【20221029】-【20221225】-【20230215】-【20230317】

对《PRML》**8 GRAPHICAL MODELS** 的梳理与总结。

Graphical models

In a probabilistic graphical model, each node represents a random variable (or group of random variables), and the links express probabilistic relationships between these variables. 概率图模型主要有两大类：

1. *Bayesian networks*, also known as *directed graphical models*，in which the links of the graphs have a particular directionality indicated by arrows.
2. *Markov random fields*, also known as *undirected graphical models*, in which the links do not carry arrows and have no directional significance.

它们的特点: Directed graphs are useful for expressing causal relationships between random variables, whereas undirected graphs are better suited to expressing soft constraints between random variables.

而为了推断问题，我们最后又引入了因子图: For the purposes of solving inference problems, it is often convenient to convert both directed and undirected graphs into a different representation called a *factor graph*.

1 概率有向图模型

1.1 Bayesian network

对应 **8.1 Bayesian network** 节，介绍 Bayesian network 是什么，主要是有向图与概率分布 decomposition 的相互表示。

从全连接到部分连接：So far, we have worked with completely general joint distributions, so that the decompositions, and their representations as fully connected graphs, will be applicable to any choice of distribution. As we shall see shortly, it is the *absence* of links in the graph that conveys interesting information about the properties of the class of distributions that the graph represents.

概率有向图模型的定义，即 *DAGs* (*directed acyclic graphs*): The directed graphs that we are considering are subject to an important restriction namely that there must be no *directed cycles* (是否构成环需要考虑方向性), in other words there are no closed paths within the graph such that we can move from node to node along links **following the direction of the arrows** and end up back at the starting node. This is equivalent to the statement that there exists an ordering of the nodes such that there are no links that go from any node to any lower numbered node (等价于结点间存在一个排序，排在后面的结点(编号大)不存在指向排在前面的结点(编号小)的边). 我们强调，概率有向图是无环的，但通常提起它时会省略"无环"二字。此外，这里在判断是否构成一个环时还要考虑方向，若判断环时不考虑方向，则得到的有向无环图的结构就是后面会介绍的有向树或多树 (polytree)，是这里 DAGs 的子集。

8.1.1 Example: Polynomial regression 小节通过 Polynomial regression 的例子来介绍概率有向图和其表示方法 (也就是画法)：

1. we shall adopt the convention that random variables will be denoted by **open circles**, and deterministic parameters will be denoted by **smaller solid circles**.
2. In a graphical model, we will denote such *observed variables* by **shading the corresponding nodes**.
3. the plate notation (即盘式标记法) : 一种简化表示，the box labelled N 表示 box 中的量有 N 份。

一个完整的例子见 Figure 8.6.

8.1.2 Generative models 小节讨论了 (1) 概率有向图模型的采样 (*ancestral sampling*)，(2) generative models 以及 (3) 隐变量的意义。对 (2), (3) 的讨论事实上并不局限于有向图模型。

generative models 的含义：The graphical model captures the *causal* process by which the observed data was generated. For this reason, such models are often called generative models. $p(y|x)$ 就不是 generative model，because there is no probability distribution associated with the input variable x, and so it is not possible to generate synthetic data points from this model. We could make it generative by introducing a suitable prior distribution $p(x)$, at the expense of a more complex model.

隐变量的意义：

1. The primary role of the latent variables is to allow a complicated distribution over the observed variables to be represented in terms of a model constructed from simpler (typically exponential family) conditional distributions. 即引入隐变量可以基于简单的概率分布构建对观测变量的复杂的概率分布。
2. The hidden variables in a probabilistic model need not, however, have any explicit physical interpretation but may be introduced simply to allow a more complex joint distribution to be constructed from simpler components. 即隐变量并不一定要有物理意义，其引入可以只是为了构建复杂的概率分布。

8.1.3 Discrete variables 小节介绍离散随机变量的概率有向图模型，表明：

1. From a graphical perspective, we have reduced the number of parameters by dropping links in the graph, at the expense of having a restricted class of distributions. 进一步，An alternative way to reduce the number of independent parameters in a model is by *sharing* parameters (also known as *tying* of parameters).
2. Another way of controlling the exponential growth in the number of parameters in models of discrete variables is to use parameterized models for the conditional distributions instead of complete tables of conditional probability values.

上一小节介绍了离散随机变量的例子，**8.1.4 Linear-Gaussian models** 小节就以高斯分布为例介绍连续随机变量的概率有向图模型。此外，本小节还提到了一个重要的思路，即 hierarchical Bayesian model，它实际上就是说参数与超参数的辩证关系：可将超参降格为参数，引入先验分布，而先验分布又可能存在新的超参；对新的超参又可以重复上述步骤，无限套娃，直到我们想要的那一层。

1.2 Conditional independence

8.2 conditional independence 节，介绍如何根据概率有向图得到分布的条件独立性，即 d-separation criterion.

An important and elegant feature of graphical models is that conditional independence properties of the joint distribution can be read directly from the graph without having to perform any analytical manipulations. The general framework for achieving this is called *d-separation*, where the 'd' stands for 'directed'.

首先，**8.2.1 Three example graphs** 给出了三个基础 case，即 (1) tail to tail node; (2) head to tail node; (3) head to head node (包括存在 descendant 的情况) 条件独立性的判断：In summary, a tail-to-tail node or a head-to-tail node leaves a path unblocked unless it is observed in which case it blocks the path. By contrast, a head-to-head node **blocks** a path if it is unobserved, but once the node, and/or at least one of its descendants, is observed the path becomes unblocked (这也总结得太好了!).

> 注：path 被 block (阻塞) 意味着 path 连接的另外两个 node 此时相互独立；反之 unblock (连通) 则不具备相互独立性。此外，这里说的是条件独立性，但“条件项”为空集也是可以的，比如 head to head node 的情况。

head to head node 的条件独立性结论也称 explain away (相消解释，直译实际上就是"解释")，即在 c 已知的情况下，条件独立性并不成立 $p(a,b|c) \neq p(a|c)p(b|c)$ (或等价地 $p(a|c) \neq p(a|c,b)$)，反而在 c 未知的情况下，独立性才成立。事实上，可从括号内的式子一窥 explain away 称呼的来源：假设 a, b 的发生都会诱发 c 的发生，那么在已知 c 发生的情况下，若还已知 b 发生了，那么 b 的发生在一定程度上就会解释 (explain away) c 的发生，此时倒推 a 发生的概率，就不会那么高了，即 $p(a=1|c=1,b=1) < p(a=1|c=1)$.

