Probability Graph 01 Background

Chen Gong

23 November 2019

机器学习的重要思想就是，对已有的数据进行分析，然后对未知数据来进行预判或者预测等。这里的图和我们之前学习的数据结构中的图有点不太一样，俗话说有图有真相，这里的图是将概率的特征引入到图中，方便我们进行直观分析。

1 概率的基本性质

我们假设现在有一组高维随机变量，$p(x_1, x_2, \cdots, x_n)$，它有两个非常基本的概率，也就是条件概率和边缘概率。条件概率的描述为 $p(x_i)$，条件概率的描述为 $p(x_j|x_i)$。

同时，根据这两个基本的概率，我们可以得到两个基本的运算法则：Sum Rule 和 Product Rule。

Sum Rule：$p(x_1) = \int p(x_1, x_2)dx_2$。

Product Rule：$p(x_1, x_2) = p(x_1)p(x_2|x_1) = p(x_2)p(x_1|x_2)$。

根据这两个基本的法则，我们可以推出 Chain Rule 和 Bayesian Rule。

Chain Rule：

$$p(x_1, x_2, \cdots, x_N) = \prod_{i=1}^{N} p(x_i|x_1, x_2, \cdots, x_{i-1}) \tag{1}$$

Bayesian Rule：

$$p(x_2|x_1) = \frac{p(x_1, x_2)}{p(x_1)} = \frac{p(x_1, x_2)}{\int p(x_1, x_2)dx_2} = \frac{p(x_1|x_2)p(x_2)}{\int p(x_1|x_2)p(x_2)dx_2} \tag{2}$$

2 条件独立性

首先，我们想想高维随机变量所遇到的困境，也就是维度高，计算复杂度高。大家想想，当维度较高时，这个 $p(x_1, x_2, \cdots, x_N) = \prod_{i=1}^{N} p(x_i|x_1, x_2, \cdots, x_{i-1})$ 肯定会算炸去。所以，我们需要简化运算，之后我们来说说我们简化运算的思路。

1. 假设每个维度之间都是相互独立的，那么我们有 $p(x_1, x_2, \cdots, x_N) = \prod_{i=1}^{N} p(x_N)$。比如，朴素贝叶斯就是这样的设计思路，也就是 $p(x|y) = \prod_{i=1}^{N} p(x_i|y)$。但是，我们觉得这个假设太强了，实际情况中的依赖比这个要复杂很多。所以我们像放弱一点，增加之间的依赖关系，于是我们有提出了马尔科夫性质 (Markov Propert)。

2. 假设每个维度之间是符合马尔科夫性质 (Markov Property) 的。所谓马尔可夫性质就是，对于一个序列 $\{x_1, x_2, \cdots, x_N\}$，第 i 项仅仅只和第 $i-1$ 项之间存在依赖关系。用符号的方法我们可以表示为：

$$x_j \perp x_{i+1} | x_i, j < i \tag{3}$$

在 HMM 里面就是这样的齐次马尔可夫假设，但是还是太强了，我们还是要想办法削弱。自然界中经常会出现，序列之间不同的位置上存在依赖关系，因此我们提出了条件独立性。

3. 条件独立性：**条件独立性假设是概率图的核心概念。它可以大大的简化联合概率分布**。而用图我们可以大大的可视化表达条件独立性。我们可以描述为：

$$X_A \perp X_B | X_C \tag{4}$$

而 X_A, X_B, X_C 是变量的集合，彼此之间互不相交。

3 概率图算法分类

概率图的算法大致可以分为三类，也就是，表示 (Representation)，推断 (Inference) 和学习 (Learning)。

3.1 Representation

知识表示的方法，可以分为有向图，Bayesian Network；和无向图，Markov Network，这两种图通常用来处理变量离散的情况。对于连续性的变量，我们通常采用高斯图，同时可以衍生出，Gaussian Bayesian Network 和 Guassian Markov Network。

3.2 Inference

推断可以分为精准推断和近似推断。所谓推断就是给定已知求概率分布。近似推断中可以分为确定性推断 (变分推断) 和随机推断 (MCMC)，MCMC 是基于蒙特卡罗采样的。

3.3 Learning

学习可以分为参数学习和结构学习。在参数学习中，参数可以分为变量数据和非隐数据，我们可以采用有向图或者无向图来解决。而隐变量的求解我们需要使用到 EM 算法，这个 EM 算法在后面的章节会详细推导。而结构学习则是，需要我们知道使用那种图结构更好，比如神经网络中的节点个数，层数等等，也就是现在非常热的 Automate Machine Learning。

Probability Graph 02 Bayesian Network

Chen Gong

24 November 2019

概率图模型中，图是用来表达的，将概率嵌入到了图中之后，使得表达变得非常的清晰明了。在我们的联合概率计算中，出现了一些问题：

$$p(x_1, x_2, \cdots, x_N) = \prod_{i=1}^{N} p(x_i|x_{1:i-1}) \tag{1}$$

这样的计算维度太高了，所以我们引入了条件独立性，表达为 $X_A \perp X_B|X_C$。那么采用因子分解的方法我们可以将联合概率的计算进行分解为：

$$p(x_1, x_2, \cdots, x_N) = \prod_{i=1}^{N} p(x_i|x_{pa\{i\}}) \tag{2}$$

其中，$pa\{i\}$ 表示为 x_i 的父亲节点。而概率图可以有效的表达条件独立性，直观性非常的强，我们接下来看看概率图中经典的三种结构。

1 概率图的三种基本结构

对于一个概率图，我们可以使用拓扑排序来直接获得，条件独立性的关系。如果存在一个关系由一个节点 x_i 指向另一个节点 x_j，我们可以记为 $p(x_j|x_i)$。我们现在需要定义一些规则来便于说明，对于一个概率图如下所示：

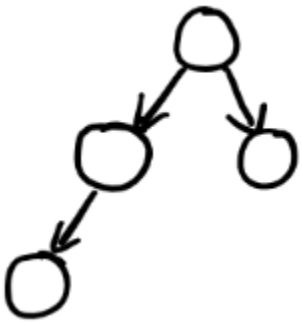

图 1: 基本概率图模型

对于一个箭头 $\longrightarrow$ 来说，箭头所在的方向称为 Head，另一端被称为 Tail。

1.1 Tail to Tail 结构

Tail to Tail 的模型结构图，如下图所示，由于 b 节点在 a 节点和 c 节点的 Tail 部分，所以被我们称为 Tail to Tail 结构。

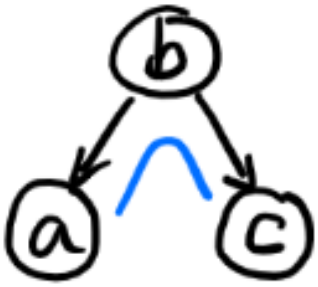

图 2: Tail to Tail 结构示意图

我们使用因子分析来计算联合概率可以得到：

$$p(a,b,c) = p(b)p(a|b)p(c|b) \tag{3}$$

使用链式法则，同样我们也可以得到：

$$p(a,b,c) = p(b)p(a|b)p(c|b,a) \tag{4}$$

对比一下公式 (3) 和公式 (4)，我们可以对比得到：

$$p(c|b) = p(c|b,a) \tag{5}$$

实际上，这里就已经就可以看出 $a \perp c$ 了，因为 a 的条件增不增加都不会改变 c 的概率，所以 a 和 c 之间是相互独立的。可能有的同学还是觉得不好理解，那么我们做进一步的分析：

$$p(c|b)p(a|b) = p(c|b,a)p(a|b) = p(a,c|b) \tag{6}$$

$$\Rightarrow p(c|b)p(a|b) = p(a,c|b) \tag{7}$$

这样，我们就可以看得很明白了。这就是条件独立性，在 a 的条件下，b 和 c 是独立的。实际在概率图中就已经蕴含这个分解了，只看图我们就可以看到这个性质了，这就是图的直观性，条件独立性和图是一样的。那么 $a \perp c$ 可以被我们看为：给定 b 的情况下，如果 b 被观测到，那么 a 和 c 之间是阻塞的，也就是相互独立。

