第十一章、深度概率图模型

赵涵

2023 年 5 月 23 日

1 受限玻尔兹曼机

玻尔兹曼机是一种存在隐节点的无向图模型。在图模型中最简单的是朴素贝叶斯模型（朴素贝叶斯假设），引入单个隐变量后，发展出了GMM，如果单个隐变量变成序列的隐变量，就得到了状态空间模型（引入齐次马尔可夫假设和观测独立假设就有HMM，Kalman Filter，Particle Filter），为了引入观测变量之间的关联，引入了一种最大熵模型-MEMM，为了克服MEMM中的局域问题，又引入了CRF，CRF是一个无向图，其中，破坏了齐次马尔可夫假设，如果隐变量是链个链式结构，那么又叫线性链CRF。

在无向图的基础上，引入隐变量得到了玻尔兹曼机，这个图模型的概率密度函数是一个指数族分布。对隐变量和观测变量作出一定的限制，就得到了受限玻尔兹曼机（RBM）。

可以看出，不同的概率图模型对下几个个特点作出假设：

1. 方向——边的性质；
2. 离散/连续/混合——点的性质；
3. 条件独立性——边的性质；
4. 隐变量——点的性质；
5. 指数族——结构特点。

将观测变量和隐变量分别记为$v, h, h = \{h_1, \dots, h_m\}, v = \{v_1, \dots, v_n\}$。我们知道，无向图根据最大团的分解，可以写为玻尔兹曼分布的形式$p(x) = \frac{1}{Z}\prod_{i=1}^{K}\psi_i(x_{ci}) = \frac{1}{Z}\exp(-\sum_{i=1}^{K}E(x_{ci}))$，这也是一个指数族分布。

一个玻尔兹曼机存在一系列的问题，在其推断任务中，想要精确推断，是无法进行的，想要近似推断，计算量过大。为了解决这个问题，一种简化的玻尔兹曼机-**受限玻尔兹曼机**（Restrict Boltzmann Machine）作出了假设，所有隐变量内部以及观测变量内部没有连接，只在隐变量和观测变量之间有连接，这样一来：

$$p(x) = p(h, v) = \frac{1}{Z}\exp(-E(v, h)) \tag{1}$$

其中能量函数$E(v, h)$可以写出三个部分，包括与节点集合相关的两项以及与边w相关的一项，记为：

$$E(v, h) = -(h^T wv + \alpha^T v + \beta^T h) \tag{2}$$

带入原方程，展开有：

$$\begin{aligned} p(x) &= \frac{1}{Z}\exp(h^T wv)\exp(\alpha^T v)\exp(\beta^T h) \qquad (3)\\ &= \frac{1}{Z}\prod_{i=1}^{m}\prod_{j=1}^{n}\exp(h_i w_{ij} v_j)\prod_{j=1}^{n}\exp(\alpha_j v_j)\prod_{i=1}^{m}\exp(\beta_i h_i) \qquad (4)\end{aligned}$$

上面这个式子也和RBM的因子图一一对应，用概率图表示（1）如下：

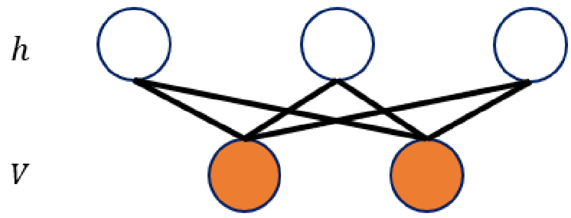

Figure 1: 受限的玻尔兹曼机。

1.1 推断

推断任务包括求后验概率$p(v|h), p(h|v)$，以及求边缘概率$p(v)$。我们先看$p(h|v)$。对于一个无向图，满足局域的Markov性质，即：

$$p(h_1|h - \{h_1\}, v) = p(h_1|Neighbour(h_1)) = p(h_1|v) \tag{5}$$

我们可以得到：

$$p(h|v) = \prod_{i=1}^{m} p(h_i|v) \tag{6}$$

考虑Binary RBM，所有的隐变量只有两个取值$0, 1$：

$$p(h_l = 1|v) = \frac{p(h_l = 1, h_{-l}, v)}{p(h_{-l}, v)} = \frac{p(h_l = 1, h_{-l}, v)}{p(h_l = 1, h_{-l}, v) + p(h_l = 0, h_{-l}, v)} \tag{7}$$