8.2.2 D-separation 小节则 give a general statement of the d-separation property for directed graphs. 总结如下：

Consider a general directed graph in which A, B, and C are arbitrary nonintersecting sets of nodes (whose union may be smaller than the complete set of nodes in the graph). We wish to ascertain whether a particular conditional independence statement $A \perp\!\!\!\perp B|C$ is implied by a given directed acyclic graph. To do so, we consider **all possible paths** from any node in A to any node in B. Any such path is said to be **blocked** if it includes a node such that either

(a) the arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set C, or

(b) the arrows meet head-to-head at the node, and neither the node, nor any of its descendants, is in the set C.

If all paths are blocked, then A is said to be **d-separated** from B by C, and the joint distribution over all of the variables in the graph will satisfy $A \perp\!\!\!\perp B|C$.

注意：

1. 我们可能会将上述 d-seperation 规则视为一个判断独立性和依赖性的二分类器，但需要注意的是，对于给定的有向图，我们只承认图中能读出来的独立性：若 A, B 之间的所有 path 没有被 blocked，即得不到 $A \perp\!\!\!\perp B|C$，但我们并不会说给定 C 时，A, B 一定相互依赖，而只会认为它们之间的独立性待定。也就是说，在满足概率图所包含的所有条件独立性的情况下，某个概率分布可以包含更多的条件独立性，该分布仍然是不违背该有向图的，这一点在笔记的 **1.3 小节**体现得很明显。
2. 记除了 A, B, C 外剩余的 nodes 为 D，则完整的联合概率分布为 $p(A, B, C, D)$，若条件独立性成立，即 $p(A, B|C) = p(A|C)p(B|C)$，可以看到 D 并未出现在式子中，表示未知的，未被观测的 (not observed)，在数学上就是积分被积掉了，即 $p(A, B|C) = \int p(A, B, D|C)\mathrm{d}D$，对 $p(A|C), p(B|C)$ 同理。

马尔可夫毯：The Markov blanket of a node x_i comprises the set of parents, children and co-parents of the node. It has the property that the conditional distribution of x_i, conditioned on all the remaining variables in the graph, is dependent only on the variables in the Markov blanket. 若记 x_i 的马尔可夫毯为 M_i，则：

$$p(\mathrm{x}_i|\mathrm{x}_{\{j\neq i\}}) = p(\mathrm{x}_i|\mathrm{M}_i)$$

1.3 Graph: decomposition vs d-separation criterion

8.2.2 小节的部分内容，给出了概率有向图模型的理论基础，即 decomposition 与 d-separation criterion 的等价性

We have seen that a particular directed graph represents a specific decomposition of a joint probability distribution into a product of conditional probabilities (也就是说，给定概率有向图，我们可以得到对应的 decomposition $p(\mathbf{x}) = \prod_{i=1}^{K} p(x_k \mid \mathrm{pa}_k)$，即式 (8.5)). The graph also expresses a set of conditional independence statements obtained through the d-separation criterion (另一方面，概率有向图可以基于 d-separation criterion 得到一系列的条件独立性的论断), and the d-separation theorem (注意，是 theorem 而不是 criterion，这是两个不同的东西，我们通常说的 d-separation 指的是 criterion) is really an expression of the equivalence of these two properties (最后，the d-separation theorem 告诉我们，所有满足 decomposition 的概率分布所组成的集合等价于所有满足条件独立性论断的概率分布所组成的集合，即二者所表示的概率分布族是等价的。由此，对任意的 $p(\mathbf{x})$，给定其 decomposition $p(\mathbf{x}) = \prod_{i=1}^{K} p(x_k \mid \mathrm{pa}_k)$，我们可画出相应的有向图，并使用 d-separation criterion 直接在图中分析其条件独立性，这便是概率有向图模型能 work 的理论基础). 注意，the d-separation theorem 只告诉我们等价，但并未证明，按文中所说，a formal proof can be found in Lauritzen (1996).

> We can view a graphical model (in this case a directed graph) as a filter in which a probability distribution $p(\mathbf{x})$ is allowed through the filter if, and only if, it satisfifies the directed factorization property (8.5). The set of all possible probability distributions $p(\mathbf{x})$ that pass through the filter is denoted $\mathcal{DF}$, for *directed factorization*. (filter 1)
>
> We can alternatively use the graph to filter distributions according to whether they respect all of the conditional independencies implied by the d-separation properties of the graph. (filter 2)

> The d-separation theorem says that it is the same set of distributions *DF* that will be allowed through this second kind of fifilter. (filter 1 等价于 filter 2)

Note that for any given graph, the set of distributions $\mathcal{DF}$ will include any distributions that have additional independence properties beyond those described by the graph. For instance, a fully factorized distribution will always be passed through the filter implied by any graph over the corresponding set of variables. (也就是笔记 **1.2 Conditional independence** 小节的注意点 1)

2 概率无向图模型

A *Markov random field*, also known as a *Markov network* or an *undirected graphical model*

概率无向图模型是先介绍的条件独立性，再介绍的分解。

2.1 Conditional independence

对应 **8.3.1 Conditional independence properties** 小节。

Testing for conditional independence in undirected graphs is simpler than in directed graphs. 概率无向图的条件独立性的判断更加简单。

类似于d-separation，我们有判断规则 graph separation: we consider the conditional independence property $A \perp\!\!\!\perp B|C$. To test whether this property is satisfified by a probability distribution defifined by a graph we consider all possible paths that connect nodes in set A to nodes in set B. If **all such paths** pass through **one or more nodes** in set C, then all such paths are 'blocked' and so the conditional independence property holds. However, if there is at least one such path that is not blocked, then the property does not necessarily hold, or more precisely there will exist at least some distributions corresponding to the graph that do not satisfy this conditional independence relation.

或者等价的更方便的判断方法：An alternative way to view the conditional independence test is to imagine removing all nodes in set C from the graph together with any links that connect to those nodes. We then ask if there exists a path that connects any node in A to any node in B. If there are no such paths, then the conditional independence property must hold.

无向图模型的马尔可夫毯：The Markov blanket for an undirected graph takes a particularly simple form, because a node will be conditionally independent of all other nodes conditioned only on the neighbouring nodes, as illustrated in Figure 8.28.