1.2 Head to Tail 结构

图 3: Head to Tail 结构示意图

其实，和 Head to Head 结构的分析基本是上一模一样的，我们可以得到 $a \perp c|b$。也就是给定 b 的条件下，a 和 c 之间是条件独立的。也就是 b 被观测的条件下，路径被阻塞。

1.3 Head to Head 结构

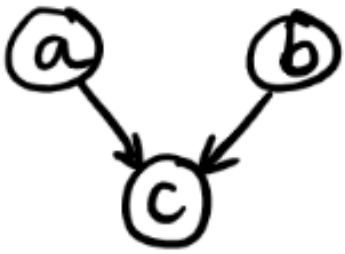

图 4: Head to Head 结构示意图

在默认情况下 $a \perp b$，也就是若 c 被观测，a 和 b 之间是有关系的。我们可以推导一下默认情况。

$$\begin{aligned} p(a,b,c) =& p(a)p(b)p(c|a,b) \\ =& p(a)p(b|a)p(c|a,b) \end{aligned} \tag{8}$$

我们可以得出 $p(b) = p(b|a)$，也就是 $a \perp b$。

Probability Graph 03 D-Separation

Chen Gong

25 November 2019

上一小节中，我们已经大致介绍了概率图之间的三种基本拓扑结构。下面我们来介绍一下，这三种拓扑结构的运用，以及如何扩展到我们的贝叶斯模型中。

1 D-separation

假设我们有三个集合，X_A, X_B, X_C，这三个集合都是可观测的，并且满足 $X_A \perp X_C | X_B$。那我们想想，如果有一些节点连成的拓扑关系图，如果一个节点 $a \in X_A, c \in X_C$，那么如果 a 和 c 之间相互独立的话，他们之间连接的节点需要有怎样的条件？我们通过一个图来进行描述。

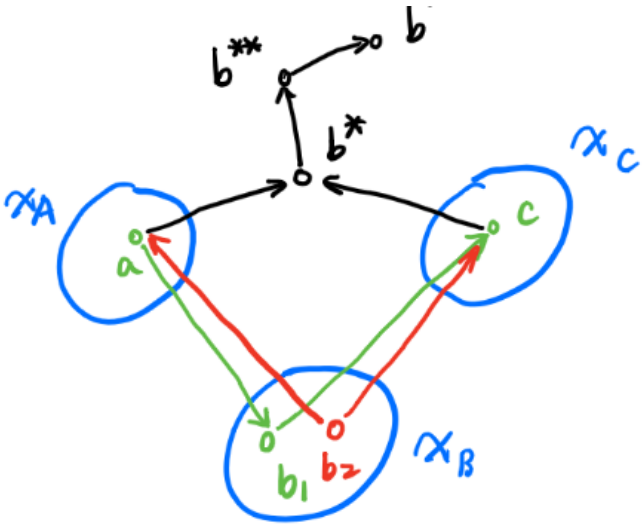

图 1: 节点之间的拓扑联系图

根据上一小节我们的讲解，我们可以想想，假如从 a 到 c 中间通过一个节点 b，那么路径什么时候是连通的？什么时候又是阻塞的呢？我们可以分两种情况来讨论，为什么是两种？上一节我们就已经讲解过，Tail to Tail 结构和 Head to Tail 结构其实是一样的，但是 Head to Head 结构是反常的。所以我们就分开进行讨论。

1. 如果是 Tail to Tail 结构和 Head to Tail 结构，那么中间节点 b_1, b_2 必然要位于可观测集合 X_B 中，那么 a 和 c 才是相互独立的。

2. 如果是 Head to Head 结构，那就不一样了，在一般情况下 $a \perp c$，那就是 $b \notin X_B$，包括他的子节点都不可以被观测到，不然就连通了。

如果，符合上述两条规则，那么我们就可以称 a 和 c 之间是 D-Separation 的，实际上还有一个名字，叫做全局马尔科夫性质 (Global Markov Property)。D-Separation 非常的关键，我们可以直接用这个性质来检测两个集合关于另外一个集合被观测的条件是不是条件独立。

2 Markov Blanket

我们首先定义 x_{-i} 为 $\{x_1, x_2, \cdots, x_{i-1}, x_{i+1}, \cdots, x_N\}$，这个序列中唯独不包括 x_i。那么，我们假设除了 x_i 节点，其他节点都是可观测的，那么我们需要计算概率：

$$p(x_i|x_{-i}) = \frac{p(x_i, x_{-i})}{p(x_{-i})} = \frac{p(x)}{\int_{x_i} p(x)dx_i} = \frac{\prod_{j=1}^{N} p(x_j|x_{pa(j)})}{\int_{x_i} \prod_{j=1}^{N} p(x_j|x_{pa(j)})dx_i} \tag{1}$$

我们分析一下上述等式，我们可以分成两部分，将和 x_i 相关的部分记为 $\bar{\Delta}$，和 x_i 不相关的部分记为 Δ。那么公式 (1) 可以被我们改写为：

$$p(x_i|x_{-i}) = \frac{\Delta \cdot \bar{\Delta}}{\int_{x_i} \Delta \cdot \bar{\Delta} dx_i} = \frac{\Delta \cdot \bar{\Delta}}{\Delta \cdot \int_{x_i} \bar{\Delta} dx_i} == \frac{\bar{\Delta}}{\int_{x_i} \bar{\Delta} dx_i} \tag{2}$$

我们可以将 $p(x_i|x_{-i})$ 表示为一个函数 $f(\bar{\Delta})$。那么 x_i 和其他所有点的关系可以被化简为只和 x_i 相关的点的关系。那么，我们将这个关系大致抽象出来，通过图来进行分析，找一找哪些节点是和 x_i 相关的，直观性也是概率图模型的一大优点。假设 x_i 是可以被观测到的

