。
 - 如果评分满足先验等价 + 似然等价，权重会~~对称~~，问题退化为最大权重森林：
 - 直接把负权边去掉，在剩下的图上跑 Kruskal（会自然得到每个连通块的 MST），就得到最优森林（无需先找连通分量）。

直观理解“删负边”：负权表示“加这条边反而拉低分数”，那就不如别加；在只剩正边的图里，Kruskal 会尽量加分高的边，同时避免形成环，于是自然停在一片“最肥”的森林上。

5) 树不够用？那就“树的混合”

- 单棵树太“直线”，很多真实关系抓不住。
- 方案 A：树的混合（mixture of trees）
 - 每个成分是一棵树，但不同成分的树拓扑可以不同；
 - 用 EM 学：
 - E 步：算每条数据属于各成分的“责任度”；
 - M 步：对每个成分，按责任度做加权版 Chow-Liu。
- 方案 B：“无限树混合”（把所有可能的树积分掉）
 - 借助矩阵树定理，一些后验边概率这类量能在 $O(V^3)$ 时间精确算到；
 - 适合做贝叶斯边缘推断（比如“这条边存在的后验概率是多少？”）；
 - 但要用它来推断隐藏结点，计算就不可行了。

要点小结

- 树是结构学习的友好起点：可高效学习结构、也可高效精确推断。

- **有向树与无向树等价**（在树里）：参数数量、学习/推断复杂度都一样；无向形式对**结构学习**友好，有向形式对**参数学习**方便。
- Chow-Liu 算法（ML 树）：
 - 边权 = 两两互信息；
 - 选最大权重生成树；
 - 复杂度约 $O(NV^2 + V^2 \log V)$ 。
- MAP 森林（基于边缘似然/BIC）：
 - MLE 不会删边；
 - 用边缘似然给边打分，允许负权 → 变成找**最大分支/最大权重森林**；
 - 评分若“先验等价+似然等价”，权重对称：**删负边后跑 Kruskal** 即得最优森林。
- 提升表达力：
 - **树的混合**：EM 学习，M 步用加权 Chow-Liu；
 - **无限树混合**：用矩阵树定理在 V^3 时间做精确的边后验等贝叶斯推断，但**不适合**隐藏结点推断。

如果你只想一句话记住：

“学树 = 算互信息 → 挑最大生成树（Chow-Liu）；想更简更快用森林（删负边跑 Kruskal）；想更强就用树的混合或用矩阵树定理做贝叶斯边缘推断。”

这一节主要在讲什么？我不太懂

已思考 7s

不打紧，这一节的核心就三件事：**用“树”来学概率模型的结构、怎么学、以及什么时候需要比一棵树更灵活。**

一句话版

- **为什么学“树”**：一般图太难（NP-hard），而树既能高效学到，又能高效做精确推断。
- **怎么学树（Chow-Liu）**：算每对变量的互信息当作边的“亲密度”，然后在所有点上选一棵**最大权重生成树**（Prim/Kruskal 就行）——这棵树就是**极大似然**的结构。
- **如果一棵树不够**：
 - 用带惩罚/贝叶斯评分会**删掉弱边**，得到几棵小树组成的**森林**（推断更快）；
 - 或者用**树的混合**（每个簇一棵树，EM 学习）；
 - 甚至可以对所有树积分（矩阵树定理， $O(V^3)$ ）来做**边的后验概率**等贝叶斯量。

再展开半步（尽量不费脑）

- **有向树 vs 无向树**: 在树里是等价的; 无向写法更对称、便于结构学习, 有向写法便于参数学习; 根选谁都不影响模型。
- **Chow-Liu 算法** (学到极大似然的树) :
 1. 计算所有变量对的互信息 (谁更“捆绑”就权重大) ;
 2. 以互信息做边权, 求最大生成树;
 3. 得到的就是最像数据的那棵树, 随后在这棵树上做推断既快又精确。
- **MAP 森林** (想更简更快) :
 - 纯 MLE 不会删边;
 - 换成边缘似然/BIC这类带惩罚评分后, 部分边会变“负收益”, 就不连, 自然得到多棵树;
 - 技术上等价于在加了权的图里找最大分支/最大权森林 (负边丢掉, Kruskal 直接跑) 。
- **树的混合 & 无限树混合**:
 - **树的混合**: 数据可能有多个模式, 每个模式一棵树, 用 EM 学;
 - **无限树混合**: 对所有树做贝叶斯积分 (用矩阵树定理), 能精确算“这条边出现的后验概率”, 但不适合拿来推断隐藏结点。

小比喻

把变量当城市、互信息当“交通流量”。Chow-Liu就是选一张**不成环但总流量最大的路网**(生成树)。如果某些路是“负收益”(惩罚后不值得修),就不修,留下**几片独立小路网**(森林)。数据里有多种出行模式?就**多张路网**(树的混合)。

要点小结

- 一般图难, 树友好: **学得快、推断快**。
- **有向/无向树等价**: 无向利于学结构, 有向利于学参数。
- Chow-Liu: 边权=互信息 → **最大生成树** = ML 树。
- 想更简更快: 用**边缘似然/BIC**得到**MAP 森林** (删掉“负边”)。
- 表达力不够: 上**树的混合**; 想要边的后验置信度: 用**矩阵树定理**做**贝叶斯边缘推断**。

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。

26.4 Learning DAG structures

In this section, we discuss how to compute (functions of) $p(G|D)$, where G is constrained to be a DAG. This is often

called Bayesian network structure learning. In this section, we assume there is no missing data, and that there are no hidden variables. This is called the complete data assumption. For simplicity, we will focus on the case where all the variables are categorical and all the CPDs are tables, although the results generalize to real-valued data and other kinds of CPDs, such as linear-Gaussian CPDs. Our presentation is based in part on (Heckerman et al. 1995), although we will follow the notation of Section 10.4.2. In particular, let $x_{it} \in \{1, \dots, K_t\}$ be the value of node t in case i , where K_t is the number of states for node t . Let $\theta_{tck} = p(x_{it} = k | x_{pa(t)} = c)$, for $k = 1: K_t$, and $c = 1: C_t$, where C_t is the number of parent combinations (possible conditioning cases). For notational simplicity, we will often assume $K_t = K$, so all nodes have the same number of states. We will also let $d_t = \text{dim}(pa(t))$ be the degree or fan-in of node t , so that $C_t = K^{d_t}$.

26.4.1 Markov equivalence

In this section, we discuss some fundamental limits to our ability to learn DAG structures from data.

Consider the following 3 DGMs: $X \rightarrow Y \rightarrow Z$, $X \leftarrow Y \leftarrow Z$ and $X \leftarrow Y \rightarrow Z$. These all represent the same set of CI statements, namely

$$X \perp Z | Y, X \perp Z \quad (26.16)$$

We say these graphs are Markov equivalent, since they encode the same set of CI assumptions. That is, they all belong to the same Markov equivalence class. However, the v-structure $X \rightarrow Y \leftarrow Z$ encodes $X \perp Z$ and $X \perp Z | Y$, which represents the opposite set of CI assumptions.

One can prove the following theorem.

Theorem 26.4.1 (Verma and Pearl (Verma and Pearl 1990)).

Two structures are Markov equivalent iff they have the same undirected skeleton and the same set of v-structures. For example, referring to Figure 26.5, we see that $G_1 \equiv G_2$, since reversing the $2 \rightarrow 4$ arc creates a new v-structure.

However, $G_1 \not\equiv G_3$, since reversing the $1 \rightarrow 5$ arc does not create a new v-structure.

We can represent a Markov equivalence class using a single partially directed acyclic graph (PDAG), also called an essential graph or pattern, in which some edges are directed and some undirected. The undirected edges represent reversible edges; any combination is possible so long as no new v-structures are created. The directed edges are called compelled edges, since changing their orientation would change the v-structures and hence

change the equivalence class. For example, the PDAG $X-Y-Z$ represents $\{X \rightarrow Y \rightarrow Z, X \leftarrow Y \leftarrow Z, X \leftarrow Y \rightarrow Z\}$ which encodes $X \perp Z$ and $X \perp Z|Y$. See Figure 26.6.

The significance of the above theorem is that, when we learn the DAG structure from data, we will not be able to uniquely identify all of the edge directions, even given an infinite amount of data. We say that we can learn DAG structure “up to Markov equivalence”. This also cautions us not to read too much into the meaning of particular edge orientations, since we can often change them without changing the model in any observable way.

26.4.2 Exact structural inference

In this section, we discuss how to compute the exact posterior over graphs, $p(G|D)$, ignoring for now the issue of computational tractability.

26.4.2.1 Deriving the likelihood

Assuming there is no missing data, and that all CPDs are tabular, the likelihood can be written as follows:

$$p(D|G, \theta) = \prod_{i=1}^N \prod_{t=1}^T \text{Cat}(x_{it}|x_i, \theta_t) \quad (26.17)$$

$$= \prod_{i=1}^N \prod_{t=1}^T \prod_{c=1}^{C_i} \text{Cat}(x_{it}|x_i, \theta_t) I(x_{it}=c) \quad (26.18)$$

$$= \prod_{i=1}^N \prod_{t=1}^T \prod_{c=1}^{C_i} \prod_{k=1}^{K_i} \theta_k I(x_{it}=k, x_i, \theta_t=c) \quad (26.19)$$

$$= \prod_{t=1}^T \prod_{c=1}^{C_t} \prod_{k=1}^{K_t} \theta_k N_{tk} \quad (26.20)$$

where N_{tk} is the number of times node t is in state k and its parents are in state c . (Technically these counts depend on the graph structure G , but we drop this from the notation.)

26.4.2.2 Deriving the marginal likelihood

Of course, choosing the graph with the maximum likelihood will always pick a fully connected graph (subject to the acyclicity constraint), since this maximizes the number of parameters. To avoid such overfitting, we will choose the graph with the maximum marginal likelihood, $p(D|G)$; the magic of the Bayesian Occam’s razor will then penalize overly complex graphs.

To compute the marginal likelihood, we need to specify priors on the parameters. We will make two standard assumptions. First, we assume global prior parameter independence, which means

$$p(\theta) = \prod_{t=1}^T p(\theta_t) \quad (26.21)$$

Second, we assume local prior parameter independence, which means

$$p(\theta_t) = \prod_{c=1}^{C_t} p(\theta_{tc}) \quad (26.22)$$

for each t . It turns out that these assumptions imply that the prior for each row of each CPT must be a Dirichlet

(Geiger and Heckerman 1997), that is,

$$p(\theta_{tc}) = \text{Dir}(\theta_{tc} | \alpha_{tc}) \quad (26.23)$$

Given these assumptions, and using the results of Section 5.3.2.2, we can write down the marginal likelihood of any DAG as follows:

$$\begin{aligned} p(D|G) &= \prod_{t=1}^T \prod_{c=1}^{C_t} \prod_{i: x_i(t)=c} \text{Cat}(x_i(t) | \theta_{tc}) \\ &\quad \prod_{c=1}^{C_t} \text{Dir}(\theta_{tc}) d\theta_{tc} \quad (26.24) \end{aligned}$$

$$= \prod_{t=1}^T \prod_{c=1}^{C_t} B(N_{tc} + \alpha_{tc}) B(\alpha_{tc}) \quad (26.25)$$

$$= \prod_{t=1}^T \prod_{c=1}^{C_t} \Gamma(N_{tc}) \Gamma(N_{tc} + \alpha_{tc}) \prod_{k=1}^{C_t} \Gamma(N_{tck} + \alpha_{tck}) \Gamma(\alpha_{tck}) \quad (26.26)$$

$$= \prod_{t=1}^T \text{score}(N_t, \alpha_{pa(t)}) \quad (26.27)$$

where $N_{tc} = k N_{tck}$, $\alpha_{tc} = k \alpha_{tck}$, $N_t, \alpha_{pa(t)}$ is the vector of counts (sufficient statistics) for node t and its parents, and $\text{score}()$ is a local scoring function defined by

$$\text{score}(N_t, \alpha_{pa(t)}) = \prod_{c=1}^{C_t} B(N_{tc} + \alpha_{tc}) B(\alpha_{tc}) \quad (26.28)$$

We say that the marginal likelihood decomposes or factorizes according to the graph structure.

26.4.2.3 Setting the prior

How should we set the hyper-parameters α_{tck} ? It is tempting to use a Jeffreys prior of the form $\alpha_{tck} = 1/2$ (Equation 5.62). However, it turns out that this violates a property called likelihood equivalence, which is sometimes considered desirable. This property says that if G_1 and G_2 are Markov equivalent (Section 26.4.1), they should have the same marginal likelihood, since they are essentially equivalent models. Geiger and Heckerman (1997) proved that, for complete graphs, the only prior that satisfies likelihood equivalence and parameter independence is the Dirichlet prior, where the pseudo counts have the form

$$\alpha_{tck} = \alpha p_0(x_t = k, x_{pa(t)} = c) \quad (26.29)$$

where $\alpha > 0$ is called the equivalent sample size, and p_0 is some prior joint probability distribution. This is called the BDe prior, which stands for Bayesian Dirichlet likelihood equivalent.

To derive the hyper-parameters for other graph structures, Geiger and Heckerman (1997) invoked an additional assumption called parameter modularity, which says that if node X_t has the same parents in G_1 and G_2 , then $p(\theta_t | G_1) = p(\theta_t | G_2)$. With this assumption, we can always derive α_t for a node t in any other graph by marginalizing the pseudo counts in Equation 26.29.

Typically the prior distribution p_0 is assumed to be uniform over all possible joint configurations. In this case, we have

$$\alpha tck = \alpha KtCt \quad (26.30)$$

since $p_0(x_t = k, x_{pa}(t) = c) = 1 / KtCt$. Thus if we sum the pseudo counts over all $Ct \times Kt$ entries in the CPT, we get a total equivalent sample size of α . This is called the BDeu prior, where the “u” stands for uniform. This is the most widely used prior for learning Bayes net structures. For advice on setting the global tuning parameter α , see (Silander et al. 2007).

26.4.2.4 Simple worked example

We now give a very simple worked example from (Neapolitan 2003, p.438). Suppose we have just 2 binary nodes, and the following 8 data cases:

X1 X2 11 12 11 22 11 21 11 22

Suppose we are interested in two possible graphs: G1 is $X1 \rightarrow X2$ and G2 is the disconnected graph. The empirical counts for node 1 in G1 are $N1 = (5, 3)$ and for node 2 are $X2 = 1 \ X2 = 2 \ X1 = 1 \ 41 \ X1 = 2 \ 12$

The BDeu prior for G1 is $\alpha 1 = (\alpha/2, \alpha/2)$, $\alpha 2|x_1=1 = (\alpha/4, \alpha/4)$ and $\alpha 2|x_1=2 = (\alpha/4, \alpha/4)$. For G2, the prior for $\theta 1$ is the same, and for $\theta 2$ it is $\alpha 2|x_1=1 = (\alpha/2, \alpha/2)$ and $\alpha 2|x_1=2 = (\alpha/2, \alpha/2)$. If we set $\alpha = 4$, and use the BDeu prior, we find $p(D|G1) = 7.2150 \times 10^{-6}$ and $p(D|G2) = 6.7465 \times 10^{-6}$. Hence the posterior probabilities, under a uniform graph prior, are $p(G1|D) = 0.51678$ and $p(G2|D) = 0.48322$.

26.4.2.5 Example: analysis of the college plans dataset

We now consider a more interesting example from (Heckerman et al. 1997). Consider the data set collected in 1968 by Sewell and Shah which measured 5 variables that might influence the decision of high school students about whether to attend college. Specifically, the variables are as follows:

- Sex Male or female
- SES Socio economic status: low, lower middle, upper middle or high.
- IQ Intelligence quotient: discretized into low, lower middle, upper middle or high.
- PE Parental encouragment: low or high
- CP College plans: yes or no.

These variables were measured for 10,318 Wisconsin high school seniors. There are $2 \times 4 \times 4 \times 2 = 128$ possible joint configurations.

Heckerman et al. computed the exact posterior over all 29,281 possible 5 node DAGs, except for ones in which SEX and/or SES have parents, and/or CP have children. (The prior probability of these graphs was set to 0, based

on domain knowledge.) They used the BDeu score with $\alpha = 5$, although they said that the results were robust to any α in the range 3 to 40. The top two graphs are shown in Figure 26.7. We see that the most probable one has approximately all of the probability mass, so the posterior is extremely peaked.

It is tempting to interpret this graph in terms of causality (see Section 26.6). In particular, it seems that socio-economic status, IQ and parental encouragement all causally influence the decision about whether to go to college, which makes sense. Also, sex influences college plans only indirectly through parental encouragement, which also makes sense. However, the direct link from socio economic status to IQ seems surprising; this may be due to a hidden common cause. In Section 26.5.1.4 we will re-examine this dataset allowing for the presence of hidden variables.

26.4.2.6 The K2 algorithm

Suppose we know a total ordering of the nodes. Then we can compute the distribution over parents for each node independently, without the risk of introducing any directed cycles: we simply enumerate over all possible subsets of ancestors and compute their marginal likelihoods.¹ If we just return the best set of parents for each node, we get the the K2 algorithm (Cooper and Herskovits 1992).

26.4.2.7 Handling non-tabular CPDs

If all CPDs are linear Gaussian, we can replace the Dirichlet-multinomial model with the normalgamma model, and thus derive a different exact expression for the marginal likelihood. See (Geiger and Heckerman 1994) for the details. In fact, we can easily combine discrete nodes and Gaussian nodes, as long as the discrete nodes always have discrete parents; this is called a conditional Gaussian DAG. Again, we can compute the marginal likelihood in closed form. See (Bottcher and Dethlefsen 2003) for the details.

In the general case (i.e., everything except Gaussians and CPTs), we need to approximate the marginal likelihood. The simplest approach is to use the BIC approximation, which has the form

$$t \log p(D_t | \theta_t) - K t C_t / 2 \log N \quad (26.31)$$

26.4.3 Scaling up to larger graphs

The main challenge in computing the posterior over DAGs is that there are so many possible graphs. More precisely,

(Robinson 1973) showed that the number of DAGs on D nodes satisfies the following recurrence:

$$f(D) = \sum_{i=1}^D (-1)^{i+1} D \binom{2i}{D-i} f(D-i) \quad (26.32)$$

for $D > 2$. The base case is $f(1) = 1$. Solving this recurrence yields the following sequence: 1, 3, 25, 543, 29281, 3781503, etc. In view of the enormous size of the hypothesis space, we are generally forced to use approximate methods, some of which we review below.

26.4.3.1 Approximating the mode of the posterior

We can use dynamic programming to find the globally optimal MAP DAG (up to Markov equivalence) (Koivisto and Sood 2004; Silander and Myllmaki 2006).

Unfortunately this method takes $V2V$ time and space, making it intractable beyond about 16 nodes. Indeed, the general problem of finding the globally optimal MAP DAG is provably NP-complete (Chickering 1996),

Consequently, we must settle for finding a locally optimal MAP DAG. The most common method is greedy hill climbing: at each step, the algorithm proposes small changes to the current graph, such as adding, deleting or reversing a single edge; it then moves to the neighboring graph which most increases the posterior. The method stops when it reaches a local maximum. It is important that the method only proposes local changes to the graph, since this enables the change in marginal likelihood (and hence the posterior) to be computed in constant time (assuming we cache the sufficient statistics). This is because all but one or two of the terms in Equation 26.25 will cancel out when computing the log Bayes factor $\delta(G \rightarrow G') = \log p(G') - \log p(G|D)$.

We can initialize the search from the best tree, which can be found using exact methods discussed in Section 26.3. For speed, we can restrict the search so it only adds edges which are part of the Markov blankets estimated from a dependency network (Schmidt 2010). Figure 26.8 gives an example of a DAG learned in this way from the 20-newsgroup data.

We can use techniques such as multiple random restarts to increase the chance of finding a good local maximum. We can also use more sophisticated local search methods, such as genetic algorithms or simulated annealing, for structure learning.

26.4.3.2 Approximating other functions of the posterior

If our goal is knowledge discovery, the MAP DAG can be misleading, for reasons we discussed in Section 5.2.1. A

better approach is to compute the probability that each edge is present, $p(G_{st} = 1 | D)$, of the probability there is a path from s to t . We can do this exactly using dynamic programming (Koivisto 2006; Parviainen and Koivisto 2011). Unfortunately these methods take V2V time in the general case, making them intractable for graphs with more than about 16 nodes.

An approximate method is to sample DAGs from the posterior, and then to compute the fraction of times there is an $s \rightarrow t$ edge or path for each (s, t) pair. The standard way to draw samples is to use the Metropolis Hastings algorithm (Section 24.3), where we use the same local proposal as we did in greedy search (Madigan and Raftery 1994).

A faster-mixing method is to use a collapsed MH sampler, as suggested in (Friedman and Koller 2003). This exploits the fact that, if a total ordering of the nodes is known, we can select the parents for each node independently, without worrying about cycles, as discussed in Section 26.4.2.6. By summing over all possible choice of parents, we can marginalize out this part of the problem, and just sample total orders. (Ellis and Wong 2008) also use order-space (collapsed) MCMC, but this time with a parallel tempering MCMC algorithm.