2.2 Factorization properties

对应 **8.3.2 Factorization properties** 小节前半部分。

引入团和最大团的概念：a *clique*, which is defifined as a subset of the nodes in a graph such that there exists a link between all pairs of nodes in the subset. In other words, the set of nodes in a clique is fully connected. Furthermore, a *maximal clique* is a clique such that it is not possible to include any other nodes from the graph in the set without it ceasing to be a clique.

概率分布的分解式：the joint distribution is written as a product of *potential functions* $\psi_C(\mathbf{x}_C)$ over the maximal cliques (当然也可以基于 cliques 进行分解，但基于 maximal cliques 显然更 general，它包含了基于 cliques 的情况) of the graph (也就是式 (8.39)):

$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \psi_C(\mathbf{x}_C)$$

partition function $Z = \sum_{\mathbf{x}} \prod_C \psi_C(\mathbf{x}_C)$, the presence of this normalization constant is one of the major limitations of undirected graphs.

In contrast to the factors in the joint distribution for a directed graph, the potentials in an undirected graph do not have a specifific probabilistic interpretation.

势函数的常用形式：

$$\psi_C(\mathbf{x}_C) = \exp\{-E(\mathbf{x}_C)\}$$

where $E(\mathbf{x}_C)$ is called an energy function (能量越小，对应概率越大). 对应的概率分布也称为 *Boltzmann distribution*.

8.3.3 Illustration: Image de-noising 则给出了一个图像去噪应用的举例，这个例子还挺有意思的，思路如下：$x_i \in \{-1, +1\}$ 为原始图片的像素点，未知；$y_i \in \{-1, +1\}$ 为有噪声的图片的像素点，可被观测。我们基于概率无向图模型对 x_i, y_i 的联合概率分布进行建模，当 x_i 的取值使概率模型的概率最大时，x_i 就作为去噪的结果输出。可以看到：

1. 这是一个无监督学习的问题；
2. x_i 更像是参数，通过优化直接得到 x_i 的值，优化算法有 3 种：(1) *iterated conditional modes*, or *ICM* (Kittler and Foglein, 1984), which is simply an application of coordinate-wise gradient ascent; (2) max-sum algorithm; (3) graph-cut algorithm.
3. 如何对 x_i, y_i 进行建模，即能量函数如何选取，就反映了构建者对问题的认知。这显然是求解问题的关键，若构建的模型无法很好地表达 x_i, y_i 之间的关系，去噪效果显然不会好。

2.3 Graph: factorization vs conditional independence

对应 **8.3.2 Factorization properties** 小节后半部分。

However, we have not made any formal connection between conditional independence and factorization for undirected graphs. To do so we need to restrict attention to potential functions $\psi_C(\mathbf{x}_C)$ that are strictly positive. To do this we again return to the concept of a graphical model as a fifilter,

filter 1: We can defifine $\mathcal{UI}$ to be the set of such distributions that are consistent with the set of conditional independence statements that can be read from the graph using graph separation.

filter 2: , we can defifine $\mathcal{UF}$ to be the set of such distributions that can be expressed as a factorization of the form (8.39) with respect to the maximal cliques of the graph.

filter 1 $\iff$ **filter 2**: The *Hammersley-Clifford* theorem (Clifford, 1990) states that the sets $\mathcal{UI}$ and $\mathcal{UF}$ are identical.

2.4 有向图模型 $\rightarrow$ 无向图模型

对应 **8.3.4 Relation to directed graphs** 小节的前半部分。

moralization (道德化)，转化后的无向图称之为 moral graph. 道德化的过程也很简单和直接：

- 图形上：Thus in general to convert a directed graph into an undirected graph, we first add additional undirected links between all pairs of parents for each node in the graph and then drop the arrows on the original links to give the moral graph. 将 parents 相互连接的这一过程就称为道德化 (this process of ‘marrying the parents’ has become known as *moralization*).
- 概率表达式上：Then we initialize all of the clique potentials of the moral graph to 1. We then take each conditional distribution factor in the original directed graph and multiply it into one of the clique potentials. 也就是先将所有团的势函数初始化为 1，再将有向图模型的条件概率分布划分到相应团的势函数中。Note that in all cases the partition function is given by $Z = 1$.

转化为无向图后，可能会丢失掉原有向图的一些条件独立性；但相比将有向图无脑转化为全连接的无向图，从而丢失掉所有的条件独立性，The process of moralization adds the fewest extra links and so retains the maximum number of independence properties.

> 此外，无论是有向图还是无向图，对于独立/依赖的判定结果 (使用 d-separation or grapha separation)，人为地，我们只承认独立性，而对依赖性，则认为可能成立也可能不成立，因此丢失掉的独立性并不是变成了确切的依赖关系，而是独立或依赖均有可能，均符合图模型所表示的概率分布空间。因此，道德图所表示的空间包含原有向图所表示的空间，最极端的情况就是全连接的无向图，所有的概率分布均符合其要求，无法得到确切的条件独立性论断，包含的信息最少。

道德化的应用场景：精确推断 (The process of converting a directed graph into an undirected graph plays an important role in exact inference techniques such as the *junction tree algorithm*).

Converting from an undirected to a directed representation is much less common and in general presents problems due to the normalization constraints. 一般不将无向图模型 $\rightarrow$ 有向图模型。

3 summary

对前面内容的归纳、总结与延伸。

3.1 概率图模型的脉络

概率图模型的两条等价路线：

1. decomposition：根据概率图写出联合概率分布的分解式（expressing the joint distribution $p(\mathbf{x})$ as a product of functions defined over sets of variables that are local to the graph），有向图的分解基于条件概率，无向图的分解基于最大团。
2. 条件独立性规则：根据人为设定的判定规则（有向图：d-separation；无向图：graph separation），从概率图中得到分布的条件独立性。

我们强调，判定规则是人为设定的。此外，给定概率图，对于需要判定的情况，根据规则，要么独立，要么依赖，不会存在第三种可能，也就是说，关于是否独立的命题空间被划分为独立与依赖两个不相交的子集，多一条独立性的命题就意味着少一条依赖性的命题，反之亦然。通常，我们会用一张概率图表示一类概率分布的集合，但这并不意味着它所表示的概率分布一定要满足图中所有独立性命题与依赖性命题，因为用概率图表示哪些概率分布也是人为设定的，比如，对于某一概率图：