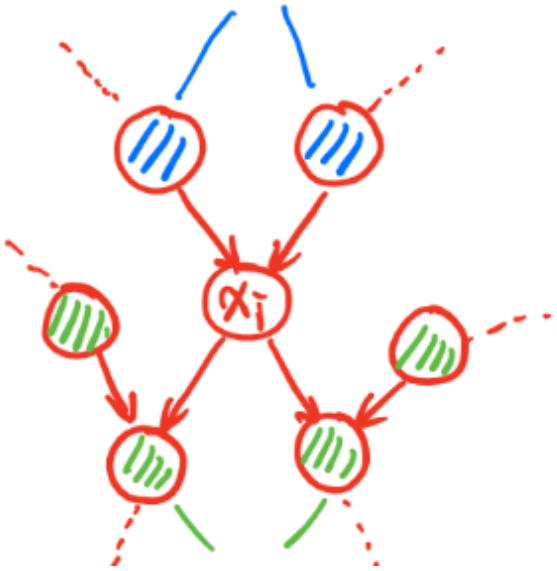

图 2: 节点之间的抽象拓扑联系图

$P(x_i|x_{-1})$ 首先包括一部分为 $p(x_i|x_{pa(i)})$，第二部分为 $p(x_{child(i)}|x_i, x_{pa(child(x_i))})$。为什么第二部分可以这样写呢？首先 x_i 作为两个父亲节点，得到两个孩子节点，毫无疑问对吧，那么我们可以写成 $p(x_{child(i)}|x_i)$。但是和 $p(x_{child(i)})$ 相关的变量除了 x_i 肯定还有其他的，也就是 $x_{pa(child(x_i))}$，所以我们就可以得到 $p(x_{child(i)}|x_i, x_{pa(child(x_i))})$。实际上，这些 x_i 周围的节点，也就是我用阴影部分画出的那些，可以被称为 Markov Blanket。从图上看就是留下和父亲，孩子，孩子的另一个双亲，其他的节点可以忽略，也就是和周围的关系的连接。

总结：D-Separation 是在概率图中，帮助我们快速的判断条件独立性。

Probability Graph 04 Example

Chen Gong

26 November 2019

上一节中，我们讲的是模型通用的一些概念，这一节开始，我们要讲一讲贝叶斯网络具体的例子。我们从单一，混合，时间和连续，四个角度来看看 Bayesian Network，这个四个方法是一步一步越来越难的。

1 单一

单一最典型的代表就是 Naive Bayesian，这是一种 classification 的模型。对于 $p(x|y)$ 的问题来说，假设各维度之间相互独立，于是就有：

$$p(x|y) = \prod_{i=1}^{N} p(x_i|y=1) \tag{1}$$

概率图模型表示如下所示：

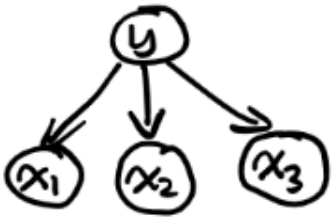

图 1: Naive Bayesian Network 拓扑表示

很显然是一个 Tail to Tail 的模型，我们很简单可以得出 $x_1 \perp x_2 \perp \cdots \perp x_N$。

2 混合

最常见的就是 Gaussian Mixture Model (GMM)，这是一种聚类模型，将每个类别拟合一个分布，计算数据点和分布之间的关系来确定，数据点所属的类别。我们假设 Z 是一个隐变量，并且 Z 是离散的变量，那么 $x|z \sim \mathcal{N}(\mu, \Sigma)$。我们用模型可以表示为：

图 2: Gaussian Mixture Model 的可视化表示

3 时间

时间上我们大致可以分成两种。第一种是 Markov chain，这是随机过程中的一种；第二种事 Gaussian Processing，实际上就是无限维的高斯分布。

实际上时间和混合可以一起看，我们称之为动态系统模型。并且，我们就可以衍生出三种常见的模型，这里讲的比较的模糊，在后面的章节我们会进行详细的分析。第一种是隐马尔可夫模型 (HMM)，这是一种离散的模型；第二种是线性动态系统 (LDS)，这是一种线性的连续的模型，包括典型的 Kalman Filter。第三种是 Particle Filter，一种非高斯的，非线性的模型。

4 连续

连续就是 Guassian Bayesian Network，前面有提到过。

大家可能听到这么多的名词会一脸懵逼呀，懵逼是很正常的，因为这些名词只是给了个印象，后面我们会进行详细的分析。实际上一个整体的趋势就是从单一到混合，从有限到无限。也就是从空间和时间两个角度来进行分析，都是从离散到连续的过程。至于具体为什么这么分，还得具体学习了算法我们才能够很好的理解，本小节只是起了一个高屋建瓴的作用。

Probability Graph 05 Markov Network

Chen Gong

27 November 2019

上一小节中，我们分析了有向图 Bayesian Network，得到了因子分解法，$p(x) = \prod_{i=1}^{N} p(x_i|x_{pa(i)})$。虽然，有向图中可以方便直观的表达条件独立性，但是它也有它的局限性。也就是我们提到的对于 Head to Head 的结构来说，当中间节点被观察到的时候，反而是两端的节点是相关的。这违反了条件独立性的特点，也就是当某些变量被观察到时，其他变量之间是独立的特点，这种情况有点反常，并不太好办。

但是，在无向图中就完全不会出现这样的情况，因为本来就没有方向，而且在无向图中也有类似的 D-Separation 性质。

1 Condition Independence in Markov Network

Markov 中的条件独立，大致可以被我们分成三种情况，Global Markov，Local Markov 和 Pair Markov。

1.1 Global Markov

假设现在有三个集合 $X_A \perp X_B | X_C$，我们想得到 $a \in X_A, b \in X_B$ 之间相互独立，这个应该怎么办？我们给出，只有 a 和 b 的中间节点至少有一个位于 c 中，那么我们就可以得到 $a \perp c$。

1.2 Local Markov

我们以下图的一个 Markov Network 为例，

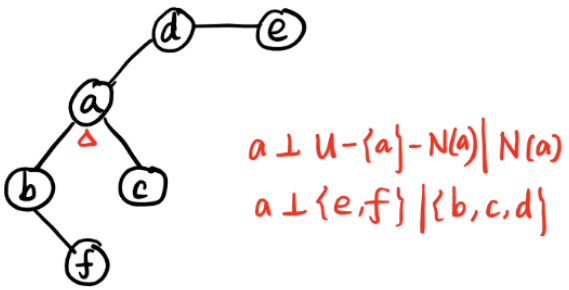

图 1: Markov Network 示意图

用文字的语言来描述就为 $a\perp\{$ 全集 $-a-$ 邻居 $\}|$ 邻居。那么在这个图中，我们就可以表示为 $a\perp\{e,f\}|\{b,c,d\}$。

1.3 Pair Markov

成对马尔可夫性质可以被我们描述为：$x_i\perp x_j|x_{-i-j}$ $(i\neq j)$。其中，x_{-i-j} 为从全集中去掉 x_i 和 x_j 而留下了的集合。