已思考 16s

下面先给出**完整、逐句且无遗漏的翻译**，然后用**通俗详尽的解释**带你吃透，最后做**要点小结**。

翻译

26.4 学习有向无环图 (DAG) 结构

在本节中，我们讨论如何计算 $p(G | D)$ （或其函数），其中 G 被约束为一个 DAG。这通常被称为**贝叶斯网络结构学习**。在本节里，我们假设**没有缺失数据**，而且**没有隐藏变量**。这被称为**完全数据假设**。为简单起见，我们将聚焦于**所有变量都是类别型且所有条件概率分布 (CPD) 都是表格形式**的情形，尽管这些结果也可以推广到实数型数据和其他类型的 CPD（例如**线性-高斯** CPD）。

我们的表述部分基于 (Heckerman 等, 1995)，但我们会遵循第 10.4.2 节的记号。具体地，让 $x_{it} \in \{1, \dots, K_t\}$ 表示样本 i 中结点 t 的取值，其中 K_t 是结点 t 的状态数。令 $\theta_{tck} = p(x_t = k | x_{\text{pa}(t)} = c)$ ，其中 $k = 1:K_t$, $c = 1:C_t$ ，且 C_t 是**父结点组合**（可能的条

件情形) 的个数。为简化记号, 我们常假设 $K_t = K$, 因此所有结点的状态数相同。我们还令 $d_t = \dim(\text{pa}(t))$ 表示结点 t 的入度/扇入数, 于是 $C_t = K^{d_t}$ 。

26.4.1 马尔可夫等价 (Markov equivalence)

本节讨论从数据中学习 DAG 结构能力的根本限度。

考虑以下 3 个有向图模型 (DGM) : $X \rightarrow Y \rightarrow Z$ 、 $X \leftarrow Y \leftarrow Z$ 与 $X \leftarrow Y \rightarrow Z$ 。

它们都表示相同的一组条件独立 (CI) 陈述, 即

$$X \perp Z \mid Y, \quad X \perp\!\!\!\perp Z \quad (26.16)$$

我们说这些图是**马尔可夫等价的**, 因为它们编码了相同的 CI 假设集合。也就是说, 它们都属于同一个**马尔可夫等价类**。然而, **v 结构** $X \rightarrow Y \leftarrow Z$ 则编码了 $X \perp Z$ 和 $X \perp\!\!\!\perp Z \mid Y$, 这恰好代表**相反**的一组 CI 假设。

可以证明如下定理。

定理 26.4.1 (Verma 与 Pearl (1990)) : 当且仅当两个结构具有**相同的无向骨架 (skeleton)** 以及**相同的 v 结构集合**时, 它们是马尔可夫等价的。

例如, 参见图 26.5, 我们看到 $G_1 \equiv G_2$, 因为把 $2 \rightarrow 4$ 的弧反向会产生一个新的 v 结构。然而, $G_1 \not\equiv G_3$, 因为把 $1 \rightarrow 5$ 的弧反向**不会**产生新的 v 结构。

我们可以用一个**部分有向无环图 (PDAG)** 来表示一个**马尔可夫等价类**, 也称为**本质图 (essential graph)** 或**模式 (pattern)**, 其中有些边是有向的, 有些是无向的。**无向边**代表可逆的边; 只要不创造新的 v 结构, 任意方向的组合都是可能的。**有向边**称为**被迫边 (compelled edges)**, 因为改变其方向会改变 v 结构, 因而改变等价类。例如, PDAG $X-Y-Z$ 表示集合 $\{X \rightarrow Y \rightarrow Z, X \leftarrow Y \leftarrow Z, X \leftarrow Y \rightarrow Z\}$, 其编码为 $X \perp\!\!\!\perp Z$ 和 $X \perp Z \mid Y$ (见图 26.6)。

上述定理的重要意义在于: 当我们从数据学习 DAG 结构时, 即使有**无限多数据**, 我们也无法**唯一确定所有边的方向**。我们说只能“**到马尔可夫等价为止**”地学习 DAG 结构。这也提醒我们, 不要过度解读某条边的具体方向, 因为我们常常可以改变方向而不会在可观测层面改变模型。

26.4.2 精确的结构推断

本节讨论如何计算精确的图的后验 $p(G \mid D)$, 暂且不考虑计算可行性问题。

26.4.2.1 似然的推导

在假设**无缺失**且所有 CPD 都是**表格**的情况下, 似然可写成

$$p(D \mid G, \theta) = \prod_{i=1}^N \prod_{t=1}^V \text{Cat}(x_{it} \mid x_{i,\text{pa}(t)}, \theta_t) \quad (26.17)$$

$$= \prod_{i=1}^N \prod_{t=1}^V \prod_{c=1}^{C_t} \text{Cat}(x_{it} \mid \theta_{tc}) \mathbf{I}(x_{i,\text{pa}(t)} = c) \quad (26.18)$$

$$= \prod_{i=1}^N \prod_{t=1}^V \prod_{c=1}^{C_t} \prod_{k=1}^{K_t} \theta_{tc k}^{\mathbf{I}(x_{i,t} = k, x_{i,\text{pa}(t)} = c)} \quad (26.19)$$

$$= \prod_{t=1}^V \prod_{c=1}^{C_t} \prod_{k=1}^{K_t} \theta_{tc k}^{N_{tc k}} \quad (26.20)$$

其中 $N_{tc k}$ 是“结点 t 取状态 k 且其父组合为 c ”的出现次数。 (严格来说，这些计数依赖于图结构 G ，但在记号上我们略去。)

26.4.2.2 边缘似然的推导

当然，选择**极大似然**的图会总是挑一个**尽可能稠密**的图（在无环约束下），因为那样能最大化参数数量。为避免这种**过拟合**，我们将选择**最大边缘似然**的图 $p(D \mid G)$ ；此时**贝叶斯的奥卡姆剃刀**会自动惩罚过于复杂的图。

要计算边缘似然，需要对参数指定先验。我们做两个标准假设。首先，**全局参数先验独立**：

$$p(\theta) = \prod_{t=1}^V p(\theta_t) \quad (26.21)$$

其次，**局部参数先验独立**：

$$p(\theta_t) = \prod_{c=1}^{C_t} p(\theta_{tc}) \quad (26.22)$$

对每个 t 都成立。事实证明，这些假设意味着每个 CPT 的每一行的先验必须是一个 Dirichlet (Geiger 与 Heckerman, 1997)，即

$$p(\theta_{tc}) = \text{Dir}(\theta_{tc} \mid \alpha_{tc}) \quad (26.23)$$

在这些假设下，并利用第 5.3.2.2 节的结果，我们可以把任意 DAG 的边缘似然写为

$$p(D \mid G) = \prod_{t=1}^V \prod_{c=1}^{C_t} \left[\int \prod_{i: x_{i,\text{pa}(t)} = c} \text{Cat}(x_{it} \mid \theta_{tc}) \right] \text{Dir}(\theta_{tc}) d\theta_{tc} \quad (26.24)$$

$$= \prod_{t=1}^V \prod_{c=1}^{C_t} \frac{B(N_{tc} + \alpha_{tc})}{B(\alpha_{tc})} \quad (26.25)$$

$$= \prod_{t=1}^V \prod_{c=1}^{C_t} \frac{\Gamma(N_{tc})}{\Gamma(N_{tc} + \alpha_{tc})} \prod_{k=1}^{K_t} \frac{\Gamma(N_{tck} + \alpha_{tck}^G)}{\Gamma(\alpha_{ijk}^G)} \quad (26.26)$$

$$= \prod_{t=1}^V \text{score}(N_{t,\text{pa}(t)}) \quad (26.27)$$

其中 $N_{tc} = \sum_k N_{tck}$, $\alpha_{tc} = \sum_k \alpha_{tck}$, $N_{t,\text{pa}(t)}$ 是结点 t 及其父结点的计数（充分统计量）向量，而 $\text{score}(\cdot)$ 是如下定义的**局部评分函数**：

$$\text{score}(N_{t,\text{pa}(t)}) = \prod_{c=1}^{C_t} \frac{B(N_{tc} + \alpha_{tc})}{B(\alpha_{tc})} \quad (26.28)$$

我们说边缘似然按照图结构分解/因子化。

26.4.2.3 先验的设定

我们该如何设定超参数 α_{tck} ? 使用 Jeffreys 先验 $\alpha_{tck} = \frac{1}{2}$ (式 5.62) 看上去很诱人。然而，这会违反一个有时很重要的性质：**似然等价** (likelihood equivalence)。该性质说：如果 G_1 与 G_2 马尔可夫等价 (见 26.4.1 节)，那么它们的**边缘似然应相同**，因为它们本质上是等价模型。Geiger 与 Heckerman (1997) 证明，对于完全图，唯一同时满足**似然等价与参数独立**的先验是 Dirichlet 先验，且伪计数具有如下形式

$$\alpha_{tck} = \alpha p_0(x_t = k, x_{\text{pa}(t)} = c) \quad (26.29)$$

其中 $\alpha > 0$ 称为**等效样本量** (equivalent sample size)， p_0 是某个**先验联合分布**。这称为**BDe 先验** (Bayesian Dirichlet, 满足似然等价)。

为了给其他图结构推导超参数，Geiger 与 Heckerman (1997) 又引入了一个假设：**参数模块性** (parameter modularity)，其含义是：如果结点 X_t 在 G_1 与 G_2 中有相同的父结点，那么 $p(\theta_t | G_1) = p(\theta_t | G_2)$ 。在该假设下，我们总能通过对式 (26.29) 中的伪计数做边缘化来为任意图中的任意结点 t 推导 α_t 。

通常先验分布 p_0 假定在所有可能的联合配置上**均匀**。此时

$$\alpha_{tck} = \frac{\alpha}{K_t C_t} \quad (26.30)$$

因为 $p_0(x_t = k, x_{\text{pa}(t)} = c) = \frac{1}{K_t C_t}$ 。因此，若把 CPT 中所有 $C_t \times K_t$ 个条目的伪计数加总，可得到**总等效样本量**为 α 。这称为**BDeu 先验** ("u" 表示 uniform)。这是学习贝叶斯网结构时**最常用的**先验。关于全局调参 α 的设置建议，见 (Silander 等, 2007)。

26.4.2.4 一个简单的算例

我们给出 (Neapolitan 2003, 第 438 页) 中的一个非常简单的例子。假设我们只有 2 个二值结点，以及如下 8 条数据：

| X_1 | X_2 |
|-------|-------|
| 1 | 1 |
| 1 | 2 |
| 1 | 1 |
| 2 | 2 |
| 1 | 1 |
| 2 | 1 |
| 1 | 1 |
| 2 | 2 |

假设我们只关心两个可能的图： G_1 为 $X_1 \rightarrow X_2$, G_2 为断开图。在 G_1 下，结点 1 的经验计数为 $N_1 = (5, 3)$, 结点 2 的列联表为

| | | $X_2 = 1$ | $X_2 = 2$ |
|-----------|---|-----------|-----------|
| | | 4 | 1 |
| $X_1 = 1$ | 1 | 2 | |
| $X_1 = 2$ | | | |

G_1 的 BDeu 先验为 $\alpha_1 = (\alpha/2, \alpha/2)$, $\alpha_{2|x_1=1} = (\alpha/4, \alpha/4)$, $\alpha_{2|x_1=2} = (\alpha/4, \alpha/4)$ 。对 G_2 , θ_1 的先验相同, 而 θ_2 的先验为 $\alpha_{2|x_1=1} = (\alpha/2, \alpha/2)$ 与 $\alpha_{2|x_1=2} = (\alpha/2, \alpha/2)$ 。若我们取 $\alpha = 4$, 并使用 BDeu 先验, 可得

$$p(D | G_1) = 7.2150 \times 10^{-6}, p(D | G_2) = 6.7465 \times 10^{-6}.$$

因此在均匀的图先验下, 后验为 $p(G_1 | D) = 0.51678$, $p(G_2 | D) = 0.48322$ 。

26.4.2.5 例子：college plans 数据集的分析

我们来看 (Heckerman 等, 1997) 中的一个更有趣的例子。Sewell 与 Shah 在 1968 年收集了一个数据集, 测量了 5 个可能影响高中生是否上大学决策的变量。具体变量如下:

- Sex 性别: 男或女
- SES 社会经济地位: 低、次中、中上或高
- IQ 智商: 离散化为低、次中、中上或高
- PE 父母鼓励: 低或高
- CP 上大学计划: 是或否

这些变量测量于威斯康星州 10,318 名高中毕业生。共有 $2 \times 4 \times 4 \times 2 \times 2 = 128$ 种可能的联合配置。

Heckerman 等对所有 29,281 个可能的 5 结点 DAG (排除那些 SEX 和/或 SES 有父结点, 以及/或者 CP 有子结点的图) 精确计算了后验 (基于领域知识将此类图的先验概率设为 0)。他们使用 $\alpha = 5$ 的 BDeu 评分, 不过他们指出结果对 α 在 3 到 40 的范围内都很稳健。前两个概率最高的图见图 26.7。我们看到最可能的那个图几乎占据了全部概率质量, 因此后验极度尖峰。

人们很容易想把这个图解释为因果关系 (见 26.6 节)。尤其看起来社会经济地位、智商和父母鼓励都会因果地影响是否上大学的决策——这很合理。同时, 性别仅通过父母鼓励间

接影响上大学计划，这也合理。然而，从社会经济地位到智商的直接连边看起来令人惊讶；这可能是由于一个隐藏的共同原因。在 26.5.1.4 节，我们将在**允许隐藏变量存在的假设**下重新审视该数据集。

26.4.2.6 K2 算法

假设我们已知结点的一个**全序**。则我们可以**相互独立地**为每个结点计算其父结点的分布，而不用担心引入任何有向环：我们只需枚举所有可能的**祖先子集**并计算其**边缘似然**。¹若我们仅返回每个结点的**最佳父集**，就得到**K2 算法** (Cooper 与 Herskovits, 1992)。

26.4.2.7 处理非表格 CPD

若所有 CPD 都是**线性-高斯的**，我们可以用**正态-伽马 (normal-gamma) 模型替换 Dirichlet-多项式模型**，从而导出另一种精确的边缘似然表达式。详见 (Geiger 与 Heckerman, 1994)。事实上，只要**离散结点的父结点都是离散的**，我们可以容易地把**离散结点与高斯结点结合**，这称为**条件高斯 DAG**。同样可得到闭式的边缘似然。详见 (Bottcher 与 Dethlefsen, 2003)。

在**一般情形** (即除高斯与 CPT 外的所有情况)，我们需要**近似边缘似然**。最简单的方法是使用**BIC 近似**，形式为

$$\sum_t \log p(D_t | \hat{\theta}_t) - \frac{K_t C_t}{2} \log N \quad (26.31)$$

26.4.3 扩展到更大的图

计算 DAG 的后验的主要挑战在于**可能图的数量太多**。更确切地，Robinson (1973) 证明了 D 个结点上的 DAG 数量满足如下递推：

$$f(D) = \sum_{i=1}^D (-1)^{i+1} \binom{D}{i} 2^{i(D-i)} f(D-i), \quad D > 2 \quad (26.32)$$

基例为 $f(1) = 1$ 。解该递推得到序列：1, 3, 25, 543, 29281, 3781503, ...²。鉴于假设空间**巨大**，我们通常被迫使用**近似方法**；下面回顾其中一些。

26.4.3.1 近似后验众数 (MAP)

我们可以用**动态规划**来寻找**全局最优的 MAP DAG** (到马尔可夫等价为止) (Koivisto 与 Sood 2004; Silander 与 Myllymäki 2006)。不幸的是，该方法需要约 $V^2 2^V$ 级别的**时间与空间**，在约 16 个结点以上就不可行。事实上，寻找全局最优 MAP DAG 的一般问题是**NP 完全的** (Chickering 1996)。

因此，我们只能满足于**局部最优**的 MAP DAG。最常见的方法是**贪心爬山**：每一步对当前图提出一个小改动 (加/删/反向一条边)，然后移动到使后验**增幅最大的**邻居图；达到局部最大时停止。重要的是，方法**只提出局部修改**，这使得**边缘似然的变化** (从而后验的变

化) 可在**常数时间**内计算 (假设我们缓存了充分统计量)。这是因为在计算对数 Bayes 因子

$$\delta(G \rightarrow G') = \log p(G' | D) - \log p(G | D)$$

时, 式 (26.25) 中的除了一**两项**外其他项都会相互抵消。

我们可以从**最优的树结构** (第 26.3 节) 初始化搜索。为加速, 我们可以**限制搜索**只添加那些属于**依赖网络估计的马尔可夫毯**的边 (Schmidt 2010)。图 26.8 给出了用这种方法从 20-newsgroup 数据学到的一个 DAG。

我们可以用**多次随机重启**来提高找到好局部极值的机会。也可以用更复杂的**局部搜索**, 比如**遗传算法**或**模拟退火**, 来做结构学习。

26.4.3.2 近似后验的其他函数

若目标是**知识发现**, MAP DAG 可能会**误导**, 原因见第 5.2.1 节。更好的做法是计算**每条边出现的概率** $p(G_{st} = 1 | D)$, 或“从 s 到 t 存在路径”的概率。我们可以用**动态规划**精确计算 (Koivisto 2006; Parviainen 与 Koivisto 2011)。不幸的是, 一般情形下这些方法需要约 V^{22V} 时间, 对超过约 16 个结点的图就不可行。

一种近似方法是**从后验中抽样 DAG**, 然后对每个 (s, t) 对, 统计出现 $s \rightarrow t$ 边或路径的**频率**。标准的采样方式是使用 Metropolis–Hastings (MH) (第 24.3 节), 并使用与贪心搜索相同的**局部提议分布** (Madigan 与 Raftery 1994)。

一种**混合更快**的方法是使用**折叠 (collapsed) MH 采样器** (Friedman 与 Koller 2003)。其利用了这样一个事实: 如果我们知道结点的一个**全序**, 就可以在**不担心成环**的情况下**独立地**选择每个结点的父结点 (见 26.4.2.6)。通过**对所有父集求和**, 我们可以把这部分问题**边缘化**, 只需要**对全序**进行采样。(Ellis 与 Wong 2008) 也使用了**序空间 (order-space)**的**折叠 MCMC**, 但采用的是**并行退火**的 MCMC 算法。

注: 文中的¹、² 是脚注标记, 原文未在摘录中给出脚注内容。

通俗详解 (尽量不用高数)

这节在干嘛?

- 我们要学**贝叶斯网 (DAG) **的结构: 给定数据 D , 评估不同图 G 的好坏 (比如比较 $p(G | D)$ 、 $p(D | G)$), 并挑出好的图或从后验中进行统计总结 (如“这条边出现的概率多大?”)。

1) 马尔可夫等价: 方向并非都能识别

- 有些边的方向**凭数据看不出来** (即使无限数据也不行), 因为**多个 DAG 在可观测层面表达同样的独立关系**。

- 判定标准：同骨架 + 同 v 结构 \Rightarrow 同一个等价类。
- 影响：学习结果通常只能确定到等价类，别过度解读某条边的朝向。

2) 评分（边缘似然）怎么来？为什么用 BDe/BDeu？

- 直接用极大似然会偏爱更稠密的图（过拟合）。
- 换成边缘似然 $p(D | G)$ ：把参数积分掉，复杂图会被“奥卡姆剃刀”惩罚。
- 在“全局/局部参数先验独立”的标准假设下，每个 CPT 的每一行用 Dirichlet 先验，得到可分解的闭式公式（式 26.24–26.28）。
- 想让等价的 DAG 拿到相同分数（似然等价），伪计数应设为 $\alpha \times p_0$ （局部联合）
(BDe)；若 p_0 取均匀，就是 BDeu。 α 是“等效样本量”，直观上控制“先验力度”。

3) 小例子

- 2 个二值变量、8 条数据：算出 $p(D | G_1)$ 与 $p(D | G_2)$ （在 $\alpha = 4$ 的 BDeu 下）非常接近，但稍微偏向有边的 G_1 。这展示了边缘似然如何“折中”拟合度与复杂度。

4) 实战例子（上大学数据）

- 5 个变量，先用领域知识排除不合理的边，再用 BDeu ($\alpha = 5$) 算精确后验。
- 结果的后验很尖：几乎集中在一个图上（强信号）。
- 直观因果解读看似合理，但也要小心：例如 SES \rightarrow IQ 可能是因为有隐藏共同原因；下一节（26.5）会引入隐藏变量重新分析。

5) K2：已知结点顺序时的便捷法

- 如果变量的因果/时间顺序已知，对每个结点只需在其祖先子集中择优选父，彼此独立地计算边缘似然，不会成环。
- “每点选最优父集”就是 K2 算法（简单、快）。

6) 非表格 CPD 怎么办？

- 线性-高斯：有精确边缘似然（正态-伽马先验）。
- 条件高斯 DAG（离散父 + 高斯子）也能闭式。
- 其他更一般的模型：用 BIC 近似（“数据拟合”–“复杂度惩罚”）。

7) 大图的挑战与近似

- D 个结点的 DAG 数量爆炸（递推式 26.32），16 个结点左右就把精确法逼疯。
- 找 MAP 图：
 - 精确 DP 法复杂度约 $n^2 2^n$ （16 个点左右就顶不住）。
 - 实务上用贪心爬山（加/删/反向一条边），因局部改动使得 Bayes 因子可常数时间更新；可配多次重启、遗传、退火。
 - 可从最优树（Chow-Liu）启动，并用依赖网络估的马尔可夫链限边加速。
- 算边的后验概率或“存在路径”的概率：
 - 精确 DP 同样 $n^2 2^n$ 量级，16 个点以上就不行。

- 近似：从后验采样 DAG (MH)；
- 更快：折叠 MH → 在序空间采样（只采结点全序，把父集边缘化），甚至配并行退火。

要点小结

- **学习对象：**有向无环图（贝叶斯网）结构；目标是评估/选择 $p(G | D)$ 或其函数（如边存在的后验概率）。
- **马尔可夫等价：**同骨架 + 同 v 结构 \Rightarrow 同等价类；**边方向未必可识别**，结论通常“到等价类为止”。
- **评分（边缘似然）：**在 Dirichlet 行先验下得到**可分解的闭式**；BDe/BDeu保证（或近似）**似然等价**， α 为**等效样本量**。
- **案例：**小算例与“上大学”数据展示了边缘似然如何平衡拟合与复杂度，以及领域知识/隐藏变量的重要性。
- K2：已知全序时，逐点在祖先子集中选父，简单高效。
- **非表格 CPD：**线性-高斯与条件高斯可闭式；一般模型用 BIC 近似。
- **可扩展性：**DAG 数量爆炸；精确法约止步于 ~16 结点。实务靠**贪心局部搜索 + 多重重启/退火/遗传**；
后验函数可用 MCMC 近似，**折叠序空间采样**常更高效。

一句话记忆：“**等价类限制可识别性；用 BDe(u) 做可分解打分；小图可精确，大图靠近似；爬山找 MAP、MCMC 估边概率，已知顺序用 K2，非表格用 BIC/高斯闭式。**”

这一节主要在讲什么？我不太懂

已思考 7s

没事～这节（26.4）说的其实就是：**怎么从数据里学一张“贝叶斯网（DAG）”的结构，以及在现实里应该怎么打分、怎么算、算不到时怎么近似。**

一句话总览

- 目标：给定数据 D ，给每个候选图 G 评估“好不好”（如 $p(G | D)$ 或 $p(D | G)$ ），从而学到一张合适的**有向无环图**。
- 假设：**完全数据**（无缺失、无隐藏变量）、变量先按**离散表格 CPD**讲，后面再说高斯/混合型。

关键点（不烧脑版）

1. 方向不是都能学出来：马尔可夫等价

多个 DAG 可能表达**同样的独立关系**（同骨架+同 v 结构）。所以**即便有无限数据**，很多边的方向也**不可唯一确定**——我们最多学到“**等价类**”。别过度解读单条边朝向。

2. 怎么给图打分：用边缘似然 (BDe/BDeu)

- 直接用极大似然会偏爱**更稠密的图**（过拟合）。
- 把参数积分掉得**边缘似然** $p(D | G)$ ；在“参数独立”的假设下，每个 CPT 的**每一行**用 Dirichlet 先验，有**闭式、可分解**的打分公式。
- 想让**等价的 DAG**打分一样（“**似然等价**”），用 BDe；若先验联合取**均匀**就是 BDeu。超参 α =“**等效样本量**”，调先验力度。

3. 一个小算例 + 一个真实例子

- 2 个二值变量的玩具例子：BDeu 下两张图分数接近，稍偏向有边那张。
- “上大学”数据：用领域约束 + BDeu 算**精确后验**，结果非常尖（几乎一张图称王）。直觉像因果，但要当心**隐藏共同原因**的可能。

4. 已知变量顺序时很省事：K2

已知一个**总顺序**就不会成环；每个点只在其**祖先**里挑父集，独立打分，挑最好的一组父→这就是 K2。

5. 不仅限表格 CPD

- 线性高斯、条件高斯 DAG**都有**闭式边缘似然**；
- 更一般的模型就用 BIC 近似（= 拟合度 - 复杂度惩罚）。

6. 规模化的现实：精确法顶多十来个点

DAG 数量**爆炸**；精确 DP 找 MAP 或算边存在概率到 ~16 个结点就不行了。

实务上：

- 贪心爬山**（加/删/反向一条边），因是**局部改动**，Bayes 因子可**常数时间更新**；配**多次重启、遗传、退火**更稳。常从**最优树**初始化，或用**依赖网络**的马尔可夫毯**限边提速**。
- 知识发现**想要“这条边的后验概率”：对小图可精确 DP；大图就**从后验抽样** DAG (MH)。更快的是**折叠** (order-space) MCMC：只采结点**全序**，把选父这步边缘化，混合更快。

小比喻

把 DAG 当“配方”：节点是原料，箭头是“先放谁再放谁”的依赖。

- 等价类**：有些步骤顺序互换也不影响成品口味（数据看不出差别）。
- BDe/BDeu 打分**：口味合不合（拟合）+步骤别太啰嗦（复杂度惩罚）。
- K2：已知大致做菜顺序时，每步只在“之前的原料”里选需要谁。
- 近似搜索/MCMC**：菜谱太多，没法全试，就用聪明的局部改动或抽样来找好配方。

要点速记

- **学什么**: 贝叶斯网结构 (DAG) , 在**完全数据下**评估 $p(G | D)/p(D | G)$ 。
- **可识别性**: 只能到马尔可夫等价为止 (同骨架+同 v 结构) 。
- **打分**: 用边缘似然; BDe/BDeu给出**可分解闭式**; α =等效样本量。
- **算法**: 已知顺序→K2; 一般情况→**贪心+重启/退火/遗传**, 从**最优树起步**、用**马尔可夫毯限边提速**。
- **其他 CPD**: 高斯/条件高斯可闭式; 其余用 BIC。
- **后验摘要** (知识发现) : 算**边的后验概率**; 大图用 (**折叠**) MCMC。

一句话记忆: “DAG 结构学习=等价类约束 + BDe(u)打分 + 小图精确/大图近似 (K2、贪心、MCMC) 。”

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。

26.5 Learning DAG structure with latent variables

Sometimes the complete data assumption does not hold, either because we have missing data, and/ or because we have hidden variables. In this case, the marginal likelihood is given by

$$p(D|G) = \int p(D, h|\theta, G)p(\theta|G)d\theta = \int p(D, h|\theta, G)p(\theta|G)d\theta \quad (26.33)$$

where h represents the hidden or missing data.