1. 我们可以令其表示所有满足图中条件独立性的概率分布所组成的集合，除此之外，不再要求概率分布满足图中的依赖关系，即图中读出来依赖性对于在这一集合中的概率分布而言可以成立也可以不成立。换一种说法就是只用图的条件独立性做filter去筛概率分布，只要某个概率分布能满足图中所有的条件独立性，则该分布通过 filter，放入集合。按照下一小节的说法，概率图在这里被我们用作 I map。
2. 类似的，我们也可以令其表示所有满足图中依赖性的概率分布所组成的集合，即概率分布可以存在比概率图更多的依赖性，换而言之，概率分布可以违背图中的独立性。相当于我们只用图的依赖性做 filter，只要概率分布满足图中所有的依赖关系，则通过 filter，放入集合，这一情况就对应下一小节的 D map。
3. 进一步，我们可以既用独立性又用依赖性去做 filter，即概率图对应的概率分布既要满足概率图中所有的独立性，又要满足概率图中所有的依赖性，此时，概率图被用作 perfect map.

我们指出，前面介绍的概率有向图和概率无向图均被我们用作 I map。

联合分布的 decomposition 也是一种抽象表示，它并没有给出概率分布的具体函数表达式，而是表示一类能进行特定分解的概率分布的集合：只要某个概率分布能写成相应的分解形式，则该分布就属于这一集合。通过概率图，我们可以得到联合分布的分解式，进而可以推得概率分布一定成立的条件独立性，但注意，对某一具体分布，它可能存在进一步的分解，即可以存在更多的条件独立性。因此，从概率图出发得到联合分布的 decomposition，这一路线天然地只能给出所代表概率分布集合一定要满足的条件独立性。因此，若要使两条路线存在等价的可能，概率图只能被用作 I map。

总而言之，给定一张概率图，一方面从 decomposition 的角度，我们可以得到该概率图表示的概率分布集合 A；另一方面，通过人为选定适当的条件独立性判定规则 (d-separation/graph separation)，并只用读到的独立性作 filter，我们又可以得到概率分布集合 B，且书中不加证明地指出 $A = B$。可以看到，判定规则是人为设定的，但通过选定规则为 d-separation / graph separation + 只保证读到的条件独立性一定成立 (用作 I map)，恰好可与 decomposition 所表示的概率分布空间等价。因此，以后我们可以直接使用规则在图中读取条件独立性 (再次强调，图中读到的条件独立性是一定成立的，对具体分布而言，可能存在更多的条件独立性)，而不必从 decomposition 出发去推导了。

3.2 有向图与无向图作为 perfect map 时的表达能力

对应 **8.3.4 Relation to directed graphs** 小节后半部分。

we view a specifific (directed or undirected) graph as a filter:

1. A graph is said to be a *D map* (for 'dependency map') of a distribution if every conditional independence statement satisfified by the distribution is reflected in the graph. Thus a completely disconnected graph (no links) will be a trivial D map for any distribution. 概率分布的所有条件独立性都能在图中找到 (换而言之，图中所有的依赖关系也都能在概率分布中找到 (概率分布甚至还存在更多的依赖关系))，则该图是该分布的 D map.
2. If every conditional independence statement implied by a graph is satisfied by a specific distribution, then the graph is said to be an *I map* (for 'independence map') of that distribution. Clearly a fully connected graph will be a trivial I map for any distribution. 图中所有的条件独立性在概率分布中都能找到 (概率分布甚至还存在更多的条件独立性)，则该图是该分布的 I map. 我们介绍的有向图和无向图取的都是 I map 的含义。
3. If it is the case that every conditional independence property of the distribution is reflected in the graph, and vice versa, then the graph is said to be a *perfect map* for that distribution. A perfect map is therefore both an I map and a D map. 若一个图既是 D map 又是 I map，则该图是该分布的 perfect map。

Venn diagram illustrating the set of all distributions P over a given set of variables, together with the set of distributions D that can be represented as a perfect map using a directed graph, and the set U that can be represented as a perfect map using an undirected graph (Figure 8.34):

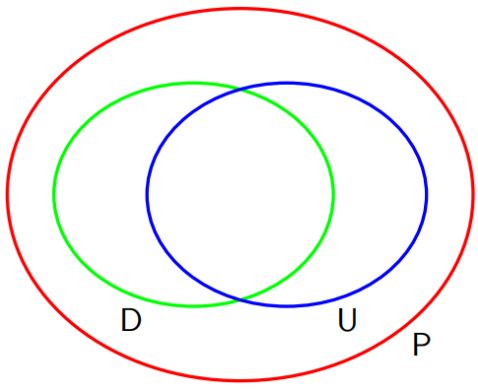


Figure 8.35, Figure 8.36 分别给出了 $\mathrm{D} - \mathrm{U}$ 和 $\mathrm{U} - \mathrm{D}$ 的例子。

---

chain graph: 既包含有向边又包含无向边，前面讨论的有向图和无向图可视为其特例 (The graphical framework can be extended in a consistent way to graphs that include both directed and undirected links. These are called *chain graphs*，and contain the directed and undirected graphs considered so far as special cases). 但将 chain graph 用作 perfect map 时，它也不能表示所有的概率分布 (Although such graphs can represent a broader class of distributions than either directed or undirected alone, there remain distributions for which even a chain graph cannot provide a perfect map).

4 因子图 & 精确推断

对应 **8.4 Inference in Graphical Models** 节。推断问题的核心就是计算概率分布的积分。

首先 **8.4.1 Inference on a chain** 小节以链式结构为例对本节要讨论的内容进行了初探，介绍了链式结构的精确推断，引出信息传递的概念和基于信息传递的 sum-product 算法。而 sum-product 算法本质上就是反向利用乘法对加法的分配率 (multiplication is distributive over addition)，或者说提取公因式来减小计算量，比如

$$\sum_{x_1}\sum_{x_2} x_1x_2 \to \sum_{x_1} x_1 \left(\sum_{x_2} x_2\right)$$

若 x_1, x_2 的可能取值有 K 种，则计算量由 K^2+K-1 降为 $2K-1$. 总而言之，就是先对某个结点积分 (sum，每积分一次就消掉一个随机变量) 再 product 相邻的factor，并不断重复这一过程直至最后求解，进一步，整个计算又可抽象成信息传递的过程。

More generally, inference can be performed effificiently using local message passing on a broader class of graphs called *trees*. In particular, we shall shortly generalize the message passing formalism derived above for chains to give the *sum-product* algorithm, which provides an effificient framework for exact inference in tree-structured graphs. 也就是说，基于局部信息传递的精确推断，即 sum-product algorithm 针对的是具有树结构的图。

4.1 Factor graphs

8.4.2 Trees 小节介绍了有向无环图 (DAGs) 和无向图的树结构，它们是前面介绍的有向图和无向图的子集。

无向树：In the case of an undirected graph, a tree is defined as a graph in which there is one, and only one, path between any pair of nodes. Such graphs therefore do not have loops.