那么条件独立性就可以提现在，Global，Local 和 Pair 中。其中 Global⇔Local⇔Pair。也就是这三种条件独立的方法得到的结果是一样的。

2 因子分解法

我们想一想在一个无向图中，如何来体现我们想要的条件独立性。这里的引入和之前的不太一样，我们首先需要引入几个概念。**团**: 这是一个关于节点的集合，集合中的节点是相互连通的。而最大团，就很好理解了吧，也就是最大的连通子集。我们可以将无向图的分离定义到团上，我们假设 $c_1,c_2,\cdots,c_k$ 表示为团。那么，我们可以将联合概率定义为：

$$p(x)=\frac{1}{Z}\prod_{i=1}^{k}\phi_i(x_{c_i}) \tag{1}$$

其中，z 是归一化因子，因为没有归一化因子的话，这不能被称为一个概率分布，因为概率分布首先就要保证和等于 1。那么，z 被定义为

$$z=\sum_{x}\prod_{i=1}^{k}\phi_i(x_{c_i})=\sum_{x_1}\cdots\sum_{x_N}\prod_{i=1}^{k}\phi_i(x_{c_i}) \tag{2}$$

3 Gibbs Distribution 和 Exponential Distribution

这个部分是上面部分的一个加深理解，首先我们需要总结一下。

1. Global Markov：$X_A\perp X_C|X_B$。也就是 X_A 和 X_B 之间所有的连接都必须在 X_C 中，此时在无向图中满足全局马尔可夫性。

2. Local Markov Network：$x_i\perp x_{-i-nb(i)}|x_{nb(i)}$，其实 nb(i)：neighbor of node i。

3. Pair Markov：$x_i\perp x_j|x_{-i-j}$。

1⇔2⇔3⇔ 因子分解 (基于最大团)。这里面实际上是有个 Hammesley-Clifford 定理的，这个定理的证明非常的困难，我们这里就不做过多的阐述 (实际上我也看的有点懵逼，有需求的小伙伴可以查下 Hammesley-Clifford 定理的证明)。

因子分解：

在我们之前的定义中，

$$p(x)=\frac{1}{Z}\prod_{i=1}^{k}\phi_i(x_{c_i}) \tag{3}$$

c_i：最大团；x_{c_i}：最大团的随机变量集合；$\phi(x_{c_i})$：势函数，必须为正。这里的概念都是来自于统计物理学和热力学的过程。这里的势函数还有可以做文章的地方。

因为，势函数必定为正，我们可以将势函数表达为 $\phi(x_{c_i}) = \exp\{-E(x_{c_i})\}$。其中，$E(x_{c_i})$ 称为 Energy function。实际上用这种形式表达的 $p(x)$，为 Gibbs Distribution，或者又被称之为 Boltzman Distribution。有了 $\phi(x_{c_i})$ 的形式，我们可以进一步推导得：

$$p(x) = \frac{1}{Z}\prod_{i=1}^{K}\phi(x_{c_i}) = \frac{1}{Z}\prod_{i=1}^{K}\exp\{-E(x_{c_i})\} = \frac{1}{Z}\exp\{-\sum_{i=1}^{k}E(x_{c_i})\} \tag{4}$$

我们再来回顾一下指数族分布，指数族分布的通用表达形式为：

$$p(x) = h(x)\exp\{\eta^T\phi(x) - A(\eta)\} = h(x)\frac{1}{Z(\eta)}\exp\{\eta^T\phi(x)\} \tag{5}$$

在这里我们把 $\exp\{-A(\eta)\}$，直接记为 $Z(\eta)$。大家观察就会发现势函数也就是 Gibbs Distribution 就是一个指数族分布。Gibbs 是来自统计物理学，形式上和指数族分布时一样的。而指数族分布实际上是由最大熵分布得到的，那么我们可以理解 Gibbs 分布也是有最大熵原理得到的。而马尔可夫随机场 (Markov Random Field) 实际上等价于 Gibbs 分布。至于为什么？这实际上全部都在 Hammesley-Clifford 定理中，有兴趣的同学，请自行查阅。

Probability Graph 06 Inference Background

Chen Gong

28 November 2019

推断 (Inference) 这个词，对于有一定机器学习基础的同学来说，一定是听说过，这也是贝叶斯方法中一个非常重要的理论性研究。那么什么是推断呢？推断说白了，就是求概率。比如，对于一个联合概率密度函数 $p(x) = p(x_1, x_2, \cdots, x_p)$。我们需要求的有哪些呢?

1. 边缘概率：$p(x_i) = \sum_{x_1} \cdots \sum_{x_{i-1}} \cdots \sum_{x_{i+1}} \cdots \sum_{x_p} p(x)$。

2. 条件概率：$p(x_A|x_B)$，令 $x = x_A \cup x_B$。

3. MAP Inference：也就是 $\hat{z} = \arg\max_z \ p(z|x) \propto \arg\max \ p(x, z)$。因为 $p(z|x) = \frac{p(x,z)}{p(x)} \propto p(x, z)$，我们的目标是求一个最优的参数 z，并不需要知道具体的数值是多少，只要知道谁大谁小就行，所以 $p(x)$ 可以直接不看了。

现在我们知道了，Inference 在求什么？下一步，我们要总结 Inference 有哪些方法。

1 Inference 求解方法

1.1 精准推断 (Deterministic Inference)

Variable Elimination (VE)，变量消除法；Belief Propagation (BP) 信念传播，这个可不是我们之前学习的反向传播算法，这里需要注意。同时这个算法衍生出的 Sum Product Algorithm，这就是推断的核心，这是一种树结构的；而 Junction Tree Algorithm，这是一种普通图结构。

1.2 近似推断 (Approximate Inference)

典型的有有向环 (Loop Belief Propagation)；采样方法，包括 Mente Carlo Inference：Importance Sampling，MCMC；最后一个就是我现在主要研究的变分推断 (Variational Inference)。

2 隐马尔可夫模型 (Hidden Markov Model)

Hidden Markov Model (HMM) 算法将在后面的章节中做详细的描述，在这一小节中，我们主要

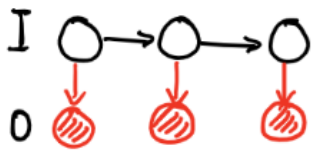

图 1: HMM 的模型结构

做一点概述性的描述。HMM 的模型可视化为上图所示，其中 O 是隐变量，也就是我们的观测变量。

我们主要考虑三个问题，在三个问题为 Inference 的问题。1. Evaluation，也就是求一个边缘密度 $p(O) = \sum_I P(I, O)$。2. Learning，也就是寻找 $\hat{\lambda}$。3. Decoding：$\hat{I} = \arg\max_I P(I|O)$，包括 Vitebi Algorithm，这是一个动态规划算法。而隐马尔可夫模型实际上一种动态规划模型 (Dynamic Bayesian Network)。

Probability Graph 07 Variable Elimination

Chen Gong

07 December 2019

在上一小节中，我们简单的介绍了推断的背景和分类，我们知道了大致有哪些推断的方法。推断的任务可以被我们介绍为：给定已知的 $p(x) = (x_1, x_2, \cdots, x_p)$，我们需要求的有三个：

1. 边缘概率：$p(x_i) = \sum_{x_1,\cdots,x_{i-1},x_{i+1},\cdots,x_p} p(x_1, x_2, \cdots, x_p)$。
2. 条件概率：$p(x_A|x_B)$，也就是在已知 x_B 集合的情况下，如何求得 x_A 集合的概率。
3. 最大后验概率 (MAP)：$\hat{x}_A = \arg\max_{x_A} p(x_A|x_B) = \arg\max_{x_A}\ p(x_A, x_B)$。