In general this is intractable to compute. For example, consider a mixture model, where we don't observe the cluster label. In this case, there are KN possible completions of the data (assuming we have K clusters); we can evaluate the inner integral for each one of these assignments to h , but we cannot afford to evaluate all of the integrals. (Of course, most of these integrals will correspond to hypotheses with little posterior support, such as assigning single data points to isolated clusters, but we don't know ahead of time the relative weight of these assignments.)

In this section, we discuss some ways for learning DAG structure when we have latent variables and/or missing data.

26.5.1 Approximating the marginal likelihood when we have missing data

The simplest approach is to use standard structure

learning methods for fully visible DAGs, but to approximate the marginal likelihood. In Section 24.7, we discussed some Monte Carlo methods for approximating the marginal likelihood. However, these are usually too slow to use inside of a search over models. Below we mention some faster deterministic approximations.

26.5.1.1 BIC approximation

A simple approximation is to use the BIC score, which is given by

$$\text{BIC}(G) = \log p(D|\hat{\theta}, G) - \log N 2 \dim(G) \quad (26.34)$$

where $\dim(G)$ is the number of degrees of freedom in the model and $\hat{\theta}$ is the MAP or ML estimate. However, the BIC score often severely underestimates the true marginal likelihood (Chickering and Heckerman 1997), resulting in it selecting overly simple models.

26.5.1.2 Cheeseman-Stutz approximation

We now present a better method known as the Cheeseman-Stutz approximation (CS) (Cheeseman and Stutz 1996). We first compute a MAP estimate of the parameters $\hat{\theta}$ (e.g., using EM). Denote the expected sufficient statistics of the data by $D = D(\hat{\theta})$; in the case of discrete variables, we just “fill in” the hidden variables with their expectation. We then use the exact marginal likelihood equation on this filled-in data:

$$p(D|G) \approx p(D|G) = p(D|\theta, G)p(\theta|G)d\theta \quad (26.35)$$

However, comparing this to Equation 26.33, we can see that the value will be exponentially smaller, since it does not sum over all values of θ . To correct for this, we first write

$$\log p(D|G) = \log p(D|G) + \log p(D|G) - \log p(D|G) \quad (26.36)$$

and then we apply a BIC approximation to the last two terms:

$$\begin{aligned} \log p(D|G) - \log p(D|G) &\approx \log p(D|\hat{\theta}, G) - N 2 \dim(\hat{\theta}) \\ (26.37) \end{aligned}$$

$$- \log p(D|\hat{\theta}, G) - N 2 \dim(\hat{\theta}) \quad (26.38)$$

$$= \log p(D|\hat{\theta}, G) - \log p(D|\hat{\theta}, G) \quad (26.39)$$

Putting it altogether we get

$$\begin{aligned} \log p(D|G) &\approx \log p(D|G) + \log p(D|\hat{\theta}, G) - \log p(D|\hat{\theta}, G) \\ (26.40) \end{aligned}$$

The first term $p(D|G)$ can be computed by plugging in the filled-in data into the exact marginal likelihood. The second term $p(D|\hat{\theta}, G)$, which involves an exponential sum (thus matching the “dimensionality” of the left hand side) can be computed using an inference algorithm. The final term $p(D|\hat{\theta}, G)$ can be computed by plugging in the filled-

in data into the regular likelihood.

26.5.1.3 Variational Bayes EM

An even more accurate approach is to use the variational Bayes EM algorithm. Recall from Section 21.6 that the key idea is to make the following factorization assumption:

$$p(\theta, z_{1:N}|D) \approx q(\theta)q(z) = q(\theta) \prod_i q(z_i) \quad (26.41)$$

where z_i are the hidden variables in case i . In the E step, we update the $q(z_i)$, and in the M step, we update $q(\theta)$.

The corresponding variational free energy provides a lower bound on the log marginal likelihood. In (Beal and Ghahramani 2006), it is shown that this bound is a much better approximation to the true log marginal likelihood (as estimated by a slow annealed importance sampling procedure) than either BIC or CS. In fact, one can prove that the variational bound will always be more accurate than CS (which in turn is always more accurate than BIC).

26.5.1.4 Example: college plans revisited

Let us revisit the college plans dataset from Section 26.4.2.5. Recall that if we ignore the possibility of hidden variables there was a direct link from socio economic status to IQ in the MAP DAG. Heckerman et al. decided to see what would happen if they introduced a hidden variable H , which they made a parent of both SES and IQ, representing a hidden common cause. They also considered a variant in which H points to SES, IQ and PE. For both such cases, they considered dropping none, one, or both of the SES-PE and PE-IQ edges. They varied the number of states for the hidden node from 2 to 6. Thus they computed the approximate posterior over $8 \times 5 = 40$ different models, using the CS approximation.

The most probable model which they found is shown in Figure 26.9. This is $2 \cdot 10^{10}$ times more likely than the best model containing no hidden variable. It is also $5 \cdot 10^9$ times more likely than the second most probable model with a hidden variable. So again the posterior is very peaked.

These results suggests that there is indeed a hidden common cause underlying both the socio-economic status of the parents and the IQ of the children. By examining the CPT entries, we see that both SES and IQ are more likely to be high when H takes on the value 1. They interpret this to mean that the hidden variable represents “parent quality” (possibly a genetic factor). Note, however, that the arc between H and SES can be reversed without changing the vstructures in the graph,

and thus without affecting the likelihood; this underscores the difficulty in interpreting hidden variables.

Interestingly, the hidden variable model has the same conditional independence assumptions amongst the visible variables as the most probable visible variable model. So it is not possible to distinguish between these hypotheses by merely looking at the empirical conditional independencies in the data (which is the basis of the constraint-based approach to structure learning (Pearl and Verma 1991; Spirtes et al. 2000)). Instead, by adopting a Bayesian approach, which takes parsimony into account (and not just conditional independence), we can discover the possible existence of hidden factors. This is the basis of much of scientific and everyday human reasoning (see e.g. (Griffiths and Tenenbaum 2009) for a discussion).

26.5.2 Structural EM

One way to perform structural inference in the presence of missing data is to use a standard search procedure (deterministic or stochastic), and to use the methods from Section 26.5.1 to estimate the marginal likelihood.

However, this approach is very efficient, because the marginal likelihood does not decompose when we have missing data, and nor do its approximations. For example, if we use the CS approximation or the VBEM approximation, we have to perform inference in every neighboring model, just to evaluate the quality of a single move!

(Friedman 1997b; Thiesson et al. 1998) presents a much more efficient approach called the structural EM algorithm. The basic idea is this: instead of fitting each candidate neighboring graph and then filling in its data, fill in the data once, and use this filled-in data to evaluate the score of all the neighbors. Although this might be a bad approximation to the marginal likelihood, it can be a good enough approximation of the difference in marginal likelihoods between different models, which is all we need in order to pick the best neighbor.

More precisely, define $D(G_0, \theta^0)$ to be the data filled in using model G_0 with MAP parameters θ^0 . Now define a modified BIC score as follows:

$$\text{scoreBIC}(G, D) = \log p(D|\theta, G) - \log N \cdot 2 \cdot \text{dim}(G) + \log p(G) + \log p(\theta|G) \quad (26.42)$$

where we have included the log prior for the graph and parameters. One can show (Friedman 1997b) that if we pick a graph G which increases the BIC score relative to

G_0 on the expected data, it will also increase the score on the actual data, i.e.,

$$\begin{aligned} \text{scoreBIC}(G, D(G_0, \hat{\theta}_0)) - \text{scoreBIC}(G_0, D(G_0, \hat{\theta}_0)) &\leq \\ \text{scoreBIC}(G, D) - \text{scoreBIC}(G_0, D) &(26.43) \end{aligned}$$

To convert this into an algorithm, we proceed as follows.

First we initialize with some graph G_0 and some set of parameters θ_0 . Then we fill-in the data using the current parameters — in practice, this means when we ask for the expected counts for any particular family, we perform inference using our current model. (If we know which counts we will need, we can precompute all of them, which is much faster.) We then evaluate the BIC score of all of our neighbors using the filled-in data, and we pick the best neighbor. We then refit the model parameters, fill-in the data again, and repeat. For increased speed, we may choose to only refit the model every few steps, since small changes to the structure hopefully won't invalidate the parameter estimates and the filled-in data too much. One interesting application is to learn a phylogenetic tree structure. Here the observed leaves are the DNA or protein sequences of currently alive species, and the goal is to infer the topology of the tree and the values of the missing internal nodes. There are many classical algorithms for this task (see e.g., (Durbin et al. 1998)), but one that uses SEM is discussed in (Friedman et al. 2002). Another interesting application of this method is to learn sparse mixture models (Barash and Friedman 2002). The idea is that we have one hidden variable C specifying the cluster, and we have to choose whether to add edges $C \rightarrow X_t$ for each possible feature X_t . Thus some features will be dependent on the cluster id, and some will be independent. (See also (Law et al. 2004) for a different way to perform this task, using regular EM and a set of bits, one per feature, that are free to change across data cases.)

26.5.3 Discovering hidden variables

In Section 26.5.1.4, we introduced a hidden variable “by hand”, and then figured out the local topology by fitting a series of different models and computing the one with the best marginal likelihood. How can we automate this process?

Figure 11.1 provides one useful intuition: if there is a hidden variable in the “true model”, then its children are likely to be densely connected. This suggest the following heuristic (Elidan et al. 2000): perform structure learning in

the visible domain, and then look for structural signatures, such as sets of densely connected nodes (near-cliques); introduce a hidden variable and connect it to all nodes in this near-clique; and then let structural EM sort out the details. Unfortunately, this technique does not work too well, since structure learning algorithms are biased against fitting models with densely connected cliques.

Another useful intuition comes from clustering. In a flat mixture model, also called a latent class model, the discrete latent variable provides a compressed representation of its children. Thus we want to create hidden variables with high mutual information with their children.

One way to do this is to create a tree-structured hierarchy of latent variables, each of which only has to explain a small set of children. (Zhang 2004) calls this a hierarchical latent class model. They propose a greedy local search algorithm to learn such structures, based on adding or deleting hidden nodes, adding or deleting edges, etc. (Note that learning the optimal latent tree is NP-hard (Roch 2006).)

Recently (Harmeling and Williams 2011) proposed a faster greedy algorithm for learning such models based on agglomerative hierarchical clustering. Rather than go into details, we just give an example of what this system can learn. Figure 26.10 shows part of a latent forest learned from the 20-newsgroup data. The algorithm imposes the constraint that each latent node has exactly two children, for speed reasons. Nevertheless, we see interpretable clusters arising. For example, Figure 26.10 shows separate clusters concerning medicine, sports and religion. This provides an alternative to LDA and other topic models (Section 4.2.2), with the added advantage that inference in latent trees is exact and takes time linear in the number of nodes.

An alternative approach is proposed in (Choi et al. 2011), in which the observed data is not constrained to be at the leaves. This method starts with the Chow-Liu tree on the observed data, and then adds hidden variables to capture higher-order dependencies between internal nodes. This results in much more compact models, as shown in Figure 26.11. This model also has better predictive accuracy than other approaches, such as mixture models, or trees where all the observed data is forced to be at the leaves.

Interestingly, one can show that this method can recover

the exact latent tree structure, providing the data is generated from a tree. See (Choi et al. 2011) for details. Note, however, that this approach, unlike (Zhang 2004; Harmeling and Williams 2011), requires that the cardinality of all the variables, hidden and observed, be the same. Furthermore, if the observed variables are Gaussian, the hidden variables must be Gaussian also.

26.5.4 Case study: Google's Rephil

In this section, we describe a huge DGM called Rephil, which was automatically learned from data. The model is widely used inside Google for various purposes, including their famous AdSense system.

The model structure is shown in Figure 26.12. The leaves are binary nodes, and represent the presence or absence of words or compounds (such as "New York City") in a text document or query. The latent variables are also binary, and represent clusters of co-occurring words. All CPDs are noisy-OR, since some leaf nodes (representing words) can have many parents. This means each edge can be augmented with a hidden variable specifying if the link was activated or not; if the link is not active, then the parent cannot turn the child on. (A very similar model was proposed independently in (Singliar and Hauskrecht 2006).)

Parameter learning is based on EM, where the hidden activation status of each edge needs to be inferred (Meek and Heckerman 1997). Structure learning is based on the old neuroscience idea that "nodes that fire together should wire together". To implement this, we run inference and check for cluster-word and cluster-cluster pairs that frequently turn on together. We then add an edge from parent to child if the link can significantly increase the probability of the child. Links that are not activated very often are pruned out. We initialize with one cluster per "document" (corresponding to a set of semantically related phrases). We then merge clusters A and B if A explains B's top words and vice versa. We can also discard clusters that are used too rarely.

The model was trained on about 100 billion text snippets or search queries; this takes several weeks, even on a parallel distributed computing architecture. The resulting model contains 12 million word nodes and about 1 million latent cluster nodes. There are about 350 million links in the model, including many cluster-cluster dependencies. The longest path in the graph has length 555, so the

model is quite deep.

Exact inference in this model is obviously infeasible.

However note that most leaves will be off, since most words do not occur in a given query; such leaves can be analytically removed, as shown in Exercise 10.7. We can also prune out unlikely hidden nodes by following the strongest links from the words that are on up to their parents to get a candidate set of concepts. We then perform iterative conditional modes to find a good set of local maxima. At each step of ICM, each node sets its value to its most probable state given the values of its neighbors in its Markov blanket. This continues until it reaches a local maximum. We can repeat this process a few times from random starting configurations. At Google, this can be made to run in 15 milliseconds!

26.5.5 Structural equation models

A structural equation model (Bollen 1989) is a special kind of directed mixed graph (Section 19.4.4.1), possibly cyclic, in which all CPDs are linear Gaussian, and in which all bidirected edges represent correlated Gaussian noise. Such models are also called path diagrams. SEMs are widely used, especially in economics and social science. It is common to interpret the edge directions in terms of causality, where directed cycles are interpreted as in terms of feedback loops (see e.g., (Pearl 2000, Ch.5)). However, the model is really just a way of specifying a joint Gaussian, as we show below. There is nothing inherently "causal" about it at all. (We discuss causality in Section 26.6.)

We can define an SEM as a series of full conditionals as follows:

$$x_i = \mu_i + \epsilon_i \quad (26.44)$$

where $\epsilon_i \sim N(0, \Psi)$. We can rewrite the model in matrix form as follows:

$$x = Wx + \mu \Rightarrow x = (I - W)^{-1}(\epsilon + \mu) \quad (26.45)$$

Hence the joint distribution is given by $p(x) = N(\mu, \Sigma)$ where $\Sigma = (I - W)^{-1}\Psi(I - W)^{-T}$ (26.46)

We draw an arc $X_i \leftarrow X_j$ if $|w_{ij}| > 0$. If W is lower triangular then the graph is acyclic. If, in addition, Ψ is diagonal, then the model is equivalent to a Gaussian DGM, as discussed in Section 10.2.5; such models are called recursive. If Ψ is not diagonal, then we draw a bidirected arc $X_i \leftrightarrow X_j$ for each non-zero off-diagonal term. Such edges represent correlation, possibly due to a hidden common cause.

When using structural equation models, it is common to partition the variables into latent variables, Z_t , and observed or manifest variables Y_t . For example, Figure 26.13 illustrates the following model:

$$(X_1 \ X_2 \ X_3 \ X_4 \ X_5 \ X_6) = (Z_1 \ Z_2 \\ Z_3 \ Y_1 \ Y_2 \ Y_3) + (w_{13} \ w_{14} \ w_{15} \ w_{16} \ w_{23} \ w_{24} \ w_{25} \ w_{26} \ w_{31} \ w_{32} \ w_{34} \ w_{35} \ w_{36}) \\ (Z_1 \ Z_2 \ Z_3 \ Y_1 \ Y_2 \ Y_3) + (w_{13} \ w_{14} \ w_{15} \ w_{16} \ w_{23} \ w_{24} \ w_{25} \ w_{26} \ w_{31} \ w_{32} \ w_{34} \ w_{35} \ w_{36}), \quad (26.47)$$

where

$$\Psi = (\psi_{11} \ \psi_{12} \ \psi_{13} \ \psi_{14} \ \psi_{15} \ \psi_{16} \ \psi_{21} \ \psi_{22} \ \psi_{23} \ \psi_{24} \ \psi_{25} \ \psi_{26} \ \psi_{31} \ \psi_{32} \ \psi_{33} \ \psi_{34} \ \psi_{35} \ \psi_{36}) \quad (26.48)$$

The presence of a feedback loop $Z_1 \rightarrow Z_2 \rightarrow Z_3$ is evident from the fact that W is not lower triangular. Also the presence of confounding between Y_1 and Y_2 is evident in the off-diagonal terms in Ψ .

Often we assume there are multiple observations for each latent variable. To ensure identifiability, we can set the mean of the latent variables Z_t to 0, and we can set the regression weights of $Z_t \rightarrow Y_t$ to 1. This essentially defines the scale of each latent variable. (In addition to the Z 's, there are the extra hidden variables implied by the presence of the bidirected edges.)

The standard practice in the SEM community, as exemplified by the popular commercial software package called LISREL (available from

<http://www.ssicentral.com/lisrel/>), is to build the structure by hand, to estimate the parameters by maximum likelihood, and then to test if any of the regression weights are significantly different from 0, using standard frequentist methods. However, one can also use Bayesian inference for the parameters (see e.g., (Dunson et al. 2005)). Structure learning in SEMs is rare, but since recursive SEMs are equivalent to Gaussian DAGs, many of the techniques we have been discussing in this section can be applied.

SEMs are closely related to factor analysis (FA) models (Chapter 12). The basic difference is that in an FA model, the latent Gaussian has a low-rank covariance matrix, and the observed noise has a diagonal covariance (hence no bidirected edges). In an SEM, the covariance of the latent Gaussian has a sparse Cholesky decomposition (at least if W is acyclic), and the observed noise might have a full covariance matrix.

Note that SEMs can be extended in many ways. For

example, we can add covariates/ input variables (possibly noisily observed), we can make some of the observations be discrete (e.g., by using probit links), and so on.

已思考 7s

下面先给出**完整、逐字且无遗漏的翻译**（含公式），再给出**通俗详尽的解释**，最后**总结要点**，便于快速回顾。

翻译

26.5 含有潜变量的 DAG 结构学习

有时**完全数据假设并不成立**，要么因为**有缺失数据**，要么因为存在**隐藏变量**。在这种情况下，边缘似然为

$$p(D | G) = \sum_h \int p(D, h | \theta, G) p(\theta | G) d\theta = \int_h \sum p(D, h | \theta, G) p(\theta | G) \quad (26.33)$$

其中 h 表示隐藏或缺失的数据。

一般来说这不可 tractable (难以计算)。例如，考虑一个**混合模型**，我们没有观察到**簇标签**。此时，数据存在 K^N 种可能的补全（假设有 K 个簇）；我们可以对每一种 h 的赋值求内层积分，但我们无法承担把所有积分都算一遍。（当然，其中多数积分对应的假设在后验中权重很小，比如把单个样本分到孤立的簇里；但我们事先并不知道这些赋值的相对权重。）

本节讨论在存在**潜变量和/或缺失数据**时进行 DAG 结构学习的一些方法。

26.5.1 在有缺失数据时近似边缘似然

最简单的做法是：继续使用**完全可见 DAG**的标准结构学习方法，但**近似**边缘似然。第 24.7 节讨论了用**蒙特卡罗方法**来近似边缘似然。然而，把它们放进“在模型空间上搜索”的循环里通常太慢。下面我们提到一些**更快的确定性近似**。

26.5.1.1 BIC 近似

一个简单近似是使用 BIC 评分：

$$\text{BIC}(G) = \log p(D | \hat{\theta}, G) - \frac{\log N}{2} \dim(G) \quad (26.34)$$

其中 $\dim(G)$ 是模型的自由度个数， $\hat{\theta}$ 是 MAP 或 ML 估计。然而，BIC 评分往往**会严重低估**真实的边缘似然 (Chickering and Heckerman 1997)，从而导致它选择**过于简单的模**

型。

26.5.1.2 Cheeseman–Stutz 近似

现在给出一个更好的方法，即 Cheeseman–Stutz 近似 (CS) (Cheeseman and Stutz 1996)。我们首先计算参数的 MAP 估计 $\hat{\theta}$ (例如用 EM)。用 $\tilde{D} = \tilde{D}(\hat{\theta})$ 记数据的**期望充分统计量**；在离散变量情形，我们就是把隐藏变量用其**期望**“填充”。然后在这份“填充后”的数据上使用**精确边缘似然公式**：

$$p(D | G) \approx p(\tilde{D} | G) = \int p(\tilde{D} | \theta, G) p(\theta | G) d\theta \quad (26.35)$$

然而，与式 (26.33) 对比，可以看出该数值会**指数级更小**，因为它**没有对所有 h 的取值求和**。为修正这一点，先写成

$$\log p(D | G) = \log p(\tilde{D} | G) + \underbrace{[\log p(D | G) - \log p(\tilde{D} | G)]}_{\text{差值项}} \quad (26.36)$$

然后对最后两个项做 BIC 近似：

$$\begin{aligned} \log p(D | G) - \log p(\tilde{D} | G) &\approx [\log p(D | \hat{\theta}, G) - \frac{N}{2} \dim(\hat{\theta})] - [\log p(\tilde{D} | \hat{\theta}, G) - \frac{N}{2} \dim(\hat{\theta})] \\ &= \log p(D | \hat{\theta}, G) - \log p(\tilde{D} | \hat{\theta}, G) \end{aligned}$$

把它们合起来得到

$$\log p(D | G) \approx \log p(\tilde{D} | G) + \log p(D | \hat{\theta}, G) - \log p(\tilde{D} | \hat{\theta}, G) \quad (26.40)$$

其中第一项 $p(\tilde{D} | G)$ 用“填充数据”的**精确边缘似然**计算；第二项 $p(D | \hat{\theta}, G)$ 含有一个**指数和**（从而与左侧的“维度”匹配），可用一个**推断算法**计算；最后一项 $p(\tilde{D} | \hat{\theta}, G)$ 用“填充数据”的**普通似然**计算。

26.5.1.3 变分贝叶斯 EM (VBEM)

更准确的方法是使用**变分贝叶斯 EM**。回忆第 21.6 节的关键思想：做如下因式分解假设

$$p(\theta, z_{1:N} | D) \approx q(\theta)q(z) = q(\theta) \prod_i q(z_i) \quad (26.41)$$

其中 z_i 是第 i 个样本的隐藏变量。在 E 步更新 $q(z_i)$ ，在 M 步更新 $q(\theta)$ 。相应的**变分自由能**给出了 $\log p(D)$ 的一个**下界**。在 (Beal and Ghahramani 2006) 中表明，该下界比 BIC 或 CS 更接近真实的对数边缘似然（后者用缓慢的**退火重要性采样**估计）。事实上，可证明变分下界**总是比 CS 更准确**（而 CS 又总比 BIC 更准确）。