将能量函数写成和l相关或不相关的两项：

$$E(v, h) = -(\sum_{i=1, i\neq l}^{m} \sum_{j=1}^{n} h_i w_{ij} v_j + h_l \sum_{j=1}^{n} w_{ij} v_j + \sum_{j=1}^{n} \alpha_j v_j + \sum_{i=1, i\neq l}^{m} \beta_i h_i + \beta_l h_l) \tag{8}$$

定义：

$$h_l H_l(v) = h_l \sum_{j=1}^{n} w_{ij} v_j + \beta_l h_l \tag{9}$$

$$\overline{H}(h_{-l}, v) = \sum_{i=1, i\neq l}^{m} \sum_{j=1}^{n} h_i w_{ij} v_j + \sum_{j=1}^{n} \alpha_j v_j + \sum_{i=1, i\neq l}^{m} \beta_i h_i \tag{10}$$

把定义好的带入贝叶斯公式，有：

$$\begin{aligned} p(h_l = 1|v) &= \frac{\exp(H_l(v) + \overline{H}(h_{-l}, v))}{\exp(H_l(v) + \overline{H}(h_{-l}, v)) + \exp(\overline{H}(h_{-l}, v))} && (11) \\ &= \frac{1}{1 + \exp(-H_l(v))} = \sigma(H_l(v)) && (12) \end{aligned}$$

于是就得到了后验概率。对于v的后验是对称的，所以类似的可以求解。接下来求边缘概率分布$p(v)$，有：

$$\begin{aligned} p(v) &= \sum_h p(h, v) = \sum_h \frac{1}{Z} \exp(h^T x v + \alpha^T v + \beta^T h) && (13) \\ &= \exp(\alpha^T v) \frac{1}{Z} \sum_{h_l} \exp(h_1 w_1 v + \beta_1 h_1) \ldots \sum_{h_m} \exp(h_m w_m v + \beta_m h_m) && (14) \\ &= \exp(\alpha^T v) \frac{1}{Z} (1 + \exp(w_1 v + \beta_1)) \ldots (1 + \exp(w_m v + \beta_m)) && (15) \\ &= \frac{1}{Z} \exp(\alpha^T v + \sum_{i=1}^{m} \log(1 + \exp(w_i v + \beta_i))) && (16) \end{aligned}$$

其中，$\log(1 + \exp(x))$叫做Softplus 函数。

2 配分函数的逼近

在学习和推断中，对于一个概率的归一化因子很难处理，这个归一化因子和配分函数相关。假设一个概率分布：

$$p(x|\theta)=\frac{1}{Z(\theta)}\hat{p}(x|\theta),Z(\theta)=\int\hat{p}(x|\theta)dx \tag{17}$$

2.1 包含配分函数的MLE

在学习任务中，采用最大似然：

$$\begin{aligned}\hat{\theta} &= \arg\max_{\theta}p(x|\theta)=\arg\max_{\theta}\sum_{i=1}^{N}\log p(x^{(i)}|\theta) && (18)\\ &= \arg\max_{\theta}\sum_{i=1}^{N}\log\hat{p}(x|\theta)-N\log Z(\theta) && (19)\\ &= \arg\max_{\theta}\frac{1}{N}\sum_{i=1}^{N}\log\hat{p}(x|\theta)-\log Z(\theta) && (20)\\ &= \arg\max_{\theta}\ell(\theta) && (21)\end{aligned}$$

对参数求梯度，有：

$$\begin{aligned}\nabla_{\theta}\log Z(\theta) &= \frac{1}{Z(\theta)}\nabla_{\theta}Z(\theta) && (22)\\ &= \frac{p(x|\theta)}{\hat{p}(x|\theta)}\int\nabla_{\theta}\hat{p}(x|\theta)dx && (23)\\ &= \int\frac{p(x|\theta)}{\hat{p}(x|\theta)}\nabla_{\theta}\hat{p}(x|\theta)dx && (24)\\ &= E_{p(x|\theta)}[\nabla_{\theta}\log\hat{p}(x|\theta)] && (25)\end{aligned}$$