下面我们要介绍最简单的一个精确推断中中的东西，名为变量消除法 (Variable Elimination)。这是一种最简单的推断方法，也是我们学习推断法的核心概念之一。下面我们做详细的解释。

1 变量消除法 (Variable Elimination Algorithm)

假如我们有一个马氏链：

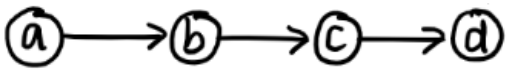

图 1: 一个马氏链的抽象模型

那么我们怎么来求 $p(d)$ 呢？根据公式我们可以得到：

$$p(d) = \sum_{a,b,c} p(a, b, c, d) \tag{1}$$

然后使用因子分解，我们可以得到：

$$p(d) = \sum_{a,b,c} p(a)p(b|a)p(c|b)p(d|c) \tag{2}$$

假定，a, b, c, d 都为均匀离散的二值 random variable，所以 $a, b, c, d \in \{0, 1\}$。
所以，

$$\begin{aligned} p(d) =& p(a=0)\cdot p(b=0|a=0)\cdot p(c=0|b=0)\cdot p(d|c=0) \\ &+p(a=1)\cdot p(b=0|a=1)\cdot p(c=0|b=0)\cdot p(d|c=0) \\ &+\cdots \\ &+p(a=1)\cdot p(b=1|a=1)\cdot p(c=1|b=1)\cdot p(d|c=1) \end{aligned} \tag{3}$$

实际上，这里有 8 个因子的积。那么我们来做进一步的分析，我们可以令

$$\begin{aligned} p(d) &= \sum_{a,b,c} p(a)p(b|a)p(c|b)p(d|c) \\ &= \sum_{b,c} p(c|b)p(d|c) \sum_{a} p(a)p(b|a) \end{aligned} \tag{4}$$

而 $p(a)p(b|a) = p(a,b)$，而 $\sum_a p(a)p(b|a) = p(b)$。我们可以将 a 看成 $\phi(a)$ 这是一个和 a 相关的函数，同理 $p(b|a)$ 看成 $\phi(a,b)$。所以，我们可以将 $\sum_a p(a)p(b|a)$ 看成 $\phi_a(b)$，这样就相当于一个关于 b 的函数，并且是从 a 中导出的。所以，我们做如下替换可得：

$$\sum_{b,c} p(c|b)p(d|c) \sum_{a} p(a)p(b|a) = \sum_{b,c} p(c|b)p(d|c)\phi_a(b) = \sum_{c} p(d|c) \sum_{b} p(c|b)\phi_a(b) \tag{5}$$

同理，我们将 $\sum_b p(c|b)\phi_a(b)$ 看成 $\phi_b(c)$。所以，原始将被改写为：

$$\sum_{c} p(d|c)\phi_b(c) = \phi_c(d) \tag{6}$$

这个算法的核心就是乘法对加法的分配律。那我们怎么类比到乘法的分配律呢？首先先来简单的回顾一下乘法的分配律，也就是 $ac + ab = a(b + c)$。那么我们仔细的来看看这个计算 $p(d)$ 的过程。这是不是就是一个不断的提取公因子，进行计算的过程？有没有觉得和分配律很像？先提取 a 的部分，计算 a 的部分，然后再依次的提取 b 的部分，c 的部分，最后剩下的就是 d 的部分。那么，我们就可以把这么一长串的公式进行逐步化简了，这就是变量消元的思想。同样，在无向图中，我们也可以使用到马尔可夫网络中。

$$p(a,b,c,d) = \frac{1}{z} \prod_{i=1}^{k} \phi_{c_i}(x_{c_i}) \tag{7}$$

写成因子分解的形式就是 $p(x) = \prod_{x_i} \phi_i(x_i)$。这实际上就是分配律，一个变量一个变量的提取，然后进行分解计算。同时这种算法的缺点也非常的明显。

首先，就是重复计算的问题，无论计算那个变量的概率都要重复的计算一遍所有的概率。这个原因就会导致算法的计算难度非常的大。第二个就是计算次序的问题，我们举的例子还比较的简单，所以我们可以一眼就看出来，按 $a - b - c - d$ 的次序开始算。但是，实际上，并没有这么容易就得到计算的次序，而且计算次序不一样会导致计算的难度有很大的区别。而有数学家已经证明了，确定最优的计算顺序，本身就是一个 NP hard 的问题，非常难求解。

Probability Graph 08 Belief Propagation

Chen Gong

08 December 2019

在上一小节中，我们已经介绍了变量消除 (Variable Elimination)，Variable Elimination 的思想是 Probability Graph 中的核心思想之一。上一节中我们就已经介绍了，这实际上就是乘法对加法的分配律。但是，Variable Elimination 中有很多的问题，比如重复计算和最优计算次序不好确定的问题。所以，我们这一节来介绍 Belief Propagation 来解决重复计算的问题。

1 Forward and Backward Algorithm

假设，我们现在有一个马氏链模型：

图 1: 链式马氏链模型结构图

联合概率可以被我们表示为：$p(a,b,c,d,e) = p(a) \cdot p(b|a) \cdot p(c|b) \cdot p(d|c) \cdot p(e|d)$。

如果，我们想要求的是 $p(e)$，那么：

$$\begin{aligned} p(e) &= \sum_{a,b,c,d} p(a,b,c,d,e) \\ &= \sum_{a} p(e|d) \cdot \sum_{c} p(d|c) \cdot \sum_{b} p(c|b) \cdot \sum_{a} p(b|a) \cdot p(a) \end{aligned} \tag{1}$$

为了简化表达，这里我们需要定义一个很重要的符号。因为，$\sum_a p(b|a) \cdot p(a)$，是一个关于 b 的表达式，也就是相当于把 a 给约掉了。所以我们可以把 $\sum_a p(b|a) \cdot p(a)$ 记为 $m_{a \longrightarrow b}(b)$。同理，我们也可以将 $\sum_c p(d|c) \cdot \sum_b p(c|b) \cdot m_{a \longrightarrow b}(b)$ 记为 $m_{b \longrightarrow c}(c)$。那么为了求得 $p(e)$，我们依次的求解顺序为 $m_{a \longrightarrow b}(b)$，$m_{b \longrightarrow c}(c)$，$m_{c \longrightarrow d}(d)$ 和 $m_{d \longrightarrow e}(e)$。也就相当于沿着这个链这个马氏链一直往前走，也就是前向算法 (Forward Algorithm)。我们用公式表达即为：

$$a \overset{m_{a \longrightarrow b}(b)}{\longrightarrow} b \overset{m_{b \longrightarrow c}(c)}{\longrightarrow} c \overset{m_{c \longrightarrow d}(d)}{\longrightarrow} d \overset{m_{d \longrightarrow e}(e)}{\longrightarrow} e \tag{2}$$