26.5.1.4 例子：重访 college plans 数据集

重访 26.4.2.5 节的 college plans 数据集。回忆：若忽略隐藏变量的可能性，MAP DAG 里有从**社会经济地位 (SES)** 到 IQ 的直接边。Heckerman 等人决定：**若引入一个隐藏变量 H，把它作为 SES 和 IQ 的共同父结点**（表示隐藏的共同原因），会怎样？他们还考虑了一个变体：让 H 指向 SES、IQ 和 PE。对这两种情形，他们又考虑去掉 SES→PE 与 PE→IQ 两条边中的 0、1 或 2 条，并把隐藏结点的**状态数**从 2 到 6 取不同值。于是他们对 $8 \times 5 = 40$ 个模型，用 CS 近似计算了近似后验。

他们找到的**最可能模型**如图 26.9 所示。它的可能性比“**不含隐藏变量的最佳模型**”高 $2 \cdot 10^{10}$ 倍；也比“**含隐藏变量的第二佳模型**”高 $5 \cdot 10^9$ 倍。因此后验再次**极度尖峰**。

这些结果表明：确实存在一个**隐藏的共同原因**，同时作用于父母的社会经济地位 (SES) 与孩子的 IQ。检查 CPT 可见：当 $H = 1$ 时，SES 和 IQ 都更可能取高值。他们把这个隐藏变量解释为“**父母质量**”（可能是遗传因素）。但要注意：图中 H 与 SES 之间的弧可以反向而不改变图中的 v 结构，因此不会改变似然；这凸显了解释隐藏变量的困难。

有趣的是：该**隐藏变量模型在可见变量之间的条件独立假设**与“**最可能的（不含隐藏变量的）模型**”是**相同的**。因此仅凭经验性的 CI（这正是**基于约束**的结构学习方法（Pearl and Verma 1991; Spirtes et al. 2000）所依据的）**无法区分**这些假设。相反，通过采用一个**贝叶斯方法**（考虑**简约性**，而不仅是 CI），我们才能发现可能存在的**隐藏因素**。这也是大量科学与日常人类推理的基础（参见如（Griffiths and Tenenbaum 2009）的讨论）。

26.5.2 结构 EM (Structural EM)

在有缺失数据的情况下做结构推断的一种方式是：使用标准的**搜索程序**（确定性或随机），并用 26.5.1 节的方法估计边缘似然。然而，这种方式**非常低效**，因为当有缺失数据时，**边缘似然不再分解**，其近似也不分解。例如，若用 CS 或 VBEM 近似，我们在**每一个邻居模型**上都要做一次**推断**，仅仅是为了评估一次“移动”的质量！

(Friedman 1997b; Thiesson et al. 1998) 提出了更高效的方法：**结构 EM**。基本思想是：与其“对每个候选邻居都拟合一遍、再各自填充数据”，不如**只填充一次数据**，然后用这份“填充数据”来评估**所有邻居**的评分。尽管这可能是对边缘似然的一个较差近似，但对**不同模型之间边缘似然差值**的近似可能已经足够好——而这正是挑选“最佳邻居”所需的全部信息。

更精确地，记 $\tilde{D}(G_0, \hat{\theta}_0)$ 为用模型 G_0 与 MAP 参数 $\hat{\theta}_0$ “填充”得到的数据。定义修改版的 BIC 评分：

$$\text{score}_{\text{BIC}}(G, \tilde{D}) = \log p(\tilde{D} | \hat{\theta}, G) - \frac{\log N}{2} \dim(G) + \log p(G) + \log p(\hat{\theta}) \quad (26.42)$$

其中我们把图与参数的**对数先验**也加进来了。可以证明 (Friedman 1997b) : 若在**期望数据**上使得某图 G 的 BIC 分数相对 G_0 **上升**, 那么它在**真实数据**上的分数也会上升, 即

$$\text{score}_{\text{BIC}}(G, \tilde{D}(G_0, \hat{\theta}_0)) - \text{score}_{\text{BIC}}(G_0, \tilde{D}(G_0, \hat{\theta}_0)) \leq \text{score}_{\text{BIC}}(G, D) - \text{score}_{\text{BIC}}(G_0, D)$$

将其转为算法如下: 先用某个图 G_0 与参数 θ_0 初始化; 然后用当前参数**填充数据**——实际操作是, 当我们需要某个“(结点, 父集) 家庭”的**期望计数**时, 就用当前模型做**推断**得到它们。(若提前知道会需要哪些计数, 可**预先批量计算**, 更快。) 随后, 用这份“填充数据”评估所有邻居的 BIC 分数, 选出**最佳邻居**; **再重估参数、再次填充**, 反复迭代。为提速, 可选择隔几步才重估一次参数, 因为结构的小改动希望不会过度破坏当前的参数估计与填充数据。

一个有趣应用是学习**系统发育树 (phylogenetic tree) **的结构。此处叶子是现存物种的 DNA 或蛋白序列, 目标是推断树的拓扑与内部节点的缺失值。该任务有许多经典算法 (如 Durbin 等, 1998), 而使用 SEM 的一种做法见 (Friedman et al. 2002)。

另一项应用是学习**稀疏混合模型** (Barash and Friedman 2002)。思路是: 有一个隐藏变量 C 指定簇, 我们要决定是否为每个特征 X_t 添加边 $C \rightarrow X_t$ 。于是, 部分特征**依赖于簇 ID**, 部分**独立**。(另见 (Law et al. 2004) ——不同做法: 用常规 EM, 加上一组“按样本可变”的**特征开关位**。)

26.5.3 发现隐藏变量

在 26.5.1.4 节中, 我们是**手工**引入了一个隐藏变量, 然后通过拟合一系列不同模型、**选边缘似然最佳者**来确定局部拓扑。如何**自动化**这一过程?

图 11.1 提供了一个直觉: 若“真实模型”里存在隐藏变量, 那么其**子结点往往彼此高密连接**。由此得到如下启发式 (Elidan et al. 2000): 先在**可见域**上做结构学习, 再寻找结构特征, 比如**近团 (near-clique)** ——即一组高密连边的结点; 引入一个隐藏变量并把它连到该近团的所有结点上; 然后让**结构 EM**来调整细节。不幸的是, 这招效果不佳, 因为结构学习算法通常对“高密团”模型**有偏见** (不倾向拟合)。

另一个直觉来自聚类。在**平坦混合模型 (潜类别模型)** 中, 离散潜变量**为其子结点提供了一种压缩表示**。因此, 我们希望创造出与其子结点有**高互信息**的隐藏变量。

一种方法是构造**树结构的潜变量层次**, 每个隐藏变量只需解释一个子集 (Zhang 2004 称之为**分层潜类别模型**)。他们提出了一个**贪心的局部搜索算法**来学习此类结构 (添加/删除隐藏结点、添加/删除边等)。(注意: 学习**最优潜树**是 NP-hard 的 (Roch 2006)。)

近期 (Harmeling and Williams 2011) 提出了更快的贪心算法, 基于**自底向上的层次聚类 (凝聚式)**。不展开细节, 仅展示其可学习的样子: 图 26.10 给出了从 20-newsgroup 数

据学到的一个潜森林的部分。该算法出于速度考虑，约束每个隐藏结点恰好有两个子结点。尽管如此，仍能得到可解释的簇：如图中出现了关于医学、体育、宗教的独立簇。这提供了 LDA 与其他主题模型（4.2.2 节）的一个替代方案，并具有一个额外优势：在潜树中推断是精确的、而且时间对结点数是线性的。

另一个方法由 (Choi et al. 2011) 提出，其中观测变量不必都在叶子。该方法从观测数据上的 Chow–Liu 树开始，然后在内部结点之间加入隐藏变量以捕获高阶依赖。这会得到更紧凑的模型（见图 26.11），并且其预测精度优于其他方法（如混合模型，或把所有观测都强制在叶子的树）。有趣的是：若数据确由一棵树生成，则可证明该方法能恢复精确的潜树结构 (Choi et al. 2011)。不过，与 (Zhang 2004; Harmeling and Williams 2011) 不同，这种方法要求所有变量（隐藏与观测）具有相同的基数；若观测变量是高斯，隐藏变量也必须是高斯。

26.5.4 案例研究：Google 的 Rephil

本节描述一个名为 Rephil 的巨大 DGM，它由数据自动学习得到。该模型在 Google 内部广泛使用，包括著名的 AdSense 系统。

结构如图 26.12。叶子是二值结点，表示某文本或查询中某个词或短语（如 “New York City”）是否出现。潜变量也为二值，表示词的共现簇。所有 CPD 都是 noisy-OR，因为某些叶子（词）可能有很多父结点。这意味着每条边都可以带一个隐藏变量，表示该链路是否被“激活”；若链路未激活，父结点就不能把子结点“打开”。（非常相似的模型也见 (Singliar and Hauskrecht 2006)。）

参数学习基于 EM，需要对每条边的隐藏激活状态进行推断 (Meek and Heckerman 1997)。结构学习基于一个老的神经科学想法：“一起激活的结点应该连边”。实现上，我们先做推断，检查簇-词与簇-簇对是否经常一起被激活；若把一条父→子边加入能显著提高子结点的概率，就加上；不常激活的边会被剪枝。初始化时，每个“文档”（一组语义相关短语）对应一个簇。若簇 A 能解释簇 B 的高频词而反之亦然，就将它们合并。也会丢弃使用过少的簇。

该模型以约 1000 亿文本片段或查询进行训练；即便在并行的分布式架构上，也要数周。最后模型包含 1200 万词结点与约 100 万潜簇结点；共有约 3.5 亿条边，其中有大量的簇-簇依赖。图中最长路径长度达 555，因此模型相当深。

在该模型上做精确推断显然不可行。不过注意：大多数叶子是“关”的（给定查询，大多数词不出现）；可按习题 10.7 的方法将这类叶子解析性移除。还可以从“被点亮的词”一直沿最强边向上追溯其父结点，得到一组候选概念，并据此剪除不太可能的隐藏结点。随后使用 ICM（迭代条件众数）寻找良好的局部极大：在 ICM 的每步中，每个结点把自身设

为“在其马尔可夫毯给定值下”的**最可能状态**，直到达到局部极值。可从若干随机初始配置重复几次。在 Google，这整套流程可做到15 毫秒！

26.5.5 结构方程模型 (SEM)

结构方程模型 (Bollen 1989) 是一类特殊的**有向混合图** (19.4.4.1 节)，可能含环，其所有 CPD 均为**线性高斯**，且所有**双向边表示相关的高斯噪声**。这类模型也称 **路径图** (path diagrams)。SEM 在经济学、社会科学中广泛使用。人们常将其边方向解释为**因果**，把有向环解释为**反馈回路** (见 Pearl 2000, 第 5 章)。然而，正如下文所示，该模型本质上仅仅在指定一个**联合高斯分布**；它本身并不“自带因果” (因果性见 26.6 节)。

我们可把 SEM 定义为一组**全条件**：

$$x_i = \mu_i + \sum_{j \neq i} w_{ij} x_j + \epsilon_i, \quad \epsilon \sim N(0, \Psi) \quad (26.44)$$

矩阵形式：

$$\mathbf{x} = W\mathbf{x} + \mu + \epsilon \Rightarrow \mathbf{x} = (I - W)^{-1}(\epsilon + \mu) \quad (26.45)$$

因此联合分布为 $p(\mathbf{x}) = N(\mu, \Sigma)$ ，其中

$$\Sigma = (I - W)^{-1}\Psi(I - W)^{-T} \quad (26.46)$$

当 $|w_{ij}| > 0$ 时画弧 $X_i \leftarrow X_j$ 。若 W 为**下三角**则图**无环**。若进一步 Ψ 为**对角**，则模型等价于 10.2.5 节的**高斯 DGM**；此类模型称**递归** (recursive)。若 Ψ 非对角，则对每个**非零的非对角项**画一条**双向弧** $X_i \leftrightarrow X_j$ 。这些边表示**相关性**，可能由**隐藏共同原因**导致。

使用 SEM 时，通常把变量分成**潜变量** Z_t 与**观测 (显变量)** Y_t 。例如，图 26.13 对应如下模型：

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{pmatrix} = \underbrace{\begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}}_W = \begin{pmatrix} 0 & 0 & w_{13} & 0 & 0 & 0 \\ w_{21} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{32} & 0 & 0 \\ 0 & 0 & 0 & w_{41} & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{52} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{63} \end{pmatrix} \begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \end{pmatrix} \quad (26.47)$$

其中

$$\Psi = \begin{pmatrix} \Psi_{11} & 0 & 0 & 0 & 0 & 0 \\ 0 & \Psi_{22} & 0 & 0 & 0 & 0 \\ 0 & 0 & \Psi_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & \Psi_{44} & \Psi_{45} & 0 \\ 0 & 0 & 0 & \Psi_{54} & \Psi_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & \Psi_{66} \end{pmatrix} \quad (26.48)$$

由于 W 非下三角，可见存在反馈回路 $Z_1 \rightarrow Z_2 \rightarrow Z_3$ 。又因 Ψ 的非对角项，可见 Y_1 与 Y_2 存在混杂（相关噪声）。

常假定每个潜变量有多个观测。为确保可识别性，可将潜变量 Z_t 的均值设为 0，并将 $Z_t \rightarrow Y_t$ 的回归权重设为 1，这等价于为每个潜变量定标。（除 Z 外，双向边还隐含了额外的隐藏变量。）

SEM 社区的标准实践（如商业软件 LISREL）是：手工构建结构、用 ML 估计参数，然后用频率学方法检验哪些回归权重显著非零。不过，也可以对参数做贝叶斯推断（例如 Dunson et al. 2005）。在 SEM 中做结构学习较少；但若是递归 SEM（等价于高斯 DAG），本节讨论的许多技术即可适用。

SEM 与因子分析（FA）模型（第 12 章）密切相关。主要区别是：在 FA 中，潜高斯的协方差是低秩的，观测噪声协方差对角（因此无双向边）；在 SEM 中，潜高斯的协方差具有稀疏 Cholesky 分解（若 W 无环时），而观测噪声的协方差可为满矩阵。

注意 SEM 可有多种扩展：例如加入协变量/输入（可能含噪观测）、让部分观测为离散（如 probit 链接）等。

通俗详解（不依赖高数）

这节在讲啥？一句话版

当数据里有缺失或存在看不见的原因（潜变量）时，如何学贝叶斯网结构？精确的贝叶斯计算几乎做不了，于是就用近似打分（BIC、CS、VBEM）、结构 EM 来搜索结构，并且还介绍了如何发现潜变量、一个大规模工业案例（Rephil），以及 SEM 作为线性高斯的“潜变量+有向混合图”的特例。

1) 边缘似然为什么难？

因为你既要对参数积分，又要对隐藏/缺失变量求和（式 26.33）。

- 例如混合模型：每个样本的簇标签未知 $\rightarrow K^N$ 种组合，爆炸。

2) 三种“打分近似”（给图打分好不好）

- BIC：快，但常常过度惩罚复杂度 \rightarrow 选得太简单。

- **CS (Cheeseman–Stutz)** : 先用 EM 得到参数 $\hat{\theta}$, 把隐藏部分用**期望填上** (“软填充”), 在这份“填充数据”上套**精确边缘似然**, 再加一个**校正项** (式 26.40) 以补回“未对所有隐藏变量求和”的差距。效果**显著优于 BIC**。
- **VBEM (变分贝叶斯 EM)** : 用 $q(\theta) \prod_i q(z_i)$ 近似后验, 得到 $\log p(D)$ 的**下界**, 实践上**最稳、最准确** (普遍优于 CS, 远胜 BIC)。

小结: 能用 VBEM 就用 VBEM; 不能就 CS; BIC 作最简替代但别太信它的绝对值。

3) 隐变量真的会改变结论吗? (college plans)

- 不加隐藏变量时: 出现 “SES → IQ” 直连。
- 加一个隐藏 H (共同原因) 后: 模型**后验大幅提升**, 表明“**看不见的父母质量**”可能同时影响 SES 与 IQ。
- 但要小心: 某些边方向 (如 $H \leftrightarrow SES$) 在不改变 v 结构时可互换, **似然不变** → **解释性**要谨慎。
- 也说明: 仅用“条件独立检验”很难发现隐藏因素; **贝叶斯 (含简约性) **更能把“**看不见的因子**”挖出来。

4) 结构 EM: 在缺失/隐变量时更高效的结构搜索

- 朴素做法: 每改一次边就重新推断+打分, 极慢。
- **结构 EM:** 先用当前模型**统一填充**“期望数据”, 然后**一次性评估**所有邻居的分数, 只在挑中邻居后再**重估参数/重填充**。
- 有理论保证: 在“填充数据”上分数上升 \Rightarrow 在真数据上也会上升 (式 26.43)。
- 应用: 系统发育树; **稀疏混合模型** (决定每个特征是否依赖簇)。

5) 如何自动发现隐藏变量?

- 直觉 A (near-clique) : 可见图中某一群点**连得很密** → 加一个隐藏父连到它们, 再用结构 EM 微调。但算法一般**不偏好高密团**, 效果一般。
- 直觉 B (信息压缩) : 潜变量应与其子结点有**高互信息**。
- 实践做法:
 - **分层潜类别模型** (潜树/潜森林) : 贪心地**加/删隐藏点与边**, 学出层次结构 (推断**精确且线性时间**)。
 - Choi 等方法: 先做 Chow–Liu 树, 再为内部结点间的高阶依赖加隐藏变量 → **更紧凑更准**; 还能在真由树生成时**恢复真结构**。但要求所有变量**同一基数** (高斯也要全高斯)。

6) 工业级大模型: Google Rephil

- 词与短语作叶子 (**二值**), 上面是**二值潜簇**, noisy-OR 作为 CPD。
- 结构学习遵循“**一起亮就连边**”, 再**合并**相互解释的簇, **剪枝**不常激活的连边。

- 规模巨无霸：1200 万词结点、100 万簇簇、3.5 亿边、最深555层；训练用千亿片段/查询，耗时数周。
- 推断：不可能精确。利用“多数词未出现”做解析性裁剪，再向上追强边锁定候选概念，最后用 ICM 找局部极值；工程上可做到毫秒级响应。

7) SEM：线性高斯的“潜+混合图”

- 方程： $x = Wx + \mu + \epsilon$, $\epsilon \sim N(0, \Psi) \rightarrow x \sim N(\mu, \Sigma)$ 。
 - 有向边由 W 的非零项给出； Ψ 的非对角对应双向边（相关噪声/隐藏混杂）。
 - 若 W 下三角且 Ψ 对角 \Rightarrow 与高斯 DAG 等价（递归 SEM）。
 - 常用：手工建模、ML 估参、显著性检验；也可做贝叶斯。
 - 与因子分析：FA 的潜协方差低秩、观测噪声对角；SEM 的潜协方差更像稀疏 Cholesky，观测噪声可满协方差。
 - 可扩展：加协变量、离散观测（probit）等。
-

要点小结

- 难点：含缺失/潜变量时，边缘似然要对参数积分、对隐藏求和，通常不可解。
- 近似评分：
 - BIC：快但易过简；
 - CS：EM 填充 + 校正项，较准；
 - VBEM：变分下界，最稳最准 ($VBEM \geq CS \geq BIC$)。
- 隐藏因素的识别：仅靠 CI 检验常分不清，贝叶斯含简约性能揭示“看不见的共同原因”。
- 结构 EM：一次填充、批量评估邻居，高效搜索结构；有“上升保持”保证。
- 发现潜变量的策略：近团启发、互信息、潜树/潜森林（精确线性推断）、内部结点加隐藏（Choi 等）。
- 大规模实践（Rephil）：noisy-OR、簇合并+剪枝、ICM 近似推断，实现毫秒级。
- SEM：线性高斯的有向混合图（可含环与双向边）；递归情形等价于高斯 DAG；与 FA 关系密切，常手工建模+ML，也可贝叶斯。

记忆口诀：“隐/缺 – 精确难；评分靠 VBEM/CS；结构靠 SEM (Structural EM)；潜因靠层次潜树或内部加隐；工业级靠裁剪+ICM；SEM 是线性高斯版的混合图。”

这一节主要在讲什么？我不太懂

已思考 7s

不急~这节 (26.5) 的主线很清楚：当数据里有缺失值或“看不见的原因”（潜变量）时，**怎样学贝叶斯网 (DAG) 的结构**。精确的贝叶斯计算做不起，就用更“划算”的近似打分与搜索策略；还讲了如何“发现”潜变量、一个工业级大模型实例 (Google Rephil)，以及线性高斯世界里的亲戚——**结构方程模型 (SEM)**。

一句话版

有隐/缺时，真正的边缘似然要对参数积分还要对隐藏变量求和，几乎算不动；所以用**近似评分** (BIC/CS/VBEM) + **结构 EM**来搜结构；想找潜因，可用**潜树/层次潜类**等启发；大规模实践可像 Rephil 一样靠剪枝+贪心式推断；在线性高斯下，SEM给出另一套“潜+相关噪声”的表示，但本质仍是高斯联合分布（不是自动的因果）。

核心内容（通俗版）

- **难点来源**：边缘似然 $p(D|G)$ 同时对参数和隐藏变量积分/求和，像混合模型那样会有 K^N 种“填充”可能，组合爆炸。
- **三种近似打分**
 - BIC：快，但常过度偏好简单模型（低估真边缘似然）。
 - CS (Cheeseman–Stutz)：先用 EM 得到参数，再用**期望填充**的数据套精确边缘似然，再加校正项；比 BIC 准。
 - VBEM (变分贝叶斯 EM)：用可分解的 $q(\theta) \prod q(z_i)$ 近似后验，得到对数证据的**下界**；一般最稳最准 ($\text{VBEM} \geq \text{CS} \geq \text{BIC}$)。
- **案例提醒：隐藏因子会改结论**
 - 在“上大学”数据上，加一个共同父 H (“父母质量/遗传”等) 后，模型的后验大幅提高，说明**潜在共同原因**更解释得通。
 - 但**边方向**在等价类内可互换，**解释要谨慎**；单靠“条件独立检验”常分不清有无潜因，**贝叶斯（含简约性）**更能挖出潜因。
- **结构 EM (Structural EM) = 高效搜结构的工程套路**
 - 朴素做法每改一条边都重跑一次推断，太慢。
 - 结构 EM：**先统一填充一次期望数据**，用它给**所有邻居**打分，再只对选中的邻居**重估参数/再填充**；有理论保障“在填充数据上升 \Rightarrow 真数据也上升”。
 - 可用于系统发育树、**稀疏混合模型**（决定每个特征是否依赖簇）。
- **自动“发现”潜变量的路子**
 - 直觉：**同受一因的孩子们常彼此连得很密**，或与某潜因互信息高。
 - 实践：
 - **分层潜类/潜树/潜森林**（每个潜因解释一小撮孩子，推断精确且线性时间），用贪心加/删隐节点与边；

- Choi 等：先学观测的 Chow-Liu 树，再在内部加隐节点捕捉高阶依赖 → 更紧凑、预测更好（有同基数/同高斯等限制）。
- 工业实例：Google Rephile
 - 词作为叶子（二值），上层是二值潜簇；CPD 用 noisy-OR。
 - 结构规则：“一起亮就接线”，合并能互相解释的簇，剪掉不活跃的边。
 - 规模：1200万词节点、100万潜簇、3.5亿边、最深555层；训练用千亿片段/查询。
 - 推断：靠“多数词不出现”做解析裁剪，沿强边上溯找候选概念，再用 ICM 找局部最优；可做到毫秒级。
- SEM（结构方程模型）小抄
 - 线性高斯、可含环、可有双向边（相关噪声/隐藏混杂），本质上就是在参数化一个高斯联合；“因果”解读需谨慎。
 - 递归（无环且噪声对角）时等价高斯 DAG；与因子分析相关但不相同。

要点小结

- 有隐/缺时，真实 $p(D|G)$ 不可算 → 用近似打分：VBEM > CS > BIC。
- 结构 EM：一次填充、批量评估邻居，显著提速结构搜索。
- 仅靠条件独立难发现潜因；贝叶斯+简约性能揭示隐藏共同原因。
- 潜因发现：潜树/层次潜类、内部加隐节点（Choi 等）是两条实用路。
- 大规模：noisy-OR + 剪枝 + ICM 等工程技巧能把庞大模型跑到实时。
- SEM：线性高斯的“潜+相关噪声”框架；不是天然因果，递归情形可用本章技巧学结构。

记法：“**隐/缺→精确难；打分靠VBEM/CS；搜索用结构EM；找潜因靠潜树/内部加隐；工业靠剪枝+ICM；SEM是线性高斯版的潜模型。**”

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。

26.6 Learning causal DAGs

Causal models are models which can predict the effects of interventions to, or manipulations of, a system. For example, an electronic circuit diagram implicitly provides a compact encoding of what will happen if one removes any given component, or cuts any wire. A causal medical model might predict that if I continue to smoke, I am

likely to get lung cancer (and hence if I cease smoking, I am less likely to get lung cancer). Causal claims are inherently stronger, yet more useful, than purely associative claims, such as "people who smoke often have lung cancer".