有向树：In the case of directed graphs, a tree is defined such that there is a single node, called the *root*, which has no parents, and all other nodes have one parent.

多树 (可以理解为具有多个根结点的树)： If there are nodes in a directed graph that have more than one parent, but there is still only one path (**ignoring the direction of the arrows**，注意前面在讨论有向图 (也就是有向无环图 DAGs，省略了"无环二字") 时，其"环"的定义需要考虑方向) between any two nodes, then the graph is a called a *polytree*. Such a graph will have more than one node with the property of having no parents, and furthermore, the corresponding moralized undirected graph will have loops (多树道德化为无向图后会有环).

The sum-product algorithm that we derive in the next section is applicable to undirected and directed trees and to polytrees. It can be cast in a particularly simple and general form if we first introduce a new graphical construction called a *factor graph*.

8.4.3 Factor graphs 小节则进一步介绍了因子图

一般地，概率分布可以表示成因子相乘的形式 (in the form of a product of factors):

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s)$$

where $\mathbf{x}_s$ denotes a subset of the variables.

因子图的一般画法：In a factor graph, there is a node (depicted as usual by a circle) for every variable in the distribution, as was the case for directed and undirected graphs. There are also additional nodes (depicted by small squares) for each factor $f_s(\mathbf{x}_s)$ in the joint distribution. Finally, there are undirected links connecting each factor node to all of the variables nodes on which that factor depends.

可以看到，In general, factor graphs can therefore always be drawn as two rows of nodes (variable nodes at the top and factor nodes at the bottom) with links between the rows.

一个因子可以由多个因子组成，反之亦然，因子是人为划定的。因此，对同一个分布，选择不同的因子，画出来的因子图也不同。可以看到，因子图不唯一，我们也不会试图基于因子图去判断条件独立性。

我们希望得到的因子图不存在环，即任意两个结点之间只存在一条路径，这样便于推断。

前面介绍的有向树、无向树以及多树在转化为因子图时可不引入环 (多树在道德化时会引入环)。此外，对于一些其他情况，在转化为因子图时，也可以通过选择合适的因子消除掉环 (In fact, local cycles in a directed graph due to links connecting parents of a node can be removed on conversion to a factor graph by defining the appropriate factor function, as shown in Figure 8.44.)

4.2 Exact inference in tree-structured factor graphs

无论是计算边缘分布还是配分函数，亦或是后验分布，归根结底就是对概率分布求积分。

目标：We shall now make use of the factor graph framework to derive a powerful class of effificient, exact inference algorithms that are applicable to tree-structured graphs.

the *sum-product* algorithm: evaluating local marginals over nodes or subsets of nodes；

the *max-sum* algorithm: find the most probable state.

4.2.1 Message pass algorithm

对应 **8.4.4 The sum-product algorithm** 小节

如前所述，处理对象是无向树，有向树或多树，因为它们可转化为无环的 (即tree-structure) 因子图。由此，我们只需统一地讨论无环因子图即可。

For the moment, we shall suppose that all of the variables are **hidden**. Later we shall see how to modify the algorithm to incorporate evidence corresponding to observed variables (也就是计算条件分布).

(1) interchange summations and products in order to obtain an effificient algorithm; (2) 保留中间结果（两种信息），无需重复计算，归纳抽象出 message pass algorithm.

我们先任取一结点变量 x 作为root结点，为了计算其边缘概率 $p(x)$，我们从根到外一层层地推导，抽象出信息传递的过程；而在实际计算时，则由外到根，信息从叶子结点汇聚到 root x。

首先，重写概率分布的形式，we see that the joint distribution can be written as a product of the form:

$$p(\mathbf{x}) = \prod_{s \in \mathrm{ne}(x)} F_s(x, X_s) \tag{8.62}$$

$\mathrm{ne}(x)$ denotes the set of factor nodes that are neighbours of x, and X_s denotes the set of all variables in the subtree connected to the variable node x via the factor node f_s, and $F_s(x, X_s)$ represents the product of all the factors in the group associated with factor f_s. s 表示与 variable node x 相邻的 factor node，f_s 既可表示 factor node，也可表示该结点对应的因子式。$F_s(x, X_s)$ 是一群 (group) 因子式的累积，包含一个与 x 相邻的因子 f_s，以及与 x 不相邻，但与 x 的路径会经过 f_s 的所有因子，X_s 表示以 f_s 为根节点的子图 (斩断 f_s 与 x 的那条边，剩下的包含 f_s 的部分) 中所有的变量结点的集合。因为这里考察的因子图具有树状结构，所以任意两个结点之间的路径是唯一的。若记 $|\mathrm{ne}(x)| = K$，则所有因子会被唯一地划分到这 K 个 group 中的一类。

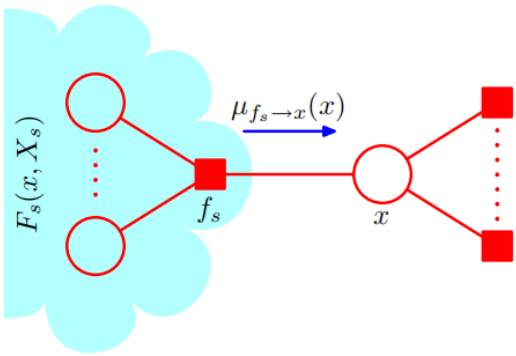
> we recall that，variable node x 只会邻接 factor node f，表示因子式 f 所包含的变量，联合概率分布 $p(\mathrm{x})$ 是所有因子式的累积 (或相差一个归一化常数)。