如果是要求 $p(c)$，那么我们的传递过程为 $a \longrightarrow b \longrightarrow c \longleftarrow d \longleftarrow e$。这里，我们就不能只用前向算法来解决了，需要用到 Forward-Backward 算法来解决了。也就是同时使用 Forward 和 Backward 的方法，那么我们来看看 $p(c)$ 怎么求?

$$\begin{aligned} p(c) &= \sum_{a,b,d,e} p(a,b,c,d,e) \\ &= (\sum_{b} p(c|b) \sum_{a} p(b|a)p(a)) \cdot (\sum_{d} p(d|c) \sum_{e} p(e|d)) \end{aligned} \tag{3}$$

对比上面的计算 $p(e)$ 的过程，我们就可以发现，$\sum_b p(c|b)\sum_a p(b|a)p(a)$ 部分的计算也就是 $m_{b\longrightarrow c}(c)$ 的计算是一模一样的。所以说，Variable Elimination 里面有大量的重复计算。Belief 的想法很简单，也就是将 $m_{i\longrightarrow j}(j)$，全部事先计算好，就像一个个积木一样，然后再用这个积木来搭建运算。那么也就是，我们事先将方向全部定义好，正向和反向的全部都求了再说。为了进一步探究 Belief Background，我们需要讨论一些更加 Generalize 的情况。也就是从 Chain$\longrightarrow$Tree，有向 $\longrightarrow$ 无项的情况。

2 Belief Propagation 的扩展

我们的 Generalize 的后，分析了一个树形的无向图结构。图的网络结构如下所示：

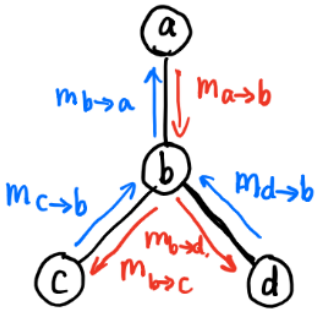

图 2: 树形无向图结构拓扑结构

下面第一步，我们把上面那个模型的设置写出来。所以，我们需要进行因式分解，我们用 $\varphi(i)$ 来表示和 i 有关的部分。所以，我们可以将联合概率密度写为：

$$p(a,b,c,d)=\frac{1}{Z}\varphi_a(a)\varphi_b(b)\varphi_c(c)\varphi_d(d)\cdot\varphi_{a,b}(a,b)\varphi_{b,c}(b,c)\varphi_{b,d}(b,d) \tag{4}$$

我们要求 $p(a)=\sum_{b,c,d}p(a,b,c,d)$ 和 $p(b)=\sum_{a,c,d}p(a,b,c,d)$，其间一定会出现大量的重复计算。这个模型中有四个点，三条边，每条边都有两个方向，所以我们要求的是 6 个 “积木”。我们来一步步的看看，如何可以得到想要的 $p(a)$。

1. 首先，我们需要求的是 $c\longrightarrow b$ 和 $d\longrightarrow b$ 两个过程。其中，$c\longrightarrow b$ 的过程也就是 $m_{c\longrightarrow b}(b)$，可以被我们表达为 $\sum_c\varphi_c\cdot\varphi_{b,c}$。同理，$m_{d\longrightarrow b}(b)$ 可以被我们表达为 $\sum_d\varphi_b\cdot\varphi_{b,d}$。

2. 第二步，我们需要求 $b\longrightarrow a$ 的过程，也就是 $m_{b\longrightarrow a}(a)$。它等于 $m_{c\longrightarrow b}(b),m_{d\longrightarrow b}(b)$ 乘上 b 自己的部分求和得到，我们可以写为：

$$m_{b\longrightarrow a}(a)=\sum_b m_{c\longrightarrow b}(b)\cdot\varphi_b\cdot\varphi_{a,b}\cdot m_{d\longrightarrow b}(b) \tag{5}$$

3. 最后，$m_{b\longrightarrow a}(a)$ 乘上 a 自己的部分就得到了 $p(a)$，也就是：$p(a)=\varphi_a\cdot m_{b\longrightarrow a}(a)$。

所以，我们总结一下：

$$m_{b\longrightarrow a}(x_a)=\sum_{x_b}\varphi_{a,b}\varphi_b m_{c\longrightarrow b}(x_b)m_{d\longrightarrow b}(x_b) \tag{6}$$

而，

$$p(a)=\varphi_a m_{b\longrightarrow a}(x_a) \tag{7}$$

我相信到这里，大家应该是可以理解这个意思的，会有点抽象，但并不是很难。下一步我想做个 Generalize 为了便于大家进行理解，我这里尽量不跳步：

$$m_{b\longrightarrow a}(x_a) = \sum_{x_b} \varphi_{a,b}\varphi_b \prod_{\{k\in NB(b)\}\longrightarrow a} m_{k\longrightarrow b}(x_b) \tag{8}$$

这里的 $NB(b)$ 代表的是所有节点 b 的邻接节点。我们可以进一步表示为：

$$m_{j\longrightarrow i}(x_i) = \sum_{x_j} \varphi_{i,j}\varphi_j \prod_{\{k\in NB(j)\}\longrightarrow i} m_{k\longrightarrow j}(x_j) \tag{9}$$

而，

$$p(x_i) = \varphi_i \prod_{k\in NB(i)} m_{k\Longrightarrow i}(x_i) \tag{10}$$

通过对上面表达式的观察，我们是不是发现了一个很有意思的现象。也就是这些概率都是由 $m_{i\longrightarrow j}$ 这样的小积木拼接起来的。所以我们可以 get a conclusion：我们不要一上来就直接去求边缘概率密度，比如 $p(a), p(b), p(c), p(d)$ 这些的。我们可以先建立一个 Cache，把 $m_{i\longrightarrow j}$ 全部算出来。然后，要求什么的话，直接进行搭建和拼接就可以了。从这里，我们就引出了 Belief Propagation。

3 Belief Propagation

我们之前就已经得到了信息传递的表达式：

$$m_{b\longrightarrow a} = \sum_{b} \varphi_{a,b}\varphi_b \cdot m_{c\longrightarrow b} \cdot m_{d\longrightarrow b} \tag{11}$$

在开始理解 Belief Propagation 之前，我们在分析一下 $m_{b\longrightarrow a}$ 的公式中，每个部分表达的具体含义。其中，$m_{c\longrightarrow b} \cdot m_{d\longrightarrow b}$ 中反映的是孩子 (Children) 的信息。而 φ_b 中表示是 b 节点自己的信息。那么，$\varphi_b \cdot m_{c\longrightarrow b} \cdot m_{d\longrightarrow b}$ 可以被我们看成是 b 节点的所有信息，包括节点自己本身的信息和其他节点传播来的信息，所以我们将这个部分记为：Belief。所以，Belief$(b) = \varphi_b \cdot$ children。b 节点收集孩子和自己的信息，整合和 b 相关的所有信息，通过 Belief$(b) = \varphi_b \cdot$ children 向 a 传去。

那么，Belief Propagation 可以看成是 BP = VE + Cashing。这个算法的核心思想就在于，直接求 $m_{i\longrightarrow j}$，然后再导出边缘概率 $p(x_i)$。第二小节中我们已经详细的给出了 $p(x_i)$ 的推导方法。下一步，我们需要知道如何来求 $m_{i\longrightarrow j}$。