Causal models are often represented by DAGs (Pearl 2000), although this is somewhat controversial (Dawid 2010). We explain this causal interpretation of DAGs below. We then show how to use a DAG to do causal reasoning. Finally, we briefly discuss how to learn the structure of causal DAGs. A more detailed description of this topic can be found in (Pearl 2000) and (Koller and Friedman 2009, Ch.21).

26.6.1 Causal interpretation of DAGs

In this section, we define a directed edge $A \rightarrow B$ in a DAG to mean that "A directly causes B", so if we manipulate A, then B will change. This is known as the causal Markov assumption. (Of course, we have not defined the word "causes", and we cannot do that by appealing to a DAG, lest we end up with a cyclic definition; see (Dawid 2010) for further discussion of this point.)

We will also assume that all relevant variables are included in the model, i.e., there are no unknown confounders, reflecting hidden common causes. This is called the causal sufficiency assumption. (If there are known to be confounders, they should be added to the model, although one can sometimes use mixed directed graphs (Section 26.5.5) as a way to avoid having to model confounders explicitly.)

Assuming we are willing to make the causal Markov and causal sufficiency assumptions, we can use DAGs to answer causal questions. The key abstraction is that of a perfect intervention; this represents the act of setting a variable to some known value, say setting X_i to x_i . A real world example of such a perfect intervention is a gene knockout experiment, in which a gene is "silenced". We need some notational convention to distinguish this from observing that X_i happens to have value x_i . We use Pearl's do calculus notation (as in the verb "to do") and write $\text{do}(X_i = x_i)$ to denote the event that we set X_i to x_i . A causal model can be used to make inferences of the form $p(x|\text{do}(X_i = x_i))$, which is different from making inferences of the form $p(x|X_i = x_i)$.

To understand the difference between conditioning on interventions and conditioning on observations (i.e., the

difference between doing and seeing), consider a 2 node DGM $S \rightarrow Y$, in which $S = 1$ if you smoke and $S = 0$ otherwise, and $Y = 1$ if you have yellow-stained fingers, and $Y = 0$ otherwise. If I observe you have yellow fingers, I am licensed to infer that you are probably a smoker (since nicotine causes yellow stains):

$$p(S = 1|Y = 1) > p(S = 1) \quad (26.49)$$

However, if I intervene and paint your fingers yellow, I am no longer licensed to infer this, since I have disrupted the normal causal mechanism. Thus

$$p(S = 1|do(Y = 1)) = p(S = 1) \quad (26.50)$$

One way to model perfect interventions is to use graph surgery: represent the joint distribution by a DGM, and then cut the arcs coming into any nodes that were set by intervention. See Figure 26.14 for an example. This prevents any information flow from the nodes that were intervened on from being sent back up to their parents. Having performed this surgery, we can then perform probabilistic inference in the resulting “mutilated” graph in the usual way to reason about the effects of interventions. We state this formally as follows.

Theorem 26.6.1 (Manipulation theorem (Pearl 2000; Spirtes et al. 2000)). To compute $p(X_i|do(X_j))$ for sets of nodes i, j , we can perform surgical intervention on the X_j nodes and then use standard probabilistic inference in the mutilated graph.

We can generalize the notion of a perfect intervention by adding interventions as explicit action nodes to the graph. The result is like an influence diagram, except there are no utility nodes (Lauritzen 2000; Dawid 2002). This has been called the augmented DAG (Pearl 2000). We can then define the CPD $p(X_i|do(X_i))$ to be anything we want. We can also allow an action to affect multiple nodes. This is called a fat hand intervention, a reference to someone trying to change a single component of some system (e.g., an electronic circuit), but accidentally touching multiple components and thereby causing various side effects (see (Eaton and Murphy 2007) for a way to model this using augmented DAGs).

26.6.2 Using causal DAGs to resolve Simpson’s paradox

In this section, we assume we know the causal DAG. We can then do causal reasoning by applying d-separation to the mutilated graph. In this section, we give an example of this, and show how causal reasoning can help resolve a famous paradox, known as Simpson’s paradox.

Simpson's paradox says that any statistical relationship between two variables can be reversed by including additional factors in the analysis. For example, suppose some cause C (say, taking a drug) makes some effect E (say getting better) more likely

$$P(E|C) > P(E|\neg C)$$

and yet, when we condition on the gender of the patient, we find that taking the drug makes the effect less likely in both females (F) and males ($\neg F$):

$$P(E|C,F) < P(E|\neg C,F)$$

$$P(E|C,\neg F) < P(E|\neg C,\neg F)$$

This seems impossible, but by the rules of probability, this is perfectly possible, because the event space where we condition on $(\neg C, F)$ or $(\neg C, \neg F)$ can be completely different to the event space when we just condition on $\neg C$. The table of numbers below shows a concrete example (from (Pearl 2000, p175)):

| | Combined | Male | Female | E | $\neg E$ | Total | Rate E | $\neg E$ | Total | Rate E | | | |
|----------|----------|------|--------|-----|----------|-------|----------|----------|-------|----------|-----|-----|-----|
| $\neg E$ | | | | 20 | 20 | 40 | 50% | 18 | 12 | 30 | 60% | | |
| Total | C | 20 | 20 | 40 | 50% | 18 | 12 | 30 | 60% | 2 | 8 | 10 | 20% |
| $\neg C$ | 16 | 24 | 40 | 40% | 7 | 3 | 10 | 70% | 9 | 21 | 30 | 30% | |
| Total | 36 | 44 | 80 | | | | | | | | | | |
| | 25 | 15 | 40 | 11 | 29 | 40 | y | | | | | | |

From this table of numbers, we see that

$$p(E|C)=20/40 = 0.5 > p(E|\neg C)=16/40 = 0.4 \quad (26.51)$$

$$p(E|C,F)=2/10 = 0.2 < p(E|\neg C,F)=9/30 = 0.3 \quad (26.52)$$

$$p(E|C,\neg F)=18/30 = 0.6 < p(E|\neg C,\neg F)=7/10 = 0.7 \quad (26.53)$$

A visual representation of the paradox is given in Figure 26.15. The line which goes up and to the right shows that the effect (y-axis) increases as the cause (x-axis) increases. However, the dots represent the data for females, and the crosses represent the data for males. Within each subgroup, we see that the effect decreases as we increase the cause.

It is clear that the effect is real, but it is still very counter-intuitive. The reason the paradox arises is that we are interpreting the statements causally, but we are not using proper causal reasoning when performing our calculations. The statement that the drug C causes recovery E is

$$P(E|do(C)) > P(E|do(\neg C)) \quad (26.54)$$

whereas the data merely tell us

$$P(E|C) > P(E|\neg C) \quad (26.55)$$

This is not a contradiction. Observing C is positive evidence for E, since more males than females take the drug, and the male recovery rate is higher (regardless of the drug). Thus Equation 26.55 does not imply Equation

26.54.

Nevertheless, we are left with a practical question: should we use the drug or not? It seems like if we don't know the patient's gender, we should use the drug, but as soon as we discover if they are male or female, we should stop using it. Obviously this conclusion is ridiculous.

To answer the question, we need to make our assumptions more explicit. Suppose reality can be modeled by the causal DAG in Figure 26.16(a). To compute the causal effect of C on E, we need to adjust for (i.e., condition on) the confounding variable F. This is necessary because there is a backdoor path from C to E via F, so we need to check the $C \rightarrow E$ relationship for each value of F separately, to make sure the relationship between C and E is not affected by any value of F.

Suppose that for each value of F, taking the drug is harmful, that is,

$$p(E|do(C), F) < p(E|do(\neg C), F) \quad (26.56)$$

$$p(E|do(C), \neg F) < p(E|do(\neg C), \neg F) \quad (26.57)$$

Then we can show that taking the drug is harmful overall:

$$p(E|do(C)) < p(E|do(\neg C)) \quad (26.58)$$

The proof is as follows (Pearl 2000, p181). First, from our assumptions in Figure 26.16(a), we see that drugs have no effect on gender

$$p(F|do(C)) = p(F|do(\neg C)) = p(F) \quad (26.59)$$

Now using the law of total probability,

$$\begin{aligned} p(E|do(C)) &= p(E|do(C), F)p(F|do(C)) + p(E|do(C), \\ &\quad \neg F)p(\neg F|do(C)) \end{aligned} \quad (26.60)$$

$$= p(E|do(C), F)p(F) + p(E|do(C), \neg F)p(\neg F) \quad (26.61)$$

Similarly,

$$\begin{aligned} p(E|do(\neg C)) &= p(E|do(\neg C), F)p(F) + p(E|do(\neg C), \neg F)p(\neg F) \\ & \quad (26.62) \end{aligned}$$

Since every term in Equation 26.61 is less than the corresponding term in Equation 26.62, we conclude that

$$p(E|do(C)) < p(E|do(\neg C)) \quad (26.63)$$

So if the model in Figure 26.16(a) is correct, we should not administer the drug, since it reduces the probability of the effect.

Now consider a different version of this example. Suppose we keep the data the same but interpret F as something that is affected by C, such as blood pressure. See Figure 26.16(b). In this case, we can no longer assume

$$p(F|do(C)) = p(F|do(\neg C)) = p(F) \quad (26.64)$$

and the above proof breaks down. So $p(E|do(C)) - p(E|do(\neg C))$ may be positive or negative.

In the true model is Figure 26.16(b), then we should not condition on F when assessing the effect of C on E, since there is no backdoor path in this case, because of the v-structure at F. That is, conditioning on F might block one of the causal pathways. In other words, by comparing patients with the same post-treatment blood pressure (value of F), we may mask the effect of one of the two pathways by which the drug operates to bring about recovery.

Thus we see that different causal assumptions lead to different causal conclusions, and hence different courses of action. This raises the question on whether we can learn the causal model from data. We discuss this issue below.

26.6.3 Learning causal DAG structures

In this section, we discuss some ways to learn causal DAG structures.

26.6.3.1 Learning from observational data

In Section 26.4, we discussed various methods for learning DAG structures from observational data. It is natural to ask whether these methods can recover the “true” DAG structure that was used to generate the data. Clearly, even if we have infinite data, an optimal method can only identify the DAG up to Markov equivalence (Section 26.4.1). That is, it can identify the PDAG (partially directed acyclic graph), but not the complete DAG structure, because all DAGs which are Markov equivalent have the same likelihood.

There are several algorithms (e.g., the greedy equivalence search method of (Chickering 2002)) that are consistent estimators of PDAG structure, in the sense that they identify the true Markov equivalence class as the sample size goes to infinity, assuming we observe all the variables. However, we also have to assume that the generating distribution p is faithful to the generating DAG G . This means that all the conditional independence (CI) properties of p are exactly captured by the graphical structure, so $I(p) = I(G)$; this means there cannot be any CI properties in p that are due to particular settings of the parameters (such as zeros in a regression matrix) that are not graphically explicit. For this reason, a faithful distribution is also called a stable distribution.

Suppose the assumptions hold and we learn a PDAG. What can we do with it? Instead of recovering the full graph, we can focus on the causal analog of edge

marginals, by computing the magnitude of the causal effect of one node on another (say A on B). If we know the DAG, we can do this using techniques described in (Pearl 2000). If the DAG is unknown, we can compute a lower bound on the effect as follows (Maathuis et al. 2009): learn an equivalence class (PDAG) from data; enumerate all the DAGs in the equivalence class; apply Pearl's do-calculus to compute the magnitude of the causal effect of A on B in each DAG; finally, take the minimum of these effects as the lower bound. It is usually computationally infeasible to compute all DAGs in the equivalence class, but fortunately one only needs to be able to identify the local neighborhood of A and B, which can be estimated more efficiently, as described in (Maathuis et al. 2009).

This technique is called IDA, which is short for "intervention-calculus when the DAG is absent".

In (Maathuis et al. 2010), this technique was applied to some yeast gene expression data. Gene knockout data was used to estimate the "ground truth" effect of each 234 single-gene deletions on the remaining 5,361 genes. Then the algorithm was applied to 63 unperturbed (wild-type) samples, and was used to rank order the likely targets of each of the 234 genes. The method had a precision of 66% when the recall was set to 10%; while low, this is substantially more than rival variable-selection methods, such as lasso and elastic net, which were only slightly above chance.

26.6.3.2 Learning from interventional data

If we want to distinguish between DAGs within the equivalence class, we need to use interventional data, where certain variables have been set, and the consequences have been measured. An example of this is the dataset in Figure 26.17(a), where proteins in a signalling pathway were perturbed, and their phosphorylation status was measured using a technique called flow cytometry (Sachs et al. 2005).

It is straightforward to modify the standard Bayesian scoring criteria, such as the marginal likelihood or BIC score, to handle learning from mixed observational and experimental data: we just compute the sufficient statistics for a CPD's parameter by skipping over the cases where that node was set by intervention (Cooper and Yoo 1999). For example, when using tabular CPDs, we modify the counts as follows:

$$N_{tck} = \sum_{xit \text{ not set}} I(x_i, t = k, x_i, pa(t) = c) \quad (26.65)$$

The justification for this is that in cases where node t is set by force, it is not sampled from its usual mechanism, so such cases should be ignored when inferring the parameter θ_t . The modified scoring criterion can be combined with any of the standard structure learning algorithms. (He and Geng 2009) discusses some methods for choosing which interventions to perform, so as to reduce the posterior uncertainty as quickly as possible (a form of active learning).

The preceding method assumes the interventions are perfect. In reality, experimenters can rarely control the state of individual molecules. Instead, they inject various stimulant or inhibitor chemicals which are designed to target specific molecules, but which may have side effects. We can model this quite simply by adding the intervention nodes to the DAG, and then learning a larger augmented DAG structure, with the constraint that there are no edges between the intervention nodes, and no edges from the "regular" nodes back to the intervention nodes.

Figure 26.17(b) shows the augmented DAG that was learned from the interventional flow cytometry data depicted in Figure 26.17(a). In particular, we plot the median graph, which includes all edges for which $p(G|D) = 1|D) > 0.5$. These were computed using the exact algorithm of (Koivisto 2006). It turns out that, in this example, the median model has exactly the same structure as the optimal MAP model, $\text{argmax}_G p(G|D)$, which was computed using the algorithm of (Koivisto and Sood 2004; Silander and Myllymaki 2006).

下面先给出逐句完整翻译（含公式/表格；不做任何省略），再用通俗详解解释关键概念，最后要点小结方便复习。

翻译

26.6 学习因果 DAG

因果模型是能够预测对一个系统进行**干预 (intervention) /操控 (manipulation) **所产生效果的模型。举例来说，一张电子电路图隐含地紧凑编码了：如果拆掉任意元件、或切断任意导线，会发生什么。一种因果医学模型可能会预测：如果我继续吸烟，我很可能得肺癌（因此如果我停止吸烟，我得肺癌的可能性会更小）。因果断言天生比纯“关联性”断言更强、但也更有用，例如“吸烟的人经常患肺癌”。

因果模型常用 DAG 表示 (Pearl 2000) , 尽管这有些争议 (Dawid 2010) 。我们在下文解释将 DAG 作因果解释的方式。然后展示如何用 DAG 做因果推理。最后, 我们简要讨论如何学习因果 DAG 的结构。对此主题更详细的描述可见 (Pearl 2000) 以及 (Koller and Friedman 2009, 第 21 章) 。

26.6.1 对 DAG 的因果解释

在本节中, 我们把 DAG 中的一条有向边 $A \rightarrow B$ 定义为“**A 直接导致 B**”, 因而如果我们操控 A, 则 B 会改变。这被称为**因果马尔可夫假设 (causal Markov assumption)**。(当然, 我们并未定义“导致”的含义; 而且我们不能借助 DAG 来作定义, 否则会陷入循环定义; 见 (Dawid 2010) 对此点的进一步讨论。)

我们还将假设**所有相关变量都已纳入模型**, 即不存在未知的混杂因素 (隐藏的共同原因)。这被称为**因果充分性假设 (causal sufficiency)**。(如果已知存在混杂因素, 应把它们加入模型; 尽管有时也可使用**有向混合图** (第 26.5.5 节) 来避免显式建模混杂。)

在愿意接受因果马尔可夫与因果充分性假设的前提下, 我们可以用 DAG 来回答因果问题。关键抽象是**完美干预 (perfect intervention)**; 它代表把某个变量设为已知值的动作, 比如把 X_i 设为 x_i 。一个现实例子是**基因敲除实验**, 在其中一个基因被“沉默”。我们需要一个记号来区分“干预设定 $X_i = x_i$ ”与“观察到 X_i 恰好等于 x_i ”: 我们使用 Pearl 的 **do 演算记号** (动词 “to do”), 写作 $\text{do}(X_i = x_i)$ 表示我们将 X_i 设为 x_i 。因果模型可用于作形如 $p(x | \text{do}(X_i = x_i))$ 的推断, 这与作形如 $p(x | X_i = x_i)$ 的推断不同。

为理解“对干预做条件化”与“对观测做条件化”的差异 (即 **doing** 与 **seeing** 的差别), 考虑一个两结点 DGM: $S \rightarrow Y$, 其中 $S = 1$ 表示你吸烟、 $S = 0$ 否则; $Y = 1$ 表示你手指泛黄、 $Y = 0$ 否则。如果我**观察到**你手指发黄, 我有理由推断你很可能是烟民 (因为尼古丁会导致黄渍) :

$$p(S = 1 | Y = 1) > p(S = 1) \quad (26.49)$$

然而, 如果我**干预**把你的手指涂成黄色, 我就不再有理由作此推断, 因为我破坏了正常的因果机制。因此

$$p(S = 1 | \text{do}(Y = 1)) = p(S = 1) \quad (26.50)$$

建模完美干预的一种方式是做**图外科手术 (graph surgery)**: 用一个 DGM 表示联合分布, 然后**切断**所有被干预设定之结点的入弧。例见图 26.14。这会阻止来自被干预结点的信息回流至其父结点。完成此“手术”后, 我们即可在所得的“**残缺 (mutilated)**”图上按常规方式进行概率推断, 以推理干预效应。形式化表述如下。

定理 26.6.1 (操纵定理; Pearl 2000; Spirtes 等 2000)。要计算结点集合 i, j 的 $p(X_i | \text{do}(X_j))$, 我们可在 X_j 上实施外科干预, 然后在该残缺图上使用标准的概率推断。

我们可以把完美干预推广为，将干预作为**显式的动作结点加入图**。其结果类似于**影响图** (influence diagram)，只是没有效用结点 (Lauritzen 2000; Dawid 2002)。这被称为**增广 DAG (augmented DAG)** (Pearl 2000)。此时我们可以把 CPD $p(X_i \mid \text{do}(X_i))$ 定义为任意想要的形式。我们也可允许一个动作**同时影响多个结点**，这被称为**“厚手 (fat hand)”干预**，意指有人想只改某个系统部件 (如电子电路)，却一不小心碰到多个部件而引发各种副作用 (见 (Eaton and Murphy 2007) 使用增广 DAG 建模的方法)。

26.6.2 用因果 DAG 解决辛普森悖论

本节我们假设**因果 DAG 已知**。随后可将 **d-分离** 应用于残缺图来作因果推理。本节给出一个例子，展示因果推理如何帮助解决著名的**辛普森悖论**。

辛普森悖论说：任意两个变量之间的统计关系，都可能在加入额外因素后被**反转**。例如，设某原因 C (如服药) 让某结果 E (如痊愈) 更可能发生：

$$P(E \mid C) > P(E \mid \neg C)$$

然而，当我们对病人的**性别**作条件化时，发现对男女两组，服药都让结果发生的可能性更低：

$$P(E \mid C, F) < P(E \mid \neg C, F)$$

$$P(E \mid C, \neg F) < P(E \mid \neg C, \neg F)$$

这看似不可能，但就概率论规则而言完全可能，因为对 $(\neg C, F)$ 或 $(\neg C, \neg F)$ 条件化时的事件空间，可能与仅对 $\neg C$ 条件化时的事件空间**完全不同**。下表给出一个具体数值例子 (来自 (Pearl 2000, 第 175 页))：

| | Combined | | | | Male | | | | Female | | | |
|----------|----------|----------|-----|------|------|----------|-----|------|--------|----------|-----|------|
| | E | $\neg E$ | Tot | Rate | E | $\neg E$ | Tot | Rate | E | $\neg E$ | Tot | Rate |
| C | 20 | 20 | 40 | 50% | 18 | 12 | 30 | 60% | 2 | 8 | 10 | 20% |
| $\neg C$ | 16 | 24 | 40 | 40% | 7 | 3 | 10 | 70% | 9 | 21 | 30 | 30% |
| Tot | 36 | 44 | 80 | | 25 | 15 | 40 | | 11 | 29 | 40 | y |

由此可见

$$p(E \mid C) = 20/40 = 0.5 > p(E \mid \neg C) = 16/40 = 0.4 \quad (26.51)$$

$$p(E \mid C, F) = 2/10 = 0.2 < p(E \mid \neg C, F) = 9/30 = 0.3 \quad (26.52)$$

$$p(E \mid C, \neg F) = 18/30 = 0.6 < p(E \mid \neg C, \neg F) = 7/10 = 0.7 \quad (26.53)$$

(图 26.15 给出悖论的可视化。向右上方的线说明：随着“原因”(横轴)增加，“结果”(纵轴)上升。然而点表示女性数据，叉表示男性数据；在每个子组内，我们看到“结果”下降

随“原因”增加。)

显然这里的效应是真实的，但却非常反直觉。悖论产生的原因是：我们在**用因果的方式理解**这些陈述，但在**计算时**并未使用正确的因果推理。断言“药物 C 导致康复 E ”应是

$$P(E \mid \text{do}(C)) > P(E \mid \text{do}(\neg C)) \quad (26.54)$$

而数据仅告诉我们

$$P(E \mid C) > P(E \mid \neg C) \quad (26.55)$$

这并不矛盾。**观察到** C 对 E 是正向证据，因为男性服药比率更高、且男性的康复率（不论是否服药）更高。因此式 (26.55) 并不蕴含式 (26.54)。

然而我们仍面临一个**实践性问题**：到底应不应该用药？似乎当我们不知道病人性别时应该用药，一旦知道其性别是男/女又该停药——显然这个结论**荒谬**。

要回答该问题，我们需要把假设讲清楚。假设现实可由图 26.16(a) 的**因果 DAG 建模**。要计算 C 对 E 的因果效应，我们需要对混杂变量 F 作**调整 (adjust)**（即条件化）。这是必要的，因为存在从 C 到 E 的**后门路径** $C \leftarrow F \rightarrow E$ ；因此我们需要对每个 F 的取值分别检查 $C \rightarrow E$ 的关系，确保它不受任何 F 取值的影响。