接着，计算边缘分布 (summing the joint distribution over all variables except x)，并 interchange summations and products (in order to obtain an effificient algorithm)，即:

$$\begin{aligned} p(x) &= \sum_{\mathrm{x}\setminus x} p(\mathrm{x}) \\ &= \sum_{\mathrm{x}\setminus x} \prod_{s\in \mathrm{ne}(x)} F_s(x, X_s) \\ &= \prod_{s\in \mathrm{ne}(x)} \sum_{\mathrm{x}\setminus x} F_s(x, X_s) \\ &\triangleq \prod_{s\in \mathrm{ne}(x)} \mu_{f_s\to x}(x) \end{aligned} \tag{8.63}$$

其中 $\mu_{f_s\to x}(x) = \sum_{\mathrm{x}\setminus x} F_s(x, X_s) = \sum_{X_s} F_s(x, X_s)$ (求和对应积分) can be viewed as *messages* from the factor nodes f_s to the variable node x. 而边缘分布 $p(x)$ 是所有相邻的因子结点对其信息的累积。

Figure 8.46 A fragment of a factor graph illustrating the evaluation of the marginal $p(x)$.



为了计算 factor node to variable node message $\mu_{f_s\to x}(x)$，我们继续向外层推导，由此引入 variable node to factor node message.

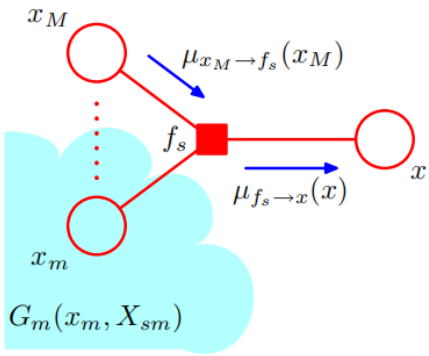
Note that each factor $F_s(x, X_s)$ is described by a factor (sub-) graph and so can itself be factorized. 类似式 (8.62)，我们有:

$$F_s(x, X_s) = f_s(x, x_1, \cdots, x_M) G_1(x_1, X_{s1}) \cdots G_M(x_M, X_{sM}) \tag{8.65}$$

where, for convenience, we have denoted the variables associated with factor f_s, in addition to x, by $x_1, \cdots, x_M$. 也就是说 $x_1, \cdots, x_M$ 是除了 x 之外与 factor node f_s 邻接的变量结点，X_{sm} 则是 x_m 那一枝子图剩余的变量，而 $G_m(x_m, X_{sm})$ 的含义与前面 $F_s(x, X_s)$ 类似，也是相应子图所有因子式的累积。

This factorization is illustrated in Figure 8.47.

Figure 8.47 Illustration of the factorization of the subgraph associated with factor node f_s.



we recall that, $x, x_1, \cdots, x_M, X_{s1}, \cdots, X_{sM}$ 均不相交，
$X_s = x_1 \bigcup \cdots \bigcup x_M \bigcup X_{s1} \bigcup \cdots \bigcup X_{sM}$. 同样地，类似式 (8.63) 的积分，我们有：

$$\begin{aligned}\mu_{f_s \to x}(x) &= \sum_{X_s} F_s(x, X_s) = \sum_{X_s} f_s(x, x_1, \cdots, x_M) G_1(x_1, X_{s1}) \cdots G_M(x_M, X_{sM}) \\ &= \sum_{x_1, \cdots, x_M} f_s(x, x_1, \cdots, x_M) \sum_{X_{s1}} G_1(x_1, X_{s1}) \cdots \sum_{X_{sM}} G_M(x_M, X_{sM}) \\ &= \sum_{x_1, \cdots, x_M} f_s(x, x_1, \cdots, x_M) \mu_{x_1 \to f_s}(x_1) \cdots \mu_{x_M \to f_s}(x_1) \\ &= \sum_{x_1, \cdots, x_M} f_s(x, x_1, \cdots, x_M) \prod_{m \in \mathrm{ne}(f_s) \backslash x} \mu_{x_m \to f_s}(x_m)\end{aligned} \tag{8.66}$$

其中我们定义了 message from variable node to factor node，即积分
$\mu_{x_m \to f_s}(x_m) = \sum_{X_{sm}} G_m(x_m, X_{sm})$.

至此，我们得到了 factor node to variable node 以及 variable node to factor node 两种message：

$$\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s) \tag{8.64}$$

$$\mu_{x_m \to f_s}(x_m) = \sum_{X_{sm}} G_m(x_m, X_{sm}) \tag{8.67}$$

In each case, we see that messages passed along a link are always **a function of the variable associated with the variable node** that link connects to. 即无论是哪种message，即不管信息传递的方向，它均是这条边所连接的变量结点的函数。

式 (8.66) 给出了如何用 variable node to factor node message $\mu_{x \to f}(x)$ 表示 factor node to variable node message $\mu_{f \to x}(x)$：The result (8.66) says that to evaluate the message sent by a factor node to a variable node along the link connecting them, take the product of the incoming messages along **all other** links coming into the factor node, multiply by the factor associated with that node, and then marginalize over all of the variables associated with the incoming messages. 可以看到, a factor node can send a message to a variable node once it has received incoming messages from all other neighbouring variable nodes.

类似地，也可以用 $\mu_{f \to x}(x)$ 表示 $\mu_{x \to f}(x)$，对应就是继续往外推一层的情况，即得书中式 (8.69)，总结而言：

Thus to evaluate the message sent by a variable node to an adjacent factor node along the connecting link, we simply take the product of the incoming messages along **all of the other** links (更简单的总结：The message sent by a variable node to a factor node, as we have seen, is simply the product of the incoming messages on other links). 同样地, we note that a variable node can send a message to a factor node once it has received incoming messages from all other neighbouring factor nodes. 可以看到，一个极端情况是变量节点只邻接2个因子节点，则变量到某个因子的信息不需要任何计算，直接传递相应的另一个因子到变量的信息即可 (Note that any variable node that has only two neighbours performs no computation but simply passes messages through unchanged.)

可以一层一层向外扩展，直到叶子结点，叶子结点处的信息传递可分如下两种情况

$$\mu_{x\to f}(x) = 1 \tag{8.70}$$
$$\mu_{f\to x}(x) = f(x) \tag{8.71}$$

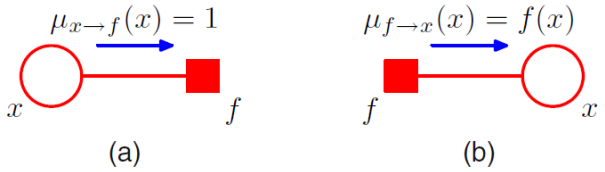


Figure 8.49 The sum-product algorithm begins with messages sent by the leaf nodes, which depend on whether the leaf node is (a) a variable node, or (b) a factor node.