3.1 Sequential Implementation

顺序计算的思路，我们需要借助一个队列来实现：

1. Get Root: 首先我们需要假设一个节点为根节点。

2. Collect Message: 对于每一个在根节点的邻接点中的节点 x_i，Collect Message (x_i)。对应的就是图 2 中蓝色的线条。

3. Distribute Message: 对于每一个在根节点的邻接点中的节点 x_i，Distribute Message (x_i)。对应的就是图 2 中红色的线条。

经过这三个步骤以后，我们可以得到所有的 $i, j \in v$，从而计算出 $p(x_k), k \in v$。

3.2 Parallel Implementation

这种思想在图结构的网络中经常使用，大致也就是随意选一个节点，然后向四周其他的节点辐射信息。其他节点收到信息之后，更新自己的状态，然后反馈自己的信息给源节点，源节点再更新自己的信息。不断地重复这个工作，直到最后收敛为止。

4 小结

实际上，Belief Propagation 就是一种 Variable Elimination。但是，我们发现了 Variable Elimination 中有很多的重复计算。所以，我们想到了提取出来先算好，要用的时候直接放进去就行了。所以，Belief Propagation 中分解了传递的过程，先计算消息传递的机制，再来组装出计算边缘概率。其实本质还是 Variable Elimination 算法，不过就是使表达更加的规范了，通过拆解的方法来消除重复计算。

Probability Graph 09 Max Product Algorithm

Chen Gong

10 December 2019

我们在这里再总结一下概率图模型有什么用。对于一个图，Graph $= \{X, E\}$，其中 X 代表的是普通变量，E 代表的是 Evidence，也就是观测变量。

1. 首先要解决的是边缘变量的问题，也就是已知：$E = \{e_1, e_2, \cdots, e_k\}$，如何求 $p(E)$ 的问题，其中 E 为一个变量或者为一个子集。实际上就是一个 likelihood 的问题。

2. 条件概率，也就是一个求后验概率的问题，目标概率为 $X = (Y, Z)$。而 $p(Y|E) = \sum_z p(X|E)$。

3. 最大后验估计 (MAP)，也被我们称为 Decoding 的问题。也就是我们希望找到一个隐序列，使得：$\hat{x} = argmax_x p(X|E)$，$\hat{y} = argmax_y p(Y|E)$。

这里的 Max-Product 算法和隐马尔可夫模型 (HMM) 中的 Viterbi 算法非常的类似。其实，从算法上讲它就是 Belief Propagation 算法的一种改进，从模型上讲是 Viterbi 算法的推广。在这里我们求的不是概率值了，而是一个最优的概率序列 $(\hat{a}, \hat{b}, \hat{c}, \hat{d}) = argmax_{a,b,c,d} p(x_a, x_b, x_c, x_d|E)$。

1 Max Product Algorithm

下面展示一个树的拓扑结构图。

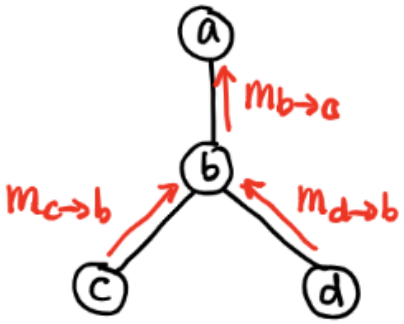

图 1: 概率树模型的拓扑结构图

在这个树中，我们将 $m_{b\longrightarrow a}$ 看成是能使 $p(x_b, x_c, x_d|E)$ 联合概率达到最大的值。每一个节点代表的是到这个节点为止的路径联合概率达到最大的值。我们表达为：

$$m_{j\longrightarrow i} = \max_{x_j} \varphi_i \varphi_{ij} \prod_{k\in\{NB(j)-i\}} m_{k\longrightarrow j} \tag{1}$$

那么，在图一所示的概率图模型中，$m_{c\longrightarrow b}$ 可以表示为：

$$m_{c\longrightarrow b} = \max_{x_c} \varphi_c \cdot \varphi_{bc} \tag{2}$$

其中，$\varphi_c \cdot \varphi_{bc}$ 可以表示为和 c 相关的函数。

$$m_{d \longrightarrow b} = \max_{x_d} \varphi_c \cdot \varphi_{cd} \tag{3}$$

其中，$\varphi_d \cdot \varphi_{cd}$ 可以表示为和 d 相关的函数。

$$m_{b \longrightarrow a} = \max_{x_b} \varphi_b \cdot \varphi_{ab} \cdot m_{c \longrightarrow b} \cdot m_{d \longrightarrow b} \tag{4}$$

最终，我们将得到的是：

$$\max p(a, b, c, d) = \max_{x_a} \varphi_a m_{b \longrightarrow a} \tag{5}$$

而 $\varphi_a m_{b \longrightarrow a}$ 就可以看成是一个关于 a 的函数。这里我们再提一下 Belief Propagation，这里的 Max-Product 实际上就是 Belief Propagation 的一个变形。

2 Belief Propagation

实际上这个算法的提出时因为，多次求边缘概率密度会发现中间有很多的步骤是重复的。我们用 $m_{i \longrightarrow j}$ 记录每一个边缘概率，最后进行组合就行。所以，

$$m_{j \longrightarrow i}(x_i) = \sum_{x_j} \varphi_{i,j}(x_i, x_j) \varphi_j(x_j) \prod_{\{k \in NB(j)\} \longrightarrow i} m_{k \longrightarrow j}(x_j) \tag{6}$$

而：

$$p(x_i) = \varphi(x_i) \prod_{k \in NB(x_i)} m_{k \longrightarrow i}(x_i) \tag{7}$$

3 Compare

其实，我们一比较就可以很简单的看出，Max-product 和 Belief Propagation 只有一个地方不一样。那就是前者是求最大，后者是求和。也就是 Max-product 到 Sum-product。在求得了 $\max p(a, b, c, d) = \max_{x_a} \varphi_a m_{b \longrightarrow a}$ 之后，我们利用回溯法我们比较就可以比较简单的得到 $x_a^*, x_b^*, x_c^*, x_d^*$ 了。在这个算法中，我们就不需要事先计算 $m_{i \longrightarrow j}$ 了，直接在迭代中进行计算就可以了，也不会存在什么重复计算的问题。

Probability Graph 10 Moral Graph & Factor Graph

Chen Gong

11 December 2019

在这一小节中，我们将要介绍两种特殊的概率结构，也就是 Moral Graph 和 Factor Graph。

1 Moral Graph

首先我们需要知道，为什么要有 Moral Graph 的存在？Moral Graph 存在的意义就是将有向图转化为无向图来研究。因为无向图比有向图更加的 Generalize 一些。在概率图中，我们可以分为贝叶斯网络 (有向图) 和马尔可夫网络 (无向图)。