设对 F 的每个取值，服药都是**有害的**，即

$$p(E \mid \text{do}(C), F) < p(E \mid \text{do}(\neg C), F) \quad (26.56)$$

$$p(E \mid \text{do}(C), \neg F) < p(E \mid \text{do}(\neg C), \neg F) \quad (26.57)$$

于是可推出总体上服药也是**有害**：

$$p(E \mid \text{do}(C)) < p(E \mid \text{do}(\neg C)) \quad (26.58)$$

证明如下 (Pearl 2000, 第 181 页)。首先由图 26.16(a) 的假设可知，药物**不影响性别**：

$$p(F \mid \text{do}(C)) = p(F \mid \text{do}(\neg C)) = p(F) \quad (26.59)$$

由全概率公式，

$$\begin{aligned} p(E \mid \text{do}(C)) &= p(E \mid \text{do}(C), F)p(F \mid \text{do}(C)) + p(E \mid \text{do}(C), \neg F)p(\neg F \mid \text{do}(C)) \\ &= p(E \mid \text{do}(C), F)p(F) + p(E \mid \text{do}(C), \neg F)p(\neg F) \end{aligned} \quad (26.60)$$

同理，

$$p(E \mid \text{do}(\neg C)) = p(E \mid \text{do}(\neg C), F)p(F) + p(E \mid \text{do}(\neg C), \neg F)p(\neg F) \quad (26.61)$$

因为式 (26.61) 每一项都小于式 (26.60) 中对应项，因此得到

$$p(E \mid \text{do}(C)) < p(E \mid \text{do}(\neg C)) \quad (26.62)$$

所以如果图 26.16(a) 的模型正确，我们就不应给药，因为它降低了“结果”发生的概率。

现在考虑该例的另一版本。保持数据不变，但把 F 解释为受 C 影响的变量，例如**血压**；见图 26.16(b)。此时我们不再能假设

$$p(F \mid \text{do}(C)) = p(F \mid \text{do}(\neg C)) = p(F) \quad (26.64)$$

于是上述证明失效。因此 $p(E \mid \text{do}(C)) - p(E \mid \text{do}(\neg C))$ 可能是正的也可能是负的。

如果真实模型是图 26.16(b)，那么在评估 C 对 E 的效应时就**不应对 F 作条件化**，因为此时在 F 处形成了**v 结构**，不存在后门路径；对 F 条件化反而可能**阻断**某条因果通路。换言之，如果我们比较**具有相同治疗后血压 (F 的值) **的病人，可能会掩盖药物通过两条路径之一发挥作用以带来康复的效应。

因此我们看到：不同的**因果假设**导致不同的**因果结论**，从而引向不同的行动路线。这就引出了一个问题：我们能否从数据中**学习因果模型**？下面讨论此问题。

26.6.3 学习因果 DAG 结构

本节讨论学习因果 DAG 结构的一些方法。

26.6.3.1 来自观测数据的学习

在第 26.4 节，我们讨论了用**观测数据**学习 DAG 结构的多种方法。自然要问：这些方法能否恢复生成数据的“**真实**” DAG 结构？显然，即使有**无限数据**，一个最优方法也只能把 DAG 识别到**马尔可夫等价**为止（第 26.4.1 节）。也就是说，它能识别 PDAG（**部分有向无环图**），但不能得到完整的 DAG 结构，因为所有马尔可夫等价的 DAG 具有相同的似然。

存在若干算法（如 (Chickering 2002) 的**贪心等价类搜索**，GES），在样本量趋于无穷、且我们观察到**所有变量**的前提下，它们是 PDAG 结构的一致估计量（能收敛到真实的马尔可夫等价类）。但我们还必须假设生成分布 p 对生成 DAG G 是**忠实 (faithful)** 的。即 p 的全部条件独立性质恰被图结构捕获： $I(p) = I(G)$ 。这意味着在 p 中**不能**有一些由参数特定设定（如回归矩阵的零项）造成、但在图上**未显式体现**的条件独立。因而**忠实分布**也称**稳定分布**。

假设上述条件成立，且我们学得了 PDAG。那么我们能做什么？与其恢复完整图，不如关注**因果版的“边边际”**——即计算某结点对另一结点（如 A 对 B ）因果效应的大小。若 DAG 已知，可用 (Pearl 2000) 中的技术完成。若 DAG 未知，可以如下计算**下界** (Maathuis 等 2009)：先从数据学到一个等价类 (PDAG)；枚举该等价类中的所有 DAG；对每个 DAG 应用 do 演算计算 A 对 B 的因果效应；最后取这些效应的**最小值**作为下界。通常枚举等价类中**全部** DAG 是不可行的，不过幸运的是，只需识别 A 与 B 的**局部邻域**即可，可更高效

估计（见 Maathuis 等 2009）。该技术称为 IDA (“当 DAG 缺失时的干预演算”， Intervention-calculus when the DAG is Absent）。

在 (Maathuis 等 2010) 中，此技术被用于一些**酵母基因表达数据**。研究用**基因敲除数据**估计了 234 个“单基因删除”对其余 5,361 个基因的“**真实**”效应；然后把算法应用到 63 个**未扰动（野生型）样本上**，用于对每个基因的可能靶标**排序**。当召回率设为 10% 时，该方法的精度为 66%；尽管不高，但**显著优于**竞争性的变量选择方法，如 lasso 与 elastic net（它们仅略高于随机）。

26.6.3.2 来自干预数据的学习

如果我们想在一个等价类内部**区分** DAG，则需要使用**干预数据**：某些变量被设定，其后果被测量。一个例子是图 26.17(a) 的数据集：信号通路中的蛋白被**扰动**，其**磷酸化状态**通过**流式细胞术测量** (Sachs 等 2005)。

把标准的贝叶斯评分准则（如边缘似然或 BIC）扩展为能处理“观测+实验”混合数据是直接的：对每个 CPD 的参数，只需在统计充分统计量时**跳过**该结点被干预设定的那些样本 (Cooper and Yoo 1999)。例如，使用表格 CPD 时，把计数修改为：

$$N_{tck} = \sum_{i: x_{it} \text{ not set}} \mathbf{I}(x_{i,t} = k, x_{i,\text{pa}(t)} = c) \quad (26.65)$$

其理由是：当结点 t 被强制设定时，它**不是**从其通常机制中采样的，因此在推断参数 θ_t 时应该**忽略**这些样本。该修改后的评分准则可以与任何标准的结构学习算法结合使用。（He and Geng 2009）讨论了如何**选择进行哪种干预**，以尽快降低后验不确定性（即一种**主动学习**）。

上述方法假设干预是**完美的**。现实中，实验者很少能控制单个分子的状态；他们往往注入各种兴奋剂或抑制剂，目标是特定分子，但可能有**副作用**。我们可以很简单地建模：把**干预结点加入** DAG，然后学习更大的**增广** DAG 结构，并添加约束：干预结点之间**没有边**，且“常规”结点**不能回指**到干预结点。

图 26.17(b) 展示了从图 26.17(a) 的干预流式细胞数据学得的**增广** DAG。具体地，我们画出**中位图** (median graph)，其中包含所有满足 $p(G_{ij} = 1 | D) > 0.5$ 的边。这些概率用 (Koivisto 2006) 的精确算法计算。结果表明，在该例中，中位模型的结构与最优 MAP 模型 $\arg \max_G p(G | D)$ **完全相同**；后者通过 (Koivisto and Sood 2004; Silander and Myllymäki 2006) 的算法计算得到。

通俗详解（尽量少数学）

这节在讲啥？

当我们关心“**如果我做 X，会发生什么**”时，仅靠相关性不够，必须用**因果模型**。DAG 可以

在两个假设下承载因果：

- **因果马尔可夫**：箭头表示“直接因果”。
- **因果充分性**：没有漏掉的混杂（或者把混杂显式放进图，或用混合图）。

核心操作是 $\text{do}()$ ： $\text{do}(X = x)$ 表示“把 X 设为 x ”。这与“看见 $X = x$ ”不同：**观察**不会破坏原机制，**干预**会切断进入 X 的父因，改写机制。因此我们在图上做“**外科手术**”：切掉被设定结点的入边，再在“残缺图”上做普通推断，就能得到干预下的概率。

辛普森悖论为何不悖？

- 关联说 $P(E \mid C) > P(E \mid \neg C)$ 并不等于因果说 $P(E \mid \text{do}(C)) > P(E \mid \text{do}(\neg C))$
- 如果性别 F 同时影响“是否服药”与“康复”，那它是**混杂**；要评估 $C \rightarrow E$ ，就要对 F **调整**（分别看男女）。若两组里都“有害”，总体也“有害”。
- 但若 F 是**被药影响的**（如血压），就不该对 F 条件化，因为它位于 **v 结构**上，会把正当的因果路径给**挡掉**。

结论：不同因果结构假设 → 不同决策。不要用“看到”代替“做到”。

怎么从数据学因果图？

- **只有观测数据**：最多学到等价类（PDAG），还需要**忠实性**等假设。可用 IDA：在等价类内枚举（或用局部邻域），对每个候选图算 $A \rightarrow B$ 的效应，再取**最小值**作为稳健的**下界**。在基因表达上，IDA 的precision 明显优于 lasso/elastic net。
- **有干预数据**：把“被设定的样本”从该结点的计数里剔除（因为它们不是按原机制产生），即可把干预自然纳入打分（边缘似然/BIC）。若干预不完美，就把**干预结点**加入图里，学增广 DAG 并加拓扑约束。现实案例（流式细胞数据）显示：**中位图**与 MAP 结构一致。

要点小结

- **因果 ≠ 相关**： $\text{do}(X = x)$ 与 $X = x$ 本质不同；干预=切断被设定结点的入边，在残缺图上推断。
- **两大前提**：因果马尔可夫（箭头=直接因果）、因果充分性（无漏混杂/或显式建模）。
- **辛普森悖论**：关联可被分层反转；用因果图判断该不该**对谁调整**（混杂要调、后效不该调）。
- **观测学习**：只能到 PDAG（等价类），需忠实性；用 IDA 给因果效应算**下界**，有生物组学验证。
- **干预学习**：把被设定样本从该结点的 CPD 计数中剔除；不完美干预→用**增广 DAG**显式建模干预结点与约束。

- **实践：**对混合观测+干预数据，可用修改后的边缘似然/BIC + 标准结构搜索；中位图可与 MAP 一致。

一句话记忆：**因果要看 do(); 悖论靠“该调则调、不该调别调”；只观测到等价类、用 IDA 算下界；有干预就把它纳入计分或增广成带干预结点的 DAG。**

这一节主要在讲什么？我不太懂

已思考 12s

不怕，这节（26.6）的主线就三件事：**什么是“因果版”的 DAG、怎么用它回答“如果我做 X 会怎样”、以及怎么从数据里把因果结构学出来。**

一句话版

- **相关≠因果：**看见 $X = x$ 和**把 X 设成 x**（记作 $\text{do}(X=x)$ ）是两回事。
- **因果 DAG 的两条前提：**
 1. **因果马尔可夫：**有向边代表“直接因果”；
 2. **因果充分性：**别漏掉会共同影响两者的混杂因素（或把它们建进图/用混合图表示）。
- **做因果推断：**对被干预的结点**切断入边**（“图外科手术”），在得到的“残缺图”上做普通概率推断，就能算 $p(\cdot \mid \text{do}(\cdot))$ 。
- **辛普森悖论：**关联能被分组后反转；关键是**该不该对某变量做调整**：混杂要调，处在因果路径/碰撞结构上的变量**别乱调**。
- **学因果结构：**
 - 只有观测数据：最多学到**等价类 (PDAG)**，需**忠实性假设**；可用 IDA 算某结点对另一结点因果效应的**下界**。
 - 有干预数据：把“被设定”的样本**从该结点的计数里剔除**来打分；干预不完美就把**干预结点**建进增广 DAG 并加约束。

稍微展开（通俗不烧脑）

1) “做”和“看”的差别 (do vs see)

- 观察到手指黄 $Y = 1 \Rightarrow$ 更可能吸烟： $p(S = 1 \mid Y = 1) > p(S = 1)$ 。
- 但**把手指涂黄** $\text{do}(Y = 1)$ 并不会提高你是烟民的概率： $p(S = 1 \mid \text{do}(Y = 1)) = p(S = 1)$ 。
- 计算方法：把 Y 的入边切掉，再在这张“残缺图”上做推断。

2) 用因果图化解辛普森悖论

- 合并看：吃药 C 似乎让康复 E 更可能（关联上升）。
- 分男女看：男女各组内，吃药反而更差。
- 解决：真正的因果结论要看 $P(E \mid \text{do}(C))$ ；若性别 F 同时影响“是否吃药”和“康复”，它就是混杂，应对 F 分层/调整。
- 但若 F 是被药物影响的中介/或与其他箭头形成 \vee 结构，此时不该对 F 条件化，否则会把正当的因果路径挡掉。
- 结论：先画因果图再决定调不调整，别用“看到的相关性”替代“做出来的因果”。

3) 学因果 DAG (结构学习)

- 仅观测数据：
 - 再多数据也只能确定到马尔可夫等价类（方向有些不可辨）；像 GES 这类方法在“忠实性”下可一致恢复 PDAG。
 - 想评估 $A \rightarrow B$ 的作用强度但 DAG 不确定？用 IDA：在等价类里（或只用 A, B 的局部邻域）计算各候选 DAG 的因果效应，取最小值当稳健下界（实践里在基因数据上优于常见变量选择法）。
- 有干预数据：
 - 把被设定的样本从该结点的 CPD 计数里跳过，即可把干预自然纳入边缘似然/BIC 等打分；正常结构搜索就能用了。
 - 干预有副作用（不完美）？把干预结点显式加入图，禁止“干预 \leftrightarrow 干预”和“普通 \rightarrow 干预”的边，学习增广 DAG。实际案例显示，用这种做法学到的“中位图”常与 MAP 结构一致。

小比喻

把系统想成电路：

- 观察是看电表读数；干预是拔掉/短接某个元件（切入边）。
- 混杂像公共电源线同时给两个模块供电——要想知道 A 是否真影响 B ，就得把公共线的影响“扣掉”（调整）；但如果某线是 A 通往 B 的内部连线，你去“扣掉它”反而把真正的作用切断了。

要点小结

- 因果问题的关键是 $\text{do}(\cdot)$ ：干预 ≠ 观察；计算用“图手术”。
- 先验假设：箭头 = 直接因果（因果马尔可夫）；别漏混杂（因果充分性/或显式建模）。
- 辛普森悖论靠因果图决定“该调谁”：混杂要调，中介/碰撞器别调。
- 学结构：观测数据 → 只能到 PDAG（需忠实性），用 IDA 给效应一个下界；有干预 → 修改计分、或学增广 DAG 处理不完美干预。

记一句：“因果看 do(), 调谁看图；观测到等价类，干预来定向。”

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。

26.7 Learning undirected Gaussian graphical models

Learning the structured of undirected graphical models is easier than learning DAG structure because we don't need to worry about acyclicity. On the other hand, it is harder than learning DAG structure since the likelihood does not decompose (see Section 19.5). This precludes the kind of local search methods (both greedy search and MCMC sampling) we used to learn DAG structures, because the cost of evaluating each neighboring graph is too high, since we have to refit each model from scratch (there is no way to incrementally update the score of a model).

In this section, we discuss several solutions to this problem, in the context of Gaussian random fields or undirected Gaussian graphical models (GGM)s. We consider structure learning for discrete undirected models in Section 26.8.

26.7.1 MLE for a GGM

Before discussing structure learning, we need to discuss parameter estimation. The task of computing the MLE for a (non-decomposable) GGM is called covariance selection (Dempster 1972).

From Equation 4.19, the log likelihood can be written as

$$I(\Omega) = \log \det \Omega - \text{tr}(\Omega) \quad (26.66)$$

where $\Omega = \Sigma^{-1}$ is the precision matrix, and $S = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})(x_i - \bar{x})^T$ is the empirical covariance matrix. (For notational simplicity, we assume we have already estimated $\bar{\mu} = \bar{x}$.) One can show that the gradient of this is given by

$$\nabla I(\Omega) = \Omega^{-1} - S \quad (26.67)$$

However, we have to enforce the constraints that $\Omega_{st} = 0$ if $G_{st} = 0$ (structural zeros), and that Ω is positive definite. The former constraint is easy to enforce, but the latter is somewhat challenging (albeit still a convex constraint). One approach is to add a penalty term to the objective if Ω leaves the positive definite cone; this is the approach

used in `gmmFitMinfunc` (see also (Dahl et al. 2008)).

Another approach is to use a coordinate descent method, described in (Hastie et al. 2009, p633), and implemented in `gmmFitHtf`. Yet another approach is to use iterative proportional fitting, described in Section 19.5.7. However, IPF requires identifying the cliques of the graph, which is NP-hard in general.

Interestingly, one can show that the MLE must satisfy the following property: $\Sigma_{st} = S_{st}$ if $G_{st} = 1$ or $s = t$, i.e., the covariance of a pair that are connected by an edge must match the empirical covariance. In addition, we have $\Omega_{st} = 0$ if $G_{st} = 0$, by definition of a GGM, i.e., the precision of a pair that are not connected must be 0. We say that Σ is a positive definite matrix completion of S , since it retains as many of the entries in S as possible, corresponding to the edges in the graph, subject to the required sparsity pattern on $\Sigma - \mathbf{1}$, corresponding to the absent edges; the remaining entries in Σ are filled in so as to maximize the likelihood.

Let us consider a worked example from (Hastie et al. 2009, p652). We will use the following adjacency matrix, representing the cyclic structure, $X_1 - X_2 - X_3 - X_4 - X_1$, and the following empirical covariance matrix:

$$G = \begin{pmatrix} & & 0101 & 1010 & 0101 & 1010 & \\ & & | & | & | & | & \\ & & 1102 & 6 & 5 & 2 & 10 & 3 & 463 & 10 \end{pmatrix}, S = \begin{pmatrix} & & 10 & 154 \\ & & | & | & | & | & \\ & & 26.68 \end{pmatrix}$$

The MLE is given by

$$\Sigma = \begin{pmatrix} & & 10.00 & 1.00 & 1.31 & 4.00 & 1.00 & 10.00 & 2.00 & 0.87 & 1.31 \\ & & | & | & | & | & | & | & | & | & | \\ & & 2.00 & 10.00 & 3.00 & 4.00 & 0.87 & 3.00 & 10.00 & | & | & | \\ & & | & | & | & | & | & | & | & | & | \\ & & -0.01 & 0 & -0.05 & -0.01 & 0.11 & -0.02 & 0 & 0 & -0.02 & 0.11 & -0.03 \\ & & | & | & | & | & | & | & | & | & | & | \\ & & -0.05 & 0 & -0.03 & 0.13 & | & | & | & | & | & | \end{pmatrix}, \Omega = \begin{pmatrix} & & 0.12 \\ & & | & | & | & | & | & | & | & | & | \\ & & -0.01 & 0 & -0.05 & -0.01 & 0.11 & -0.02 & 0 & 0 & -0.02 & 0.11 & -0.03 \\ & & | & | & | & | & | & | & | & | & | & | \\ & & -0.05 & 0 & -0.03 & 0.13 & | & | & | & | & | & | \end{pmatrix} (26.69)$$

(See `gmmFitDemo` for the code to reproduce these numbers.) The constrained elements in Ω , and the free elements in Σ , both of which correspond to absent edges, have been highlighted.

26.7.2 Graphical lasso

We now discuss one way to learn a sparse GRF structure, which exploits the fact that there is a 1:1 correspondence between zeros in the precision matrix and absent edges in the graph. This suggests that we can learn a sparse graph structure by using an objective that encourages zeros in the precision matrix. By analogy to lasso (see Section 13.3), one can define the following 1 penalized NLL:

$$J(\Omega) = -\log \det \Omega + \text{tr}(S\Omega) + \lambda \|\Omega\|_1 \quad (26.70)$$

where $\|\Omega\|_1 = \sum_{j,k} |\omega_{jk}|$ is the 1-norm of the matrix. This is called the graphical lasso or Glasso.

Although the objective is convex, it is non-smooth (because of the non-differentiable 1 penalty) and is constrained (because Ω must be a positive definite matrix). Several algorithms have been proposed for optimizing this objective (Yuan and Lin 2007; Banerjee et al. 2008; Duchi et al. 2008), although arguably the simplest is the one in (Friedman et al. 2008), which uses a coordinate descent algorithm similar to the shooting algorithm for lasso. See ggmLassoHtf for an implementation. (See also (Mazumder and Hastie 2012) for a more recent version of this algorithm.)

As an example, let us apply the method to the flow cytometry dataset from (Sachs et al. 2005). A discretized version of the data is shown in Figure 26.17(a). Here we use the original continuous data. However, we are ignoring the fact that the data was sampled under intervention. In Figure 26.18, we illustrate the graph structures that are learned as we sweep λ from 0 to a large value. These represent a range of plausible hypotheses about the connectivity of these proteins.

It is worth comparing this with the DAG that was learned in Figure 26.17(b). The DAG has the advantage that it can easily model the interventional nature of the data, but the disadvantage that it cannot model the feedback loops that are known to exist in this biological pathway (see the discussion in (Schmidt and Murphy 2009)). Note that the fact that we show many UGMs and only one DAG is incidental: we could easily use BIC to pick the “best” UGM, and conversely, we could easily display several DAG structures, sampled from the posterior.

26.7.3 Bayesian inference for GGM structure

Although the graphical lasso is reasonably fast, it only gives a point estimate of the structure. Furthermore, it is not model-selection consistent (Meinshausen 2005), meaning it cannot recover the true graph even as $N \rightarrow \infty$. It would be preferable to integrate out the parameters, and perform posterior inference in the space of graphs, i.e., to compute $p(G|D)$. We can then extract summaries of the posterior, such as posterior edge marginals, $p(G_{ij} = 1|D)$, just as we did for DAGs. In this section, we discuss how to do this.

Note that the situation is analogous to Chapter 13, where we discussed variable selection. In Section 13.2, we discussed Bayesian variable selection, where we integrated out the regression weights and computed

$p(y|D)$ and the marginal inclusion probabilities $p(y_j = 1|D)$. Then in Section 13.3, we discussed methods based on L_1 regularization. Here we have the same dichotomy, but we are presenting them in the opposite order.

If the graph is decomposable, and if we use conjugate priors, we can compute the marginal likelihood in closed form (Dawid and Lauritzen 1993). Furthermore, we can efficiently identify the decomposable neighbors of a graph (Thomas and Green 2009), i.e., the set of legal edge additions and removals. This means that we can perform relatively efficient stochastic local search to approximate the posterior (see e.g. (Giudici and Green 1999; Armstrong et al. 2008; Scott and Carvalho 2008)).

However, the restriction to decomposable graphs is rather limiting if one's goal is knowledge discovery, since the number of decomposable graphs is much less than the number of general undirected graphs.

A few authors have looked at Bayesian inference for GGM structure in the non-decomposable case (e.g., (Dellaportas et al. 2003; Wong et al. 2003; Jones et al. 2005)), but such methods cannot scale to large models because they use an expensive Monte Carlo approximation to the marginal likelihood (Atay-Kayis and Massam 2005). (Lenkoski and Dobra 2008) suggested using a Laplace approximation.

This requires computing the MAP estimate of the parameters for Ω under a GWishart prior (Roverato 2002). In (Lenkoski and Dobra 2008), they used the iterative proportional scaling algorithm (Speed and Kiiveri 1986; Hara and Takimura 2008) to find the mode. However, this is very slow, since it requires knowing the maximal cliques of the graph, which is NP-hard in general.