上述即为 message pass algorithm (sum-product algorithm) 的推导过程，从根向外，而在实际计算时则由外向根，对根节点的信息传递进行初始化，即 $\mu_{x\to f}(x) = 1$ or $\mu_{f\to x}(x) = f(x)$，再基于式 (8.66) or (8.69) 进行信息传播，直至求解。At this point, it is worth pausing to summarize the particular version of the sum-product algorithm obtained so far for evaluating the marginal $p(x)$. We start by viewing the variable node x as the root of the factor graph and initiating messages at the leaves of the graph using (8.70) and (8.71). The message passing steps (8.66) and (8.69) are then applied recursively (指信息间的递推关系，而不是说某个信息或被多次更新，某个信息只用被计算一次) until messages have been propagated along every link, and the root node has received messages from all of its neighbours. Each node can send a message towards the root once it has received messages from all of its other neighbours. Once the root node has received messages from all of its neighbours, the required marginal can be evaluated using (8.63).

任选一个结点作为 root 结点，经过从外向根再从根向外两轮传播高效计算所有边上的两个方向的信息：

1. Arbitrarily pick any (variable or factor) node and designate it as the root.
2. Propagate messages from the leaves to the root as before. At this point, the root node will have received messages from all of its neighbours.
3. It can therefore send out messages to all of its neighbours. These in turn will then have received messages from all of their neighbours and so can send out messages along the links going away from the root, and so on. In this way, messages are passed outwards from the root all the way to the leaves.

By now, a message will have passed in both directions across every link in the graph, and every node will have received a message from all of its neighbours. 严格来说，上述两轮信息传递才称之为 sum-prodcut 算法，而 message pass algorithm 只是给出了信息的传递的计算过程。不过，一般我们也不作严格的区分。

需要说明的是，书中 P403 第一次提到 *belief propagation* 算法，但并没有介绍，只是说其相当于 sum-product algorithm 的特殊情况，然后就开始介绍 sum-product algorithm 算法了 (在第 10 章 Approximate Inference 的语境中，我们可以认为提到 BP 算法时就是指 sum-product 算法)。书中之后介绍了 loopy belief propagation，但它是一种近似推断算法。

> There is an algorithm for exact inference on directed graphs without loops known as *belief propagation* (Pearl, 1988; Lauritzen and Spiegelhalter, 1988), and is equivalent to a special case of the sum-product algorithm. Here we shall consider only the sum-product algorithm because it is simpler to derive and to apply, as well as being more general. —P403

由此，可以很方便地计算如下的量：

1. 可高效计算任意变量结点的边缘分布，而不必每想得到一个变量结点的边缘分布均进行一轮信息传递。
2. 某个因子节点对应的变量集合的边缘分布：

$$p(\mathbf{x}_s) = f(\mathbf{x}_s) \prod_{i \in \mathrm{ne}(f_s)} \mu_{x_i \to f_s}(x_i)$$

3. 若因子图源于无向图模型，则还需计算归一化常数 Z，在 1 小点的基础上做一次一元积分即可。
4. 计算条件分布 （或者说概率图模型中含有可观察变量而不全是隐变量时的处理）：将 $\mathbf{x}$ 划分为可见变量 $\mathbf{v}$ 以及隐变量 $\mathbf{h}$, 问题即为计算 $p(\mathbf{h}|\mathbf{v}=\hat{\mathbf{v}})$ 或 $p(h_i|\mathbf{v}=\hat{\mathbf{v}})$ 等。无非就是在计算信息时，对可见变量 $\mathbf{v}$ 的积分退化为只用考虑取 $\hat{\mathbf{v}}$ 的情况，其取值空间中其他可能的取值对积分的贡献为 0。相当于相比所有 $\mathbf{x}$ 均不可见的情况，相当于是修改了其中部分变量 $\mathbf{v}$ 的概率，即令其取值为 $\hat{\mathbf{v}}$ 的概率变为 1，而取其他值得概率为 0，由此便回归到所有变量均不可见的因子图的处理逻辑。此外，条件概率的计算还会涉及到概率分布的归一化，处理同所有变量均不可见的因子图的情况，也很简单。

4.2.2 The max-sum algorithm

对应 **8.4.5 The max-sum algorithm** 小节。

承上启下：The sum-product algorithm allows us to take a joint distribution $p(x)$ expressed as a factor graph and efficiently find marginals over the component variables. Two other common tasks are to find a setting of the variables that has the largest probability and to find the value of that probability. These can be addressed through a closely related algorithm called max-sum, which can be viewed as an application of *dynamic programming* in the context of graphical models.

理论基础与 sum-product 算法的理论基础类似：

- In deriving the sum-product algorithm, we made use of the distributive law (8.53) for multiplication. Here we make use of the analogous law for the max operator

  $$\max(ab, ac) = a \max(b, c)$$

  which holds if $a \geqslant 0$ (as will always be the case for the factors in a graphical model). This allows us to exchange products with maximizations. + 概率分布的因子式累积的表示形式。

max-product 算法：In particular, suppose that we designate a particular variable node as the ‘root’ of the graph. Then we start a set of messages propagating inwards from the leaves of the tree towards the root, with each node sending its message towards the root once it has received all incoming messages from its other neighbours. The final maximization is performed over the product of all messages arriving at the root node, and gives the maximum value for $p(\mathbf{x})$. This could be called the **max-product** algorithm and is identical to the sum-product algorithm except that summations are replaced by maximizations. Note that at this stage, messages have been sent from leaves to the root, but not in the other direction. 任选一个root后，由外向root一轮传播即可。

进一步，取对数，由对数函数的单调性，max 与 log 可以互换。且分配率依然保持，可得 **max-sum** algorithm. 这样做是为了避免小概率累积造成的下溢问题。The max operator and the logarithm function can be interchanged, so that

$$\ln\left(\max_{\mathbf{x}} p(\mathbf{x})\right) = \max_{\mathbf{x}} \ln p(\mathbf{x})$$

The distributive property is preserved because

$$\max(a+b, a+c) = a + \max(b, c)$$

Thus taking the logarithm simply has the effect of replacing the products in the max-product algorithm with sums, and so we obtain the max-sum algorithm.

sum-product -> max-product -> max-sum.