无向图可以表示为：

$$p(x) = \frac{1}{z}\prod_{i=1}^{k}\phi_{c_i}(x_{c_i}) \tag{1}$$

有向图可以表示为：

$$p(x) = \prod_{i=1}^{p} p(x_i|x_{pa(i)}) \tag{2}$$

其中，ϕ_{c_i} 代表的是最大团的意思。通过道德图，我们可以有效的将有向图转换为无向图。

我们看一下如图所示的链式网络：

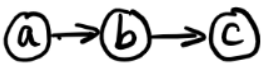

图 1: 链式有向图模型

其中，$p(a,b,c) = p(a)p(b|a)p(c|b)$。如果，把有向图转换成有向图是一件非常简单的事情，首先把所有的线条换成直线。由于在无向图中，我们考虑的是最大团，所以 $p(a)p(b|a) = \varphi(a,b)$，$p(c|b) = \varphi(b,c)$。这个的转换是非常的简单了。

第二种，我们需要讨论的图也就是 Tail to Tail 的图，所下图所示：

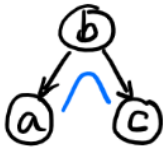

图 2: Tail to Tail 有向图模型

其中，$p(a,b,c) = p(a)p(b|a)p(c|a)$。还是按照一样的套路，首先把所有的有向箭头改成直线。那么我们就可以得到 $p(a)p(b|a) = \phi(a,b)$，$p(c|a) = \phi(a,c)$。其中 $\{a,c\}$ 和 $\{b,c\}$ 是分别属于两个团。这个也比较的简单，但是 Head to Head 的转换就有点不一样了。

第三种，我们需要讨论的图是 Head to Head 的模型，如下图所示：

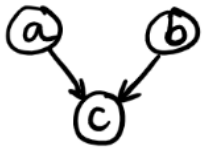

图 3: Head to Head 有向图模型

同样我们使用一样的分析思路来看这个问题，$p(a,b,c) = p(a)p(b)p(c|a,b)$。我们进行拆解的话，只能令 $p(a)p(b)p(c|a,b) = \varphi(a,b,c)$，不然再也找不到其他的拆解方法。但是，如果简单的将模型中所有的有向箭头改成直线得到的并不是一个团。因为 “团” 的概念的要求，团里面的元素都要求是两两相互连接的。所以，我们需要进行改进，将 Head to Head 的无向图形式改进为：

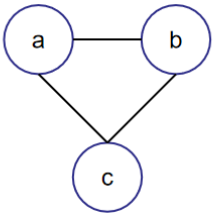

图 4: Head to Head 无向图模型

那么，将 Head to Head 的有向图转换为无向图的过程可以被我们描述为：

对于 $\forall x_i \in G$，将 $parent(x_i)$ 的两个父亲节点连接，然后将 G 中所有的有向边替换成无向边。下面我们举一个例子：

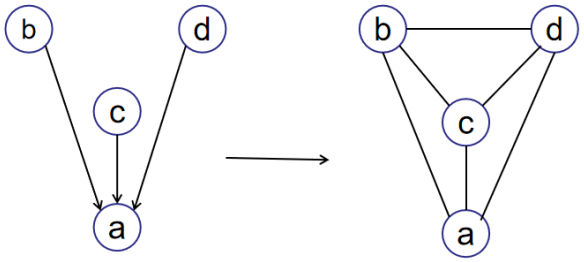

图 5: Head to Head 有向图模型转无向图模型举例

而我们将有向图转换成无向图之后，有什么好处吗？也就是在判断条件独立性的时候，有时图形非常复杂的时候。我们在有向图中很难看出来，而在无向图中却可以很简单的得到我们想要的结果。也就是 $Sep(A,B|C) \Longleftrightarrow D-Sep(A,B|C)$。

2 Factor Graph

在上一小节中，我们介绍了道德图 (Moral Graph)，它的主要作用是将有向图转换为无向图。我们考虑的都是树结构，但是在 Head to Head 结构中，会引入环的结构。但是，在我们的 Belief Propagation (BP) 算法中，只能对树进行分解。所以，这里我们就引入了因子图。因子图主要发挥两个作用：1. 去环，也就是消除无向图中的环结构；2. 使算法变得更加的简洁，简化计算。

如图二表达的那样，他的有向图和无向图的联合概率可以分别表达为：

$$p(a,b,c) = p(a)p(b|a)p(c|a) \qquad p(a,b,c) = \frac{1}{Z}\phi(a,b)\phi(a,c) \tag{3}$$

那什么是因子图分解呢？公式表达可以被我们表示为：

$$p(x) = \prod_S p(x_S) \tag{4}$$

其中，S 是图的节点子集，X_S 为对应的 X 的子集，也就是 X 的随机变量的子集。那么对于一个如图 4 所示的有环无向图，我们怎么进行因子图分解呢？

首先进行第一种分解，如下图所示：

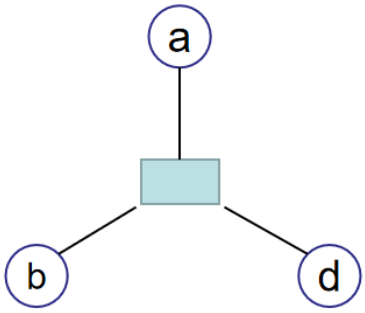

图 6: Head to Head 无向图中心节点因子图分解

这时可以被我们描述为，$f = f(a,b,c)$。或者我们也可以进行更细的分解。如下图所示：

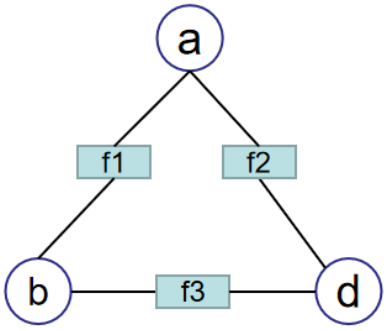

图 7: Head to Head 无向图三节点因子图分解

这个分解的结果可以被我们表示为：

$$p(x) = f_1(a,b)f_2(a,c)f_3(b,c) \tag{5}$$

不仅是可以在两个节点之间插入关系，同时也可以对于单个节点引入函数。

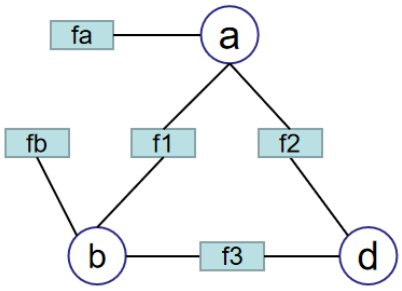

图 8: Head to Head 无向图三节点和独立节点因子图分解

那么这个分解结果可以被我们表示为：

$$p(x) = f_1(a,b)f_2(a,c)f_3(b,c)f_a(a)f_b(b) \tag{6}$$

实际上，就可以看成是对因式分解的进一步分解。这样我们就可以成功的消除环结构。如下图所示：

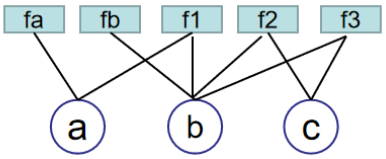

图 9: Head to Head 无向图三节点和独立节点因子图分解

所以，大家仔细一想就知道了因子图存在的意义了，它可以有效的消除环结构，通过一个重构的方式，重建出树的结构。这样可以有效的帮助我们使用 Belief Propagation 中的变量消除法等方法。