In (Moghaddam et al. 2009), a much faster method is proposed. In particular, they modify the gradient-based methods from Section 26.7.1 to find the MAP estimate; these algorithms do not need to know the cliques of the graph. A further speedup is obtained by just using a diagonal Laplace approximation, which is more accurate than BIC, but has essentially the same cost. This, plus the lack of restriction to decomposable graphs, enables fairly fast stochastic search methods to be used to approximate $p(G|D)$ and its mode. This approach significantly outperformed graphical lasso, both in terms of predictive accuracy and structural recovery, for a comparable computational cost.

26.7.4 Handling non-Gaussian data using copulas

The graphical lasso and variants is inherently limited to data that is jointly Gaussian, which is a rather severe restriction. Fortunately the method can be generalized to handle non-Gaussian, but still continuous, data in a fairly simple fashion. The basic idea is to estimate a set of D univariate monotonic transformations f_j , one per variable j , such that the resulting transformed data is jointly Gaussian. If this is possible, we say the data belongs to the nonparametric Normal distribution, or nonparanormal distribution (Liu et al. 2009). This is equivalent to the family of Gaussian copulas (Klaassen and Wellner 1997). Details on how to estimate the f_j transformations from the empirical cdf's of each variable can be found in (Liu et al. 2009). After transforming the data, we can compute the correlation matrix and then apply glasso in the usual way. One can show, under various assumptions, that this is a consistent estimator of the graph structure, representing the CI assumptions of the original distribution(Liu et al. 2009).

已思考 17s

下面先给出**完整、逐字且无遗漏的翻译**（含公式/矩阵），随后是**通俗详解**，最后是**要点小结**。

翻译

26.7 学习无向高斯图模型

学习无向图模型的结构比学习 DAG 结构更容易，因为我们不需要担心无环性。另一方面，它又比学习 DAG 结构更难，因为其似然**不分解**（见第 19.5 节）。这就排除了我们用来学习 DAG 结构的那类**局部搜索方法**（贪心搜索与 MCMC 采样），因为评估每一个“邻居图”的代价太高——我们必须从头为每个模型重新拟合（无法增量更新一个模型的得分）。

本节中，我们在**高斯随机场**或**无向高斯图模型（GGM）**的语境下讨论若干解决方案。离散无向模型的结构学习见第 26.8 节。

26.7.1 GGM 的极大似然估计（MLE）

在讨论结构学习前，我们需要先讨论参数估计。对一个（不可分解的）GGM 计算 MLE 的任务称为**协方差选择**（Dempster 1972）。

由式 (4.19)，对数似然可写为

$$\ell(\Omega) = \log \det \Omega - \text{tr}(S\Omega) \quad (26.66)$$

其中 $\Omega = \Sigma^{-1}$ 是精度矩阵， $S = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})(x_i - \bar{x})^\top$ 是经验协方差矩阵。（为简化记号，我们假设均值 $\hat{\mu} = \bar{x}$ 已估计好。）可证其梯度为

$$\nabla \ell(\Omega) = \Omega^{-1} - S \quad (26.67)$$

不过我们必须强制：若 $G_{st} = 0$ 则 $\Omega_{st} = 0$ （结构性零），且 Ω 是正定的。前者容易强制，后者稍有挑战（但仍是凸约束）。一种做法是在 Ω 离开正定锥时对目标加惩罚；这就是 `gmmFitMinfunc` 的做法（亦见 Dahl et al., 2008）。另一种是使用（Hastie et al., 2009, p633）描述并在 `gmmFitHtf` 中实现的坐标下降法。再一种是使用第 19.5.7 节描述的迭代比例拟合（IPF）。但 IPF 需要识别图的团（clique），一般情形是 NP-难的。

有趣的是，MLE 必须满足如下性质：若 $G_{st} = 1$ 或 $s = t$ ，则 $\Sigma_{st} = S_{st}$ ，即由边连接的成对协方差必须等于样本协方差。此外，按 GGM 的定义，若 $G_{st} = 0$ 则 $\Omega_{st} = 0$ ，即未连接对的精度元素为 0。我们称 Σ 是 S 的一个正定矩阵补全：它尽可能保留 S 中与图中存在的边对应的条目，同时在 Σ^{-1} 上满足稀疏模式（对缺失的边），剩余的 Σ 条目通过最大化似然来“补齐”。

来看（Hastie et al., 2009, p652）的一个算例。使用下面的邻接矩阵（环结构 $X_1-X_2-X_3-X_4-X_1$ ）与经验协方差矩阵：

$$G = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}, \quad S = \begin{pmatrix} 10 & 1 & 5 & 4 \\ 1 & 10 & 2 & 6 \\ 5 & 2 & 10 & 3 \\ 4 & 6 & 3 & 10 \end{pmatrix} \quad (26.68)$$

其 MLE 为

$$\Sigma = \begin{pmatrix} 10.00 & 1.00 & 1.31 & 4.00 \\ 1.00 & 10.00 & 2.00 & 0.87 \\ 1.31 & 2.00 & 10.00 & 3.00 \\ 4.00 & 0.87 & 3.00 & 10.00 \end{pmatrix}, \quad \Omega = \begin{pmatrix} 0.12 & -0.01 & 0 & -0.05 \\ -0.01 & 0.11 & -0.02 & 0 \\ 0 & -0.02 & 0.11 & -0.03 \\ -0.05 & 0 & -0.03 & 0.13 \end{pmatrix}$$

（复现实验见 `gmmFitDemo`）。 Ω 中受约束的元素、以及 Σ 中自由的元素（两者都对应缺失的边）在原文里被高亮。

26.7.2 图形化 lasso (graphical lasso)

下面讨论学习稀疏 GRF 结构的一种方法，它利用了“精度矩阵中零与图中无边之间 1:1 的对应关系”。这启示我们：可通过鼓励精度矩阵出现零的目标函数来学习稀疏图结构。类比 lasso（见第 13.3 节），定义如下 ℓ_1 惩罚的负对数似然：

$$J(\Omega) = -\log \det \Omega + \text{tr}(S\Omega) + \lambda \|\Omega\|_1 \quad (26.70)$$

其中 $\|\Omega\|_1 = \sum_{j,k} |\omega_{jk}|$ 是矩阵的 ℓ_1 -范数。这称为**图形化 lasso** 或 **Glasso**。

虽说目标是**凸的**，但它**非光滑** (ℓ_1 项不可导)，且有“ Ω 必须正定”的**约束**。已有多种优化算法 (Yuan & Lin, 2007; Banerjee et al., 2008; Duchi et al., 2008)，其中相对最简单的可能是 (Friedman et al., 2008) 的**坐标下降** (类似 lasso 的“shooting”算法)，实现见 `ggmLassoHtf`。 (更近的版本见 Mazumder & Hastie, 2012。)

示例：将该方法应用于 (Sachs et al., 2005) 的**流式细胞术**数据。图 26.17(a) 给出了离散化版本；这里我们用**原始连续数据**。不过我们忽略了“数据是在干预下采样的”这一事实。图 26.18 展示了当 λ 从 0 扫到较大值时学到的图结构。这些图代表了关于这些蛋白连通性的一系列**合理假设**。

把它与图 26.17(b) 学到的 **DAG** 比较是有意义的：DAG 的优势是能方便地建模**干预**的特性，但劣势是不能表达该生物通路中已知的**反馈回路** (见 Schmidt & Murphy, 2009 的讨论)。注意：我们展示了很多 UGM、却只展示了一个 DAG，这只是偶然；我们完全可以用 BIC 选出“最优”UGM，反过来也可以从后验中抽样、展示多个 DAG 结构。

26.7.3 GGM 结构的贝叶斯推断

虽然 Glasso 相当快，但它只给出**一个点估计**的结构，而且它**不是**模型选择一致的 (Meinshausen, 2005)，即便 $N \rightarrow \infty$ 也不能保证恢复真图。更理想的做法是把参数积分掉，在**图空间**做后验推断，即计算 $p(G | D)$ 。然后就能像对 DAG 那样，从后验中抽取摘要，比如**边的后验边际** $p(G_{ij} = 1 | D)$ 。本节讨论如何做到这一点。

注意，这与第 13 章的**变量选择**是类比的：13.2 节讲贝叶斯变量选择 (把回归权重积分掉，计算 $p(\gamma | D)$ 与包含概率 $p(\gamma_j = 1 | D)$)；13.3 节讲 ℓ_1 正则方法。这里也是这两条路，但顺序反过来讲。

若图是**可分解** (decomposable) 的，并且使用**共轭先验**，就能**闭式**计算边缘似然 (Dawid & Lauritzen, 1993)。而且可以高效识别一个图的**可分解邻居** (Thomas & Green, 2009)，即**合法的加/删边集合**。这意味着我们可以用相对高效的**随机局部搜索**来近似后验 (如 Giudici & Green, 1999; Armstrong et al., 2008; Scott & Carvalho, 2008)。

然而，若目标是**知识发现**，限制在可分解图上太严了，因为可分解图的数量远小于一般无向图的数量。

有少数工作研究了**不可分解**情形下的 GGM 结构贝叶斯推断 (如 Dellaportas et al., 2003; Wong et al., 2003; Jones et al., 2005)，但这类方法**无法扩展**到大模型，因为它们用的是对边缘似然的**昂贵蒙特卡罗近似** (Atay-Kayis & Massam, 2005)。 (Lenkoski & Dobra, 2008) 建议使用**拉普拉斯近似**。这需要在 **GWishart** 先验 (Roverato, 2002) 下计算 Ω 的 MAP 估计。 (Lenkoski & Dobra, 2008) 用**迭代比例缩放 (IPS)** (Speed & Kiiveri,

1986; Hara & Takimura, 2008) 来找众数。但这很慢, 因为它需要知道图的**最大团**, 而这一般是 NP-难的。

(Moghaddam et al., 2009) 提出了一个快得多的方法: 他们改造了 26.7.1 节的**基于梯度的方法**来求 MAP, 这些算法**不需要**知道图的团。再进一步, 通过只用**对角拉普拉斯近似**来加速——它比 BIC 更准确, 但开销基本相同。结合“不限于可分解图”的优点, 使得我们能用相当快的**随机搜索**近似 $p(G \mid D)$ 及其众数。就相近的计算成本而言, 这一方法在**预测精度与结构恢复上都显著优于**图形化 lasso。

26.7.4 用 copula 处理非高斯数据

Glasso 及其变体**本质上**局限于**联合高斯**的数据, 这是相当严苛的限制。幸运的是, 该方法可以相当简单地推广到**非高斯但仍连续**的数据。基本思路是: 为每个变量 j 估计一个**单变量单调变换** f_j , 使得变换后的数据**联合高斯**。若可行, 就称数据属于**非参数正态分布**或**nonparanormal 分布** (Liu et al., 2009); 它等价于**高斯 copula**族 (Klaassen & Wellner, 1997)。如何由各变量的经验分布函数估计 f_j 的细节见 (Liu et al., 2009)。数据变换后即可计算**相关矩阵**, 再按常规套用 glasso。可证在若干假设下, 该方法是原始分布 CI 结构的一致估计量 (Liu et al., 2009)。

通俗详解 (不依赖高深数学)

这是在讲什么?

- 我们要学一张**无向**的“高斯图”(节点是实值变量, 联合近似高斯)。
- 与 DAG 不同: **不用担心成环**; 但**似然不分解**, 导致“每改一条边就得从头拟合”, 无法像学 DAG 那样做轻量的贪心/MCMC 局部搜索。

1) 参数估计 (MLE) 怎么做?

- 目标函数: $\ell(\Omega) = \log \det \Omega - \text{tr}(S\Omega)$ 。
- 约束:
 - 图里**没边**就强制 $\Omega_{st} = 0$ (结构性零) ;
 - Ω 必须**正定**。
- 直觉: **有边**的成对协方差 Σ_{st} 要“贴近数据”(恰等于样本协方差), **没边**的成对精度 $\Omega_{st} = 0$ 。因此 MLE 其实在做**正定矩阵补全**: 保留样本协方差里能保留的条目, 其它位置补齐到最像真的那份 Σ 。

2) 学结构: 图形化 lasso (Glasso)

- 利用“**精度矩阵零** \leftrightarrow **图中无边**”的——对应, 直接对 Ω 加 ℓ_1 惩罚:

$$J = -\log \det \Omega + \text{tr}(S\Omega) + \lambda \|\Omega\|_1.$$

- λ 越大， Ω 越稀疏，图越稀疏。
- 优点：凸优化、速度快；缺点：**点估计**、且理论上**不一致**（样本无限也不一定能还原真图）。

3) 贝叶斯法：直接在“图空间”做后验

- 思路：把参数积分掉，算 $p(G | D)$ ，输出“边出现的后验概率”。
- **可分解图**：有闭式边缘似然，能做高效的局部搜索；但可分解的集合太小。
- **一般图**：传统蒙特卡罗近似太慢；改用**拉普拉斯近似**。
 - 过去的实现慢在要找最大团；
 - Moghaddam+2009：用梯度法求 MAP（不需要团）+对角拉普拉斯，既快又比 BIC 准 → 随机搜索可扩展，且在预测/结构恢复上优于 Glasso。

4) 非高斯怎么办？——nonparanormal / 高斯 copula

- 给每个变量做一个**单调变换**，让整体近似联高斯；
 - 变换后算相关矩阵，再跑 Glasso；
 - 在条件下能**一致**恢复原分布的 CI 结构。
-

要点小结

- **难点**：无向高斯图的似然**不分解**，常规模型搜索**代价高**。
- **MLE**：在“结构性零+正定”约束下最大化 $\log \det \Omega - \text{tr}(S\Omega)$ ；本质是做**正定矩阵补全**。
- **Glasso**： ℓ_1 惩罚精度矩阵，直接得到**稀疏图**；快，但**点估计且非一致**。
- **贝叶斯结构学习**：
 - 可分解图：闭式边缘似然 + 随机局部搜索；
 - 一般图：用（梯度 MAP + 对角拉普拉斯）近似 $p(G | D)$ ，在**精度与结构恢复上**优于 Glasso。
- **非高斯**：用**nonparanormal/高斯 copula**（单调变换 → 相关矩阵 → Glasso），在一定条件下**一致**。

记一句：“无向高斯图：似然不分解 → 直接学精度；Glasso 快但点估计，贝叶斯 + 拉普拉斯可给后验；非高斯用 copula 变个身。”

这一节主要在讲什么？我不太懂

没关系~这节(26.7)讲的核心就是：**怎么学“无向高斯图模型”(GGM)的结构**，也就是哪些变量之间**直接相关**(边)，哪些**条件独立**(没边)。重点在于方法、难点与取舍。

一句话版

- 在 GGM 里，“哪儿没边”就等于“**精度矩阵 Ω 的对应元素是 0**”。
- 学结构=想办法让 Ω 稀疏。常用两路：**频率学的 Glasso**(快、点估计)和**贝叶斯方法**(给后验概率、更稳但要近似)。
- 非高斯的连续数据？先做**单调变换**把它“拉”成接近高斯(nonparanormal / 高斯 copula)，再用同样套路。

为什么比学 DAG 又易又难？

- **易：**无向图不担心“有环没环”。
- **难：****其似然不分解**，加/删一条边不能只改一小块打分——常规“逐边试错”的局部搜索(贪心/MCMC)代价高。

三块内容(通俗版)

1) 参数估计 (先会配料，再谈做菜)

- 目标函数(别纠结符号，直观就好)：

$$\ell(\Omega) = \log \det \Omega - \text{tr}(S\Omega)$$
 约束：
 - **结构性零**：没边 $\Rightarrow \Omega_{st} = 0$ ；
 - **正定**： Ω 必须是合法的协方差逆。
- 直觉：MLE 相当于做**正定矩阵补全**：保留样本协方差里**有边**的条目，其他位置补齐到最像真的那份矩阵。

2) Graphical Lasso (Glasso)：快学稀疏结构

- **关键事实：** Ω 的零 = 图上没边。
- 于是直接惩罚 $\|\Omega\|_1$ ：

$$-\log \det \Omega + \text{tr}(S\Omega) + \lambda \|\Omega\|_1$$
 λ 大 \rightarrow 更稀疏 \rightarrow 边更少。
- **优点：**凸优化、坐标下降很快，适合高维。
- **局限：**只给一个图(点估计)，而且理论上**不保证一致恢复真图**。

3) 贝叶斯学结构：给“边”的后验概率

- 目标：算 $p(G | D)$ ，得到**“这条边存在的概率”**。
- **可分解图**：有**共轭先验+闭式边缘似然**，能做高效随机搜索；但类别太小。
- **一般图**：用**拉普拉斯近似**等技巧(更快的做法：梯度法求 MAP + 对角拉普拉斯)，**不需要找最大团**，可扩展；实践上在**预测与结构恢复**上常**优于 Glasso**(代价相近)。

4) 非高斯的连续数据: nonparanormal / 高斯 copula

- 给每个变量做一个单调变换 f_j , 让整体更接近联合高斯;
- 算相关矩阵 → 跑 Glasso;
- 在一定条件下能一致恢复原始分布的条件独立结构。

什么时候用什么? (小抄)

- 想要快速给出一张稀疏图: Glasso, 扫一条 λ 路径看结构如何变。
- 想量化不确定性/做边的置信度: 贝叶斯 (后验 $p(G_{ij} = 1 | D)$) , 用拉普拉斯近似一类的快速版。
- 数据不是高斯但连续: 先做 nonparanormal 变换, 再按上面两路。
- 样本很少、维度很高: 优先稀疏方法 (Glasso / 贝叶斯稀疏先验) , 别用满矩阵。

要点总结

- 结构 \leftrightarrow 精度: 无向边结构由精度矩阵的零刻画。
- 难点: 似然不分解, 传统逐边搜索贵。
- Glasso: 快、实用、点估计; λ 控制稀疏度。
- 贝叶斯: 给后验与边概率; 通过拉普拉斯/梯度法可扩展到一般图, 效果常优。
- 非高斯: 用高斯 copula / nonparanormal“拉直”再学图。

记一句: “学无向高斯图=让精度矩阵变稀疏; 快用 Glasso, 要不确定性用贝叶斯; 非高斯先做 copula 再上。”

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。

26.8 Learning undirected discrete graphical models
The problem of learning the structure for UGMs with discrete variables is harder than the Gaussian case, because computing the partition function $Z(\theta)$, which is needed for parameter estimation, has complexity comparable to computing the permanent of a matrix, which in general is intractable (Jerrum et al. 2004). By contrast, in the Gaussian case, computing Z only requires computing a matrix determinant, which is at most $O(V^3)$. Since stochastic local search is not tractable for general discrete UGMs, below we mention some possible alternative approaches that have been tried.

26.8.1 Graphical lasso for MRFs/CRFs

It is possible to extend the graphical lasso idea to the discrete MRF and CRF case. However, now there is a set of parameters associated with each edge in the graph, so we have to use the graph analog of group lasso (see Section 13.5.1). For example, consider a pairwise CRF with ternary nodes, and node and edge potentials given by

$$\psi_t(y_t, x) = \left(v^T t_1 x \ v^T t_2 x \ v^T t_3 x \right), \psi_{st}(y_s, y_t, x) = \left(w^T t_{11} x_w t_{12} x_w t_{13} x \ w^T t_{21} x_w t_{22} x_w t_{23} x \ w^T t_{31} x_w t_{32} x_w t_{33} x \right) \quad (26.71)$$

where we assume x begins with a constant 1 term, to account for the offset. (If x only contains 1, the CRF reduces to an MRF.) Note that we may choose to set some of the v_{tk} and w_{stjk} weights to 0, to ensure identifiability, although this can also be taken care of by the prior, as shown in Exercise 8.5.

To learn sparse structure, we can minimize the following objective:

$$J = -N \sum_{i=1}^N \log \psi_t(y_i, x_i, v) + \sum_{s=1}^S \sum_{t=s+1}^T \log \psi_{st}(y_s, y_t, x_i, w) + \lambda_1 \sum_{s=1}^S \sum_{t=s+1}^T \|w_{st}\|_p + \lambda_2 \sum_{t=1}^T \|v_t\|_2^2 \quad (26.72)$$

where $\|w_{st}\|_p$ is the p -norm; common choices are $p = 2$ or $p = \infty$, as explained in Section 13.5.1. This method of CRF structure learning was first suggested in (Schmidt et al. 2008). (The use of 1 regularization for learning the structure of binary MRFs was proposed in (Lee et al. 2006).)

Although this objective is convex, it can be costly to evaluate, since we need to perform inference to compute its gradient, as explained in Section 19.6.3 (this is true also for MRFs). We should therefore use an optimizer that does not make too many calls to the objective function or its gradient, such as the projected quasi-Newton method in (Schmidt et al. 2009). In addition, we can use approximate inference, such as convex belief propagation (Section 22.4.2), to compute an approximate objective and gradient more quickly. Another approach is to apply the group lasso penalty to the pseudo-likelihood discussed in Section 19.5.4. This is much faster, since inference is no longer required (Hoefling and Tibshirani 2009). Figure 26.19 shows the result of applying this procedure to the 20-newsgroup data, where y_{it} indicates the presence of word t in document i , and $x_i = 1$ (so the model is an MRF).

26.8.2 Thin junction trees

So far, we have been concerned with learning “sparse”

graphs, but these do not necessarily have low treewidth. For example, a $D \times D$ grid is sparse, but has treewidth $O(D)$. This means that the models we learn may be intractable to use for inference purposes, which defeats one of the two main reasons to learn graph structure in the first place (the other reason being "knowledge discovery"). There have been various attempts to learn graphical models with bounded treewidth (e.g., (Bach and Jordan 2001; Srebro 2001; Elidan and Gould 2008; Shahaf et al. 2009)), also known as thin junction trees, but the exact problem in general is hard.

An alternative approach is to learn a model with low circuit complexity (Gogate et al. 2010; Poon and Domingos 2011). Such models may have high treewidth, but they exploit contextspecific independence and determinism to enable fast exact inference (see e.g., (Darwiche 2009)).