由于这个表达式和未知的概率相关，于是无法直接精确求解，需要近似采样，如果没有这一项，那么可以采用梯度下降，但是存在配分函数就无法直接采用梯度下降了。这个期望值，是对模型假设的概率分布，定义真实概率分布为p_{data}，于是，$\ell(\theta)$中的第一项的梯度可以看成是从这个概率分布中采样出来的N 个点求和平均，可以近似期望值，即：

$$\nabla_{\theta}\ell(\theta)=E_{p_{data}}[\nabla_{\theta}\log\hat{p}(x|\theta)]-E_{p(x|\theta)}[\nabla_{\theta}\log\hat{p}(x|\theta)] \tag{26}$$

于是，相当于真实分布和模型假设越接近越好。上面这个式子第一项叫做正相，第二项叫做负相。为了得到负相的值，需要采用各种采样方法，如MCMC。假设采样得到了样本$\hat{x}^{1:m}\sim p_{model}(x|\theta^{t})$，那么：

$$\theta^{t+1}:=\theta^{t}+\eta(\sum_{i=1}^{m}\nabla_{\theta}\log\hat{p}(x^{(i)}|\theta^{t})-\sum_{i=1}^{m}\nabla_{\theta}\log\hat{p}(\hat{x}^{(i)}|\theta^{t})) \tag{27}$$

这个算法也叫做基于MCMC采样的梯度上升。每次通过采样得到的样本叫做幻想粒子，如果这些幻想粒子区域的概率高于实际分布，那么最大化参数的结果就是降低这些部分的概率。

2.2 对比散度——CD Learning

上面对于负相的采样，最大的问题是，采样到达平稳分布的步骤数量是未知的。对比散度的方法，是对上述的采样是的初始值作出限制，直接采样$\hat{x}^{(i)}=x^{(i)}$，这样可以缩短采样的混合时间。这个算法

叫做CD-k算法，k就是初始化后进行的演化时间，很多时候，即使$k=1$也是可以的。我们看MLE的表达式：

$$
\begin{aligned}
\hat{\theta} &= \arg\max_{\theta} p(x|\theta) && (28)\\
&= \arg\max_{\theta} \frac{1}{N}\sum_{i=1}^{N} \log p(x^{(i)}|\theta) && (29)\\
&= E_{p_{data}}[\log p_{model}(x|\theta)] && (30)\\
&= \arg\max_{\theta} \int p_{data} \log p_{model} dx && (31)\\
&= \arg\max_{\theta} \int p_{data} \log \frac{p_{model}}{p_{data}} dx && (32)\\
&= \arg\min_{\theta} KL(p_{data}, p_{model}) && (33)
\end{aligned}
$$

对于CD-k的采样过程，可以将初始值这些点表示为：

$$p^0 = p_{data} \tag{34}$$

而我们的模型需要采样过程达到平稳分布：

$$p^{\infty} = p_{model} \tag{35}$$

因此，我们需要的是$KL(p^0, p^{\infty})$。定义CD：

$$KL(p^0, p^{\infty}) - KL(p^k, p^{\infty}) \tag{36}$$

这就是CD-k算法第k次采样的目标函数。

2.3 RBM的学习问题

RBM的参数为：

$$
\begin{aligned}
h &= (h_1, \ldots, h_m)^T && (37)\\
v &= (v_1, \ldots, v_n)^T && (38)\\
w &= (w_{ij})_{m\times n} && (39)\\
\alpha &= (\alpha_1, \ldots, \alpha_n)^T && (40)\\
\beta &= (\beta_1, \ldots, \beta_m)^T && (41)
\end{aligned}
$$