---

max-sum 算法的信息传递递推公式以及初始化（叶子结点处的信息传递公式）如下：

$$\mu_{f\to x}(x) = \max_{x_1,\cdots,x_M}\left[\ln f(x,x_1,\cdots,x_M) + \sum_{m\in \mathrm{ne}(f)\backslash x}\mu_{x_m\to f}(x_m)\right] \tag{8.93}$$

$$\mu_{x\to f}(x) = \sum_{l\in \mathrm{ne}(x)\backslash f}\mu_{f_l\to x}(x) \tag{8.94}$$

同样地，不论是哪种类型的信息 $\mu_{f\to x}(x), \mu_{x\to f}(x)$ 它们均是所属边连接的变量结点 x 的函数 (the messages are functions of node variables x)，表示相应子图除了 x 外的其他变量的值使对应因子式之积最大的情况。

叶子结点处的初始化公式：

$$\mu_{x\to f}(x) = 0 \tag{8.95}$$

$$\mu_{f\to x}(x) = \ln f(x) \tag{8.96}$$

所有信息汇集到 root 结点 x 后，最大概率的计算公式：

$$p^{\max} = \max_x\left[\sum_{s\in \mathrm{ne}(x)}\mu_{f_s\to x}(x)\right] \tag{8.97}$$

上面计算的是最大概率，我们还需要计算最大概率对应的 $\mathbf{x}$ 的值。Now we turn to the second problem of finding the configuration of the variables for which the joint distribution attains this maximum value.

不可行的方案：类似于 sum-product 算法，进行两轮信息传递，这样对任意一个变量结点 x_i，类似于式 (8.97)，我们都可以基于式

$$\hat{x}_i = \arg\max_{x_i}\left[\sum_{s\in \mathrm{ne}(x)}\mu_{f_s\to x}(x)\right]$$

得到联合概率最大时，x_i 的取值 $\hat{x}_i$。这样，遍历每个变量结点，组合起来，得到 $\hat{\mathbf{x}}$。这样做的问题是，使概率最大的 x_i 的取值可能有多个，我们每次都是独立地考察某一个变量 x_i，因此，并不知道各结点最优值的匹配关系，因此就得不到 $\mathbf{x}$ 的全局最优解。a globally consistent maximizing confifiguration.

可行的方案：back-tracking. If a message is sent from a factor node f to a variable node x, a maximization is performed over all other variable nodes $x_1,\cdots,x_M$ that are neighbours of that factor node, using (8.93). When we perform this maximization, we keep a record of which values of the variables $x_1,\cdots,x_M$ gave rise to the maximum. Then in the back-tracking step, having found $x^{\max}$, we can then use these stored values to as sign consistent maximizing states $x_1^{\max},\cdots,x_M^{\max}$. The max-sum algorithm, with back-tracking, gives an exact maximizing confifiguration for the variables provided the factor graph is a tree.

上述算法应用于 hidden Markov model 时即为 the Viterbi algorithm (维特比算法)。

同样地，也涉及到某些变量是已观测的情况的处理。As with the sum-product algorithm, the inclusion of evidence in the form of observed variables is straightforward. 处理是也是很直接的。

具体做法: The observed variables are clamped to their observed values, and the maximization is performed over the remaining hidden variables.

数学上的描述: This can be shown formally by including identity functions for the observed variables into the factor functions, as we did for the sum-product algorithm.

4.3 Exact inference in general graphs

对应 **8.4.6 Exact inference in general graphs** 小节。

For many practical applications, however, we have to deal with graphs having loops.

The message passing framework can be generalized to arbitrary graph topologies, giving an exact inference procedure known as the *junction tree algorithm*. 文中只是简单介绍了下。

1. 无向图。If the starting point is a directed graph, it is first converted to an undirected graph by moralization, whereas if starting from an undirected graph this step is not required.
2. 三角化。Next the graph is *triangulated* (三角化), which involves finding chord-less cycles containing four or more nodes and adding extra links to eliminate such chord-less cycles. Note that the joint distribution for the resulting triangulated graph is still defined by a product of the same potential functions, but these are now considered to be functions over expanded sets of variables.
3. 构建 join tree。Next the triangulated graph is used to construct a new tree-structured undirected graph called a *join tree*, whose nodes correspond to the maximal cliques of the triangulated graph, and whose links connect pairs of cliques that have variables in common. 更多 join tree 的构建细节可参见 lec3.dvi (stanford.edu)
4. 信息传递。Finally, a two-stage message passing algorithm, essentially equivalent to the sum-product algorithm, can now be applied to this junction tree in order to find marginals and conditionals.

万变不离其宗，算法的思路：Although the junction tree algorithm sounds complicated, at its heart is the simple idea that we have used already of exploiting the factorization properties of the distribution to allow sums and products to be interchanged so that partial summations can be performed, thereby avoiding having to work directly with the joint distribution. **The role of the junction tree is to provide a precise and effificient way to organize these computations. It is worth emphasizing that this is achieved using purely graphical operations!**

4.4 近似推断：Loopy belief propagation

对应 **8.4.7 Loopy belief propagation** 小节，处理的是带环图模型的近似推断问题。不同于前文的精确推断的信息传递只需2轮，此时的信息传递可以一直进行下去，且不一定保证收敛。Here we consider one simple approach to **approximate inference in graphs with loops**, which builds directly on the previous discussion of exact inference in trees. The idea is simply to apply the sum-product algorithm even though there is no guarantee that it will yield good results. This approach is known as *loopy belief propagation*. However, because the graph now has cycles, information can flow many times around the graph. For some models, the algorithm will converge, whereas for others it will not.

为了开启计算，loopy BP 需要先对所有 message 进行初始化：We have seen that a message can only be sent across a link from a node when all other messages have been received by that node across its other links. Because there are loops in the graph, this raises the problem of how to initiate the message passing algorithm. To resolve this, we suppose that an initial message given by the unit function has been passed across every link in each direction. Every node is then in a position to send a message.

5 Learning the graph structure

对应 **8.4.8 Learning the graph structure** 小节，也就是对图的结构进行学习，道理简单，就是贝叶斯那一套，但实操难度很大，不再赘述。