已思考 13s

下面先给出逐字完整翻译（含公式、符号，不做任何省略），随后是通俗详解，最后给出要点小结。

翻译

26.8 学习无向离散图模型

对具有离散变量的 UGM（无向图模型）进行结构学习的问题比高斯情形更难，因为用于参数估计所必需的配分函数 $Z(\theta)$ 的计算，其复杂度和计算一个矩阵的 permanent 的复杂度相当，一般来说是不可 tractable 的 (Jerrum et al. 2004)。对比之下，在高斯情形中，计算 Z 只需要计算一个矩阵行列式，至多是 $O(V^3)$ 。

由于对一般的离散 UGM 来说，随机局部搜索是不可行的，下面我们提到一些已经被尝试过的可能的替代方法。

26.8.1 用于 MRF/CRF 的图形化 lasso

可以把图形化 lasso 的思想扩展到离散的 MRF 和 CRF 情形。不过，此时图中每条边都对应一组参数，所以我们必须使用图上的 group lasso (见第 13.5.1 节) 的类比。例如，考虑一个成对 (pairwise) CRF，结点是三元的，其结点势与边势如下：

$$\psi_t(y_t, x) = \begin{pmatrix} v_{t1}^\top x \\ v_{t2}^\top x \\ v_{t3}^\top x \end{pmatrix}, \quad \psi_{st}(y_s, y_t, x) = \begin{pmatrix} w_{st,11}^\top x & w_{st,12}^\top x & w_{st,13}^\top x \\ w_{st,21}^\top x & w_{st,22}^\top x & w_{st,23}^\top x \\ w_{st,31}^\top x & w_{st,32}^\top x & w_{st,33}^\top x \end{pmatrix} \quad (26.71)$$

其中我们假定 x 的首分量为常数 1，用来表示偏置。（如果 x 只包含 1，这个 CRF 退化为一个 MRF。）注意我们可以选择把某些 v_{tk} 与 $w_{st,jk}$ 的权重设为 0，以保证可识别性，尽管这也可以由先验来处理，见习题 8.5。

为了学习稀疏结构，我们可以最小化下面的目标：

$$J = - \sum_{i=1}^N \sum_t \log \psi_t(y_{it}, x_i, v_t) + \sum_{s=1}^V \sum_{t=s+1}^V \log \psi_{st}(y_{is}, y_{it}, x_i, w_{st}) + \lambda_1 \sum_{s=1}^V \sum_{t=s+1}^V \|w_{st}\|_p^p + \lambda_2 \sum_{t=1}^V \|v_t\|_2^2 \quad (26.72)$$

其中 $\|w_{st}\|_p$ 是 p -范数；常见选择是 $p = 2$ 或 $p = \infty$ ，如第 13.5.1 节所述。这种 CRF 结构学习方法最先由 (Schmidt et al. 2008) 提出。（把 ℓ_1 正则用于二元 MRF 的结构学习由 (Lee et al. 2006) 提出。）

尽管这个目标是**凸的**，它的评估代价可能很高，因为我们需要进行**推断**来计算其梯度，如第 19.6.3 节所解释的（对 MRF 也是如此）。因此应使用不会对目标函数或其梯度进行太多次调用的优化器，比如 (Schmidt et al. 2009) 中的**投影拟牛顿法**。此外，我们可以使用**近似推断**，例如**凸信念传播**（第 22.4.2 节），以更快地计算近似的目 标与梯度。另一种做法是把 group lasso 惩罚作用在第 19.5.4 节讨论的**伪似然**上。这要快得多，因为不再需要推断 (Hoefling and Tibshirani 2009)。图 26.19 展示了将该过程应用于 20-newsgroup 数据时的结果，其中 y_{it} 表示单词 t 是否在文档 i 中出现，且 $x_i = 1$ （因此该模型是一个 MRF）。

26.8.2 瘦结点树（薄结点树，Thin junction trees）

到目前为止，我们一直关注学习“**稀疏**”图，但这些图不一定具有**低树宽**（treewidth）。例如，一个 $D \times D$ 的网格是稀疏的，但其树宽是 $O(D)$ 。这意味着我们学到的模型可能在推断时不可 tractable，这会违背学习图结构的两个主要理由之一（另一个理由是“知识发现”）。人们提出过多种尝试来学习**有界树宽**的图模型（例如 (Bach and Jordan 2001; Srebro 2001; Elidan and Gould 2008; Shahaf et al. 2009)），也称为**瘦结点树**（thin junction trees），但一般来说精确地解决这个问题是困难的。

一种替代方法是学习**低电路复杂度**（circuit complexity）的模型 (Gogate et al. 2010; Poon and Domingos 2011)。这类模型可能具有较高的树宽，但它们利用**情景特异独立性**（context-specific independence）与**确定性**来实现**快速的精确推断**（参见例如 (Darwiche 2009)）。

通俗详解（尽量少数学）

这节讲什么？

是在说：无向且离散的图模型（MRF/CRF）想学“哪些变量直接相连”，比高斯情形难，**最主要卡点是**——一旦要估计参数或算梯度，就要用到**配分函数** $Z(\theta)$ ，而对离散模型 Z 的计算非常难（级别接近 permanent）。

1) 为什么离散 UGM 更难？

- 高斯 UGM：对数似然里需要 $\log \det \Sigma$ ，矩阵运算，复杂度 $O(V^3)$ 可承受。
- 离散 UGM：需要 $\log Z(\theta)$ ，它是对所有状态的指数和；一般图上这等价于很难的计数问题 → **全图推断**几乎避免不了。
- 结果：像学 DAG 时那样“加一条边就快速评估一下”的**局部搜索**在这里行不通——**每个候选结构**都要几乎从头来一次推断/拟合，太贵。

2) “图形化 lasso”思路如何迁移到离散 MRF/CRF？

- 在高斯 GGM 里，**精度矩阵的零** \leftrightarrow **图上没边**，用 ℓ_1 罚促稀疏。
- 离散 MRF/CRF 没有“精度矩阵”，但**每条边有一组相互作用参数**（比如三元结点时是 3×3 个权重向量）。
- 因此对**每条边的一组参数**整体用 group lasso (ℓ_2 或 ℓ_∞ 组范数) 促整组为零 \Rightarrow **删边**。
- 目标函数（式 26.72）= “**（负）对数似然**” + “**组稀疏罚（边）**” + “**结点参数的二范数罚**”。
- **难点仍在梯度**：计算梯度要**边缘分布**，就得做**推断**。
 - 可用**投影拟牛顿**一类的少次调用优化器；
 - 用**凸信念传播等近似推断加速**；
 - 或者干脆把 group lasso 施加在**伪似然**上（逐点条件似然），**不需要全图推断**，速度大幅上升，代价是统计效率可能略降。

3) 不只“稀疏”，还要“可推断”：瘦结点树 & 低电路复杂度

- “稀疏”不等于“**低树宽**”。有些稀疏图（如网格）树宽仍大，精确推断依然贵。
- 目标：**受控的结构复杂度**，比如**树宽有界**（thin junction trees）。完全精确地学“最优低树宽结构”本身是难题，但有一系列近似/启发式工作。
- 另一条路：**学低电路复杂度**的表示（如概率电路、SPN 等）：
 - 允许图的树宽表面上不小；
 - 但用**情景特异独立性**（某些取值下很多因素互不相关）和**确定性**（某些势能为 0/1）做**结构压缩**，从而实现**快的精确推断**。

要点小结

- **核心难点：**离散 UGM 需要 $\log Z(\theta)$, 一般图上推断/计数极难 \Rightarrow 常规“逐边试错”的结构搜索不可行。
- **MRF/CRF 结构稀疏化：**对每条边的**整组参数**用 group lasso (ℓ_2/ℓ_∞ 组范数) 做零化；优化是凸的，但**每步要推断**。
 - 加速：投影拟牛顿、**凸 BP** 近似、或把组罚加在**伪似然**上（免推断，快）。
- **可用性（推断可行）同样重要：**
 - 仅“稀疏”不保证**低树宽**；
 - 目标可转向**薄结点树**（有界树宽）或**低电路复杂度**的模型，用**情景特异独立/确定性**达成**快速精确推断**。

记一句：“**离散无向图难在 Z ；学结构靠 group lasso (最好用**伪似然/近似推断**)；不仅要稀疏，更要推断快——靠**低树宽或低电路复杂度**。**”

这一节主要在讲什么？我不太懂

已思考 5s

不慌~这节 (26.8) 说的是：**怎么学“无向、离散”的图模型** (MRF/CRF) 的结构，为什么它比高斯情形难，以及常用的“折中”办法。

一句话版

难点在于**配分函数 $Z(\theta)$** 太难算（一般图上等同很难的计数问题），所以不能像学 DAG 或高斯图那样用轻量的局部搜索；因此要用**正则化、近似目标或限制结构复杂度**的策略。

主要内容

- **为什么更难：**离散 UGM 的似然要算 Z ，几乎每评估一次结构都得跑一次“全图推断”，开销爆炸；而高斯情形只需求行列式，便宜得多。
- **结构稀疏化 (像 Glasso 的思路)：**
 - 没有“精度矩阵零=没边”的捷径了，于是对**每条边的一整组相互作用参数**用 group lasso (ℓ_2 或 ℓ_∞ 组范数) \rightarrow 整组变零 = 删边。
 - 但**梯度要推断才拿得到** \Rightarrow 用**投影拟牛顿**这类少调用优化器；或用**近似推断**（如**凸 BP**）加速；
 - 更快的折中：把 group lasso 加在**伪似然**上（不做全图推断，速度大幅提升，统计效率略降）。
- **不仅要“稀疏”，还要“可推断”：**

- 稀疏≠低树宽（如网格很稀疏但树宽大），推断仍可能很慢。
- 方向一：学**薄结点树**（有界树宽）的模型——精确做最优一般很难，但有诸多近似/启发式。
- 方向二：学**低电路复杂度**的表示（如概率电路/SPN）：即使表面树宽高，也利用**情景特异独立与确定性**实现快速精确推断。

你该记住的

- 离散 UGM 难在 Z ：评估结构就要推断。
- 学**结构常用 group lasso**：整组为零=删边；可用近似推断或伪似然来降本。
- 模型**可用性很重要**：别只追稀疏，还要关注**树宽/推断复杂度**；必要时改用**薄结点树**或**低电路复杂度**的模型。

基于本对话里我贴过的整章原文，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 26s

下面基于你贴过的整章 (26.1–26.8) 内容，给出三个部分：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，含可选 Mermaid 图) ③本章总结。

① 知识点总结 (分点 | 关键词 | 用途)

- 26.1 结构学习的目标与难度
 - 关键词： $p(G|D)$ 、知识发现、密度估计、边后验 $p(G_{st=1}|D)$ 、MAP 图、组合爆炸
 - 用途：明确两类任务——**画图找关系** (知识发现) 与**建模做预测** (密度估计)；决定输出是“**边置信度图**”还是“**单个最优结构**”。
- 26.2 “快而脏”的可视化方法
 - 26.2.1 Relevance Network
 - 关键词：互信息 MI、相关系数、协方差图
 - 用途：**极速可视化**变量两两关联；但过密、含传递依赖。
 - 26.2.2 Dependency Network
 - 关键词：稀疏全条件 $p(x_t|x-t)$ 、11 回归/逻辑回归/树、马尔可夫毯、Gibbs 近似

- 用途：更稀疏直观的可视化；可做缺失值填补/初始化复杂方法；但不保证有效联合分布（推断仅近似）。
- 26.3 学习树结构
 - 关键词：有向树=无向树（等价）、Chow-Liu、MST(互信息权重)、MAP 森林、最大分支、树模型混合、矩阵树定理
 - 用途：在可精确推断前提下，快速、全局最优地学到结构（尤其高维离散）；混合树提升表达力仍保高效。
- 26.4 学习 DAG (无隐藏、数据完全)
 - 关键词：马尔可夫等价/PDAG、BDe/BDeu 分数、Dirichlet 先验、等价样本量 α 、K2(已知顺序)、BIC 近似、动态规划≤16点、贪心爬山/边翻转、MH/顺序空间 MCMC(Collapsed)
 - 用途：可度量与可比较的打分（边缘似然）；在中小规模上近似或精确找MAP，或估计边后验用于知识发现。
- 26.5 含潜变量/缺失的 DAG 学习
 - 关键词： $p(D|G)=\sum_h \prod_j p(d_j|g_j)$ 、BIC/CS/VBEM（VBEM > CS > BIC）、结构 EM(SEM)、潜树/层次潜类/凝聚聚类、内部隐结点、Rephil(noisy-OR, ICM)、SEM(线性高斯、相关误差、反馈)
 - 用途：当有缺失/潜因时，用更准的证据近似（推荐 VBEM/CS）+ 结构EM高效搜索；自动发现隐因素；工业级大规模学习与快速推断套路；用 SEM连接到社会科学/经济学。
- 26.6 学习因果 DAG
 - 关键词：do-演算、图外科手术(切入边)、因果马尔可夫/因果充分性、辛普森悖论、忠实性/一致性(GES)、IDA 下界、干预计数修正、增广DAG
 - 用途：回答“做了会怎样”；用干预/观察混合数据定向边；用 IDA 在仅观测下给因果效应下界；指导实验设计（主动干预）。
- 26.7 无向高斯图 (GGM)
 - 关键词： $\Omega = \Sigma^{-1}$ 、协方差选择、正定补全、Graphical Lasso(ℓ_1)、贝叶斯(拉普拉斯近似、非可分解)、nonparanormal/高斯copula
 - 用途：在连续高维场景学条件独立结构；Glasso 快速点估计；贝叶斯+拉普拉斯给后验与更好结构恢复；copula 适配非高斯连续数据。
- 26.8 无向离散图 (MRF/CRF)
 - 关键词：配分函数 Z 难、Group Lasso(边为组)、伪似然、薄结点树(低树宽)、低电路复杂度/概率电路/SPN
 - 用途：为离散 UGM做稀疏结构学习（组稀疏、可用伪似然加速）；同时关注推断可行性（低树宽或电路化表示实现快的精确推断）。

② 子章节关联 (依赖/对比/递进)

关系句 (文字版)

- 26.1 → 26.2/26.3/26.4/26.7/26.8 (递进) : 从目标与难度出发, 先给**可视化捷径** (26.2), 再到**可精确的特例树** (26.3), 再到**一般 DAG** (26.4), 并分别补充**无向高斯/离散路线** (26.7/26.8)。
- 26.2 ↔ 26.3/26.4 (对比) : 相关/依赖网络快但不严谨; 树/DAG 是**正规概率模型**, 可用于预测与严格评估。
- 26.3 → 26.4 (依赖/初始化) : Chow–Liu 的树常作为 DAG 搜索的初始结构或**参考边集**。
- 26.4 → 26.5 (递进) : 从**完全可见到含缺失/潜因**; 打分从 BDe/BDeu 扩展到 CS/VBEM; 搜索从贪心扩展到**结构EM**。
- 26.5 → 26.6 (对比) : 26.5 解决“**相关结构+潜因**”; 26.6 引入“**因果语义与干预**”, 强调 `do()` 与**实验数据**的重要性。
- 26.4/26.5 ↔ 26.7/26.8 (并行路线) : 前者是**有向家族** (DAG/因果), 后者是**无向家族** (GGM/MRF/CRF), 分别覆盖**连续与离散**数据。
- 26.7 ↔ 26.8 (对比) : 高斯 (行列式易算) → Glasso/**贝叶斯**; 离散 (z 难) → Group Lasso/**伪似然** + **低树宽/电路**以保推断可行。
- 26.5(Rephil)/26.7(copula)/26.8(**伪似然/电路**) (工程化) : 均体现**大规模与推断效率**优先的设计取舍。

Mermaid 关系图 (可选查看)

```

graph TD
    A[26.1 目标&难度] --> B[26.2 快速可视化<br/>Relevance/Dependency]
    A --> C[26.3 树结构<br/>Chow-Liu/MST/森林/混合]
    A --> D[26.4 DAG(完全可见)<br/>BDe/BDeu/K2/贪心/DP]
    A --> E[26.5 潜因/缺失<br/>CS/VBEM/结构EM/潜树/SEM/Rephil]
    A --> F[26.6 因果DAG<br/>do-演算/IDA/干预学习]
    A --> G[26.7 无向高斯GGM<br/>Glasso/贝叶斯/Laplace/Copula]
    A --> H[26.8 无向离散UGM<br/>Group Lasso/伪似然/低树宽/电路]

    C -. 初始/边候选 .-> D
    B -. 快速但不严谨 .-> D
    G <--> H
    C -. 初始/边候选 .-> F
    D -. 快速但不严谨 .-> F
    F -. 快速但不严谨 .-> H
    E -. 快速但不严谨 .-> H
    G -. 并行路线 .-> H
  
```

③ 本章总结 (策略与取舍)

- **先定目标:**
 - **知识发现/可视化:** 可用 Relevance/Dependency 网络 (快), 或输出**边后验热度图** (更严谨)。
 - **密度估计/预测:** 优先用**正规概率模型** (树、DAG、GGM、MRF/CRF), 并关注**推断复杂度** (树宽/电路化)。
- **按数据类型选模型:**

- **离散**: 树/混合树、DAG (BDeu/贪心/顺序MCMC) 、MRF/CRF (Group Lasso+伪似然; 必要时追求低树宽或概率电路)。
- **连续**: GGM (Glasso 快; 贝叶斯+拉普拉斯做不确定性与更好恢复), 非高斯先做 nonparanormal/copula。
- **规模/可行性**:
 - **小中规模**: 可用 DP/精确算法 (树、部分 DAG 统计量)。
 - **大规模**: 用稀疏正则 (l1/group) 、**近似证据** (CS/VBEM) 、**结构EM**、**伪似然**、**概率电路**, 并以**推断时间为硬约束优化结构**。
- **潜因与因果**:
 - 潜因: VBEM/CS + **结构EM** 找到更简约、解释更好的模型; 潜树/层次潜类/内部隐点可自动提炼主题/概念。
 - 因果: `do()` ≠ 观察; **辛普森悖论**提醒“先画因果图, 再决定调不调整”; 仅观测最多到 PDAG, 要**干预数据**来定向; IDA在仅观测时可给**因果效应下界**。
- **实践备忘 (决策小抄)** :
 1. **先用树 (Chow-Liu) **打底: 可视化 & 作为 DAG 搜索的初始化。
 2. **DAG (可见)** : BDeu+贪心/顺序MCMC, 必要时**边后验**而非单一 MAP。
 3. **潜因/缺失**: 优先 VBEM 或 CS; 用 **结构EM** 提速搜索。
 4. **因果**: 能做干预就做; 不能做时用 IDA 给保守结论。
 5. **连续高维**: 先 Glasso 扫 λ , 再考虑**贝叶斯+拉普拉斯**做不确定性与结构复核; 非高斯用 copula。
 6. **离散UGM**: Group Lasso+**伪似然**; 若推断吃力, 转向**低树宽**或**概率电路**表示。

一句话收束: 根据目标 (看关系 vs 做预测/因果) 与数据类型 (离散/连续/非高斯), 在“**精确性—可扩展—推断可行**”三角里选点: 树/混合树图快稳, DAG 用 BDeu/近似搜索, 潜因靠 VBEM+结构EM, 因果要 `do()` 与干预, UGM 连续走 Glasso/贝叶斯, 离散走组稀疏+伪似然/低树宽/电路。

请用通俗易懂的话总结本章, 使我能够理解

已思考 8s

好的, 我把第 26 章用“人话”串成一条主线, 帮你把“这章到底在说什么、什么时候用什么方法”讲清楚。

大图景：我们要学的不是参数，而是“谁跟谁有直接关系的结构”

- 图模型把一堆变量之间的“直接影响/直接相关”画成边。
 - 结构学习就是从数据里把这张图找出来。
 - 有两类目标：
 1. 知识发现：想看清谁跟谁有关（便于可视化/解释）。
 2. 密度估计/预测：想要一个能算概率/做推断的完整模型。
 - 难点：候选图的数量呈爆炸式增长，很多精确算法只能用在很小的图或特殊结构上。
-

两个“快速画关系图”的办法（可视化优先）

- 相关网络 (Relevance Network)：看两两互信息/相关，超过门槛就连边。优点快；缺点太密，会把“间接关系”也连上。
 - 依赖网络 (Dependency Network)：对每个变量做稀疏的“被谁一起预测”（马尔可夫毯），再把这些边拼起来。更稀疏、可读性好，但不保证来自一个严格的联合分布（推断只作参考/填缺用）。
-

特例：树结构最友好（快、准、还能精确推断）

- Chow-Liu：把每条边的权重设为互信息，求最大生成树，就得到全局最优的树形图。
 - 可延伸到森林（有些边不要）或树的混合（提高表达力，但推断仍高效）。
-

一般的有向图 (DAG)：从“分数 + 搜索”入手

- 方向有时辨不清：多张 DAG 可能表达同样的独立关系（马尔可夫等价），所以最多学到一个等价类 (PDAG)。
 - 怎么打分：用边缘似然 (BDe/BDeu)，它会惩罚太复杂的图；或用 BIC 近似。
 - 怎么搜：
 - 小图可用动态规划 (~16 个点以内)；
 - 常用 贪心爬山（加/删/翻转一条边），配合多次随机重启；
 - 还可以用 MCMC 在图空间或顺序空间抽样，统计“边出现的概率”（更适合知识发现）。
-

有缺失/潜变量时该怎么搞？

- 真正的证据要对**隐藏变量求和**，很难算。于是用**近似证据**做打分：
 - BIC（最快，往往偏简单）；
 - CS (Cheeseman-Stutz) （更准）；
 - VBEM (变分贝叶斯 EM) （一般最稳最准）。
 - **结构 EM**: 先用当前模型**填期望数据**，批量评估周边候选图，选最好的再重新估计参数——比“每个候选都完整拟合”快很多。
 - **怎么“发现”潜因**: 看一群变量是否**成团/紧密相连**，加一个潜节点去解释它们；或学**潜树/层次潜类**；也有把隐点加在内部结点的方法。
 - 工程例子：**Google Rephil**——海量词与潜概念、noisy-OR、剪枝+ICM，能在毫秒级推断。
-

因果 DAG：回答“做了会怎样”

- 关键是 **do()**: $do(X = x)$ 是“把 X 设成 x ”，跟“看见 $X=x$ ”完全不同。
 - 算法上用**图外科手术**: 把被干预结点的**入边切掉**，在“残缺图”上做普通推断。
 - **辛普森悖论**说明：只看分组/合并的相关性会反转结论；要根据**因果图**决定“该不该对一个变量做调整”。
 - **学因果结构**：
 - 只有观测数据时，最多到 PDAG，可用 GES 等；
 - 想量化效应但图不唯一，可用 IDA 给出某因果效应的**保守下界**；
 - 有**干预数据**就容易定向：被设定的样本别算进那个节点的参数统计；干预不完美就加**干预结点**（增广 DAG）建模。
-

无向图两条路：连续 vs 离散

连续 (高斯 GGM)

- 边是否存在 \leftrightarrow **精度矩阵 Ω** 的对应元素是否为 0。
- Graphical Lasso (ℓ_1) : 直接让 Ω 稀疏，速度快（点估计）。
- 想要**不确定性/更好结构恢复**：用**贝叶斯 + 拉普拉斯近似**在图空间做后验。
- 数据非高斯？先做**单调变换** (nonparanormal / 高斯 copula) 再学图。

离散 (MRF/CRF)

- 难在配分函数 Z 很难算。
- 把每条边的一整组相互作用参数当成一个“组”，用 **Group Lasso 做组稀疏**（整组为零=删边）。
- 为了快：把罚加在**伪似然**上（不用全图推断），或用**近似推断**算梯度。

- 仅“稀疏”不保证推断快，还要关注**树宽**；可以追求**薄结点树**（低树宽），或改用**概率电路/SPN**这类**低电路复杂度**的表示实现**快的精确推断**。
-

实操小抄（按场景选法）

- 先看关系**（快速可视化）：相关网络 / 依赖网络。
 - 要能精确推断、变量多**：先上 Chow–Liu (树/森林)；表达力不够再用**混合树**。
 - DAG (无缺失) 做预测**：BDeu 打分 + 贪心/顺序 MCMC；小规模可算边后验。
 - 有缺失/潜因**：VBEM 或 CS 做打分，配**结构 EM** 搜索；想找潜因就尝试潜树/层次潜类。
 - 要讨论因果**：明确 do()、画因果图；仅观测最多到 PDAG，用 IDA 给下界；能做实验就做（增广 DAG 处理不完美干预）。
 - 连续高维**：Graphical Lasso 扫 λ ；需要后验再上贝叶斯+拉普拉斯；非高斯做 copula 变换。
 - 离散无向**：Group Lasso + 伪似然；推断吃力就控制树宽或改成概率电路。
-

记住这几条就够用了

- 目标先行**：可视化 vs 预测/推断 vs 因果。
- 树是王牌**：学得快、推断准，是很好的基线/初始化。
- DAG 用“分数+搜索”**：BDeu/BIC/VBEM/CS 选其一，贪心或 MCMC 搜。
- 潜因要谨慎**：用 VBEM/CS + 结构 EM；不要过度解读边方向（等价类存在）。
- 因果要看 do()**：观察≠干预；需要实验或用 IDA 给保守结论。
- 无向图**：连续走 Glasso/贝叶斯，离散走 Group Lasso/伪似然；别忘了**推断可行性**（**树宽/电路**）。

这样把工具箱分层后，你就知道：面对不同数据与目标，本章给了你一套从“**快看关系**”到“**能推断/能因果**”的渐进路线，以及每一步的可选算法与权衡点。