学习问题关注的概率分布为：

$$
\begin{aligned}
\log p(v) &= \log \sum_h p(h, v) && (42)\\
&= \log \sum_h \frac{1}{Z} \exp(-E(v, h)) && (43)\\
&= \log \sum_h \exp(-E(v, h)) - \log \sum_{v,h} \exp(-E(h, v)) && (44)
\end{aligned}
$$

对上面这个式子求参数的梯度，第一项为：

$$
\begin{aligned}
\frac{\partial}{\partial\theta}\log\sum_h \exp(-E(v,h)) &= -\frac{\sum_h[\exp(-E(h,v))\frac{\partial E(v,h)}{\partial\theta}]}{\sum_h \exp(-E(h,v))} && (45)\\
&= -\sum_h \frac{\exp(-E(h,v))\frac{\partial E(v,h)}{\partial\theta}}{\sum_h \exp(-E(h,v))} && (46)\\
&= -\sum_h p(h|v)\frac{\partial E(v,h)}{\partial\theta} && (47)
\end{aligned}
$$

对第二项求参数的梯度，有：

$$
\begin{aligned}
\frac{\partial}{\partial\theta}\log\sum_{v,h} \exp(-E(h,v)) &= -\sum_{h,v} \frac{\exp(-E(v,h))\frac{\partial E(v,h)}{\partial\theta}}{\sum_{v,h} \exp(-E(h,v))} && (48)\\
&= -\sum_{v,h} p(v,h)\frac{\partial E(v,h)}{\partial\theta} && (49)
\end{aligned}
$$

所以有：

$$\frac{\partial}{\partial\theta}\log p(v) = -\sum_h p(h|v)\frac{\partial E(v,h)}{\partial\theta} + \sum_{v,h} p(v,h)\frac{\partial E(v,h)}{\partial\theta} \tag{50}$$

将RBM的模型假设代入：

$$E(v,h) = -(h^T wv + \alpha^T v + \beta^T h) \tag{51}$$

通过对w_{ij}求梯度作为例子，得到如下的结果：

$$\frac{\partial}{\partial w_{ij}}E(v,h) = -h_i v_j \tag{52}$$

于是有：

$$\frac{\partial}{\partial\theta}\log p(v) = \sum_h p(h|v)h_i v_j - \sum_{h,v} p(h,v)h_i v_j \tag{53}$$

第一项：

$$\sum_{h_1,h_2,\ldots,h_m} p(h_1,h_2,\ldots,h_m|v)h_i v_j = \sum_{h_i} p(h_i|v)h_i v_j = p(h_i=1|v)v_j \tag{54}$$

这里利用了h_i是二元变量。

第二项：

$$\sum_{h,v} p(h,v)h_i v_j = \sum_{h,v} p(v)p(h|v)h_i v_j = \sum_v p(v)p(h_i=1|v)v_j \tag{55}$$

这个求和是指数阶的，于是需要采样解决，可以使用CD-k方法。

对于第一项，可以直接使用训练样本得到，第二项采用CD-k 采样方法，首先使用样本$v^0 = v$，然后采样得到h^0，然后采样得到v^1，这样顺次进行，最终得到v^k，对于每个样本都得到一个v^k，最终采样得到N 个v^k，于是第二项就是：

$$p(h_i = 1|v^k)v_j^k \tag{56}$$

具体的算法为：

1. 对每一个样本中的v，进行采样：
   (a) 使用这个样本初始化采样；
   (b) 进行k次采样（$0 : k-1$）：

i. $h_i^l \sim p(h_i|v^l)$

ii. $v_i^{l+1} \sim p(v_i|h^l)$

(c) 将这些采样出来的结果累加进梯度中；

2. 重复进行上述过程，最终的梯度除以N。

2.4 近似推断

这里讲中的近似推断具体描述在深度生成模型中的近似推断。推断的目的有下几个部分：

- 推断本身，根据结果（观测）得到原因（隐变量）；
- 为参数的学习提供帮助。

但是推断本身是一个困难的任务，计算复杂度往往很高，对于无向图，由于节点之间的联系过多，那么因子分解很难进行，并且相互之间都有耦合，于是很难求解，仅仅在某些情况如RBM中可解，在有向图中，常常由于条件独立性问题，如两个节点之间**条件相关**（explain away），于是求解这些节点的条件概率就很困难，仅仅在某些概率假设情况下可解如高斯模型，于是需要近似推断。事实上，我们常常讲推断问题变为优化问题，即：

$$Log-likelihood:\sum_{v\in V}\log p(v) \tag{57}$$

对上面这个问题，由于：

$$\log p(v)=\log\frac{p(v,h)}{p(h|v)}=\log\frac{p(v,h)}{q(h|v)}+\log\frac{q(h|v)}{p(h|v)} \tag{58}$$

左右两边对h积分，左边有：

$$\int_h \log p(v)q(h|v)dh=\log p(v) \tag{59}$$

右边积分有：

$$E_{q(h|v)}[\log\frac{p(v,h)}{q(h|v)}]+KL(q(h|v),p(h|v))=E_{q(h|v)}[\log p(v,h)]+H(q)+KL(q,p) \tag{60}$$

其中前两项是ELBO，于是这就变成一个优化ELBO的问题。

3 Sigmoid信念网络

通过概率图来表示Sigmoid信念网络，有如图（2）的形式：每一个节点，都是二值的，即只能取$0,1$，这个模型是由Neal在1990年提出来的。这个模型只要足够深，就可以逼近非常复杂的二值分布。模型每条边的权重为w_{ji}（模型还应该存在一个偏置，但是不影响整个模型的推导，我们省略了），那么其中任意一个节点后验概率分布为：

$$p(S_i=1|S_{j:j<i})=\sigma(\sum_{j<i}w_{ji}S_j) \tag{61}$$

这里的S_j是节点S_i的父节点。那么另一个取值为0的概率值为：

$$\begin{aligned}p(S_i=0|S_{j:j<i}) &= 1-p(S_i=1) &(62)\\ &= \sigma(-\sum_{j<i}w_{ji}S_j) &(63)\end{aligned}$$

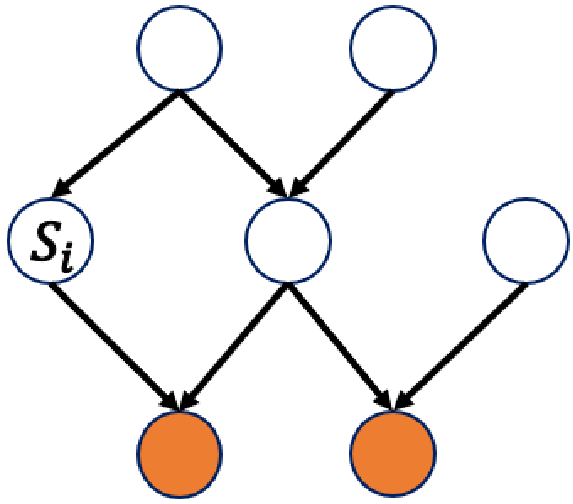

Figure 2: Sigmoid Belief Network的概率图架构。

第二个等号是把$\sigma(\cdot)$函数展开计算得到的。把上面两个方程，综合在一起，引入记号$S_i^\star = 2S_i - 1$，那么我们有：

$$p(S_i|S_{j:j<i}) = \sigma(S_i^\star \sum_{j<i} w_{ji} S_j) \tag{64}$$

这个模型的后验概率分布的计算是非常复杂的，因为存在相消解释（即有头对头的概率图）。我们接下来，写出来模型的对数似然函数和对应的梯度。我们以上面的7 个节点的图为例，分为两组：$S = \{S_1, S_2, \ldots, S_T\} = \{v, h\} = \{V, h^{(1)}, h^{(2)}\}$。

所有节点的联合概率分布为：

$$p(S) = \prod_i p(S_i|S_{j:j<i}) \tag{65}$$

那么对数似然函数为：

$$ell(w) = \sum_{v \in V} \log p(v) \tag{66}$$

这里的v为第一层的可见随机变量。那么我们对上式的参数，求梯度，梯度进入到求和号内，有：

$$\begin{aligned}
\frac{\partial}{\partial w_{ji}} \log p(v) &= \frac{1}{p(v)} \frac{\partial p(v)}{\partial w_{ji}} && (67)\\
&= \frac{1}{p(v)} \frac{\partial \sum_h p(v,h)}{\partial w_{ji}} && (68)\\
&= \sum_h \frac{1}{p(v)} \frac{\partial p(v,h)}{\partial w_{ji}} && (69)\\
&= \sum \frac{p(h|v)}{p(h,v)} \frac{\partial p(v,h)}{\partial w_{ji}} && (70)\\
&= \sum \frac{p(h|v)}{p(S)} \frac{\partial p(S)}{\partial w_{ji}} && (71)
\end{aligned}$$

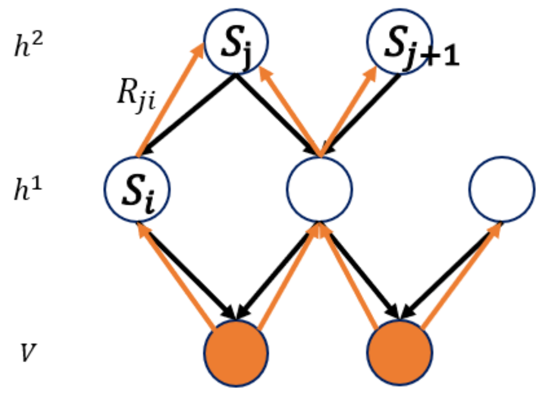

Figure 3: 醒眠算法对概率图的修改。

我们继续对上式进行化简，把$p(S)$的表达式带入，有：

$$
\begin{aligned}
\frac{1}{p(S)}\frac{\partial p(S)}{\partial w_{ji}} &= \frac{1}{\prod_k p(S_k|S_{j:j<k})}\frac{\Delta_t \partial p(S_i|S_{j:j<i})}{\partial w_{ji}} && (72)\\
&= \frac{1}{p(S_i|S_{j:j<i})\Delta_t}\frac{\Delta_t \partial p(S_i|S_{j:j<i})}{\partial w_{ji}} && (73)\\
&= \frac{1}{p(S_i|S_{j:j<i})}\sigma(S_i^\star \sum_{j<i} w_{ji}S_j)\sigma(-S_i^\star \sum_{j<i} w_{ij}S_j)S_i^\star S_j && (74)\\
&= \sigma(-S_i^\star \sum_{j<i} w_{ij}S_j)S_i^\star S_j && (75)
\end{aligned}
$$

Δ_t为其他六个节点的联合概率分布。最后整合在一起，有：

$$
\frac{\partial}{\partial w_{ji}}\sum_{v\in V}\log p(v) = \sum_{v\in V}\sum_{S} p(S|v)\sigma(-S_i^\star \sum_{j<i} w_{ij}S_j)S_i^\star S_j \tag{76}
$$

这里我们使用了$p(S|v) = p(h,v|v) = p(h|v)$ 的一个技巧。这里的$p(S|v)$是无法求出来的，退而求其次，就是采用MCMC 的方法，即$E_{(v,h)\sim p(S|v),v\sim p_v}[\cdot]$。但是很明显，网络规模一旦很大，计算开销巨大，换句话说，MCMC方法只适用于SBN网络节点很少的时候。

3.1 醒眠算法（Wake-Sleep Algorithm）

平均场方法是最初级的近似方法，把联合概率分布，近似为单个节点的边缘概率分布的连乘。但是由于网络变大，同样也变得计算量巨大。这一节，我们介绍Hinton 在1995年提出的醒眠算法。

Hinton认为，可以把SBN的每条有向边，复制一条与之反向的边，如图（3）所示： 把正向边称为Generative Connection，反向边称为Recognization Connection。算法分为两个阶段，第一个阶段称为Wake Phase，先从显层，逐步向上激活神经元，称为Bottom-up，其实就是从可见变量向上获得隐变量的样本（可见变量的样本就是训练数据）；得到样本后，从上到下学习w_{ji}，即求权重。第二个阶段称为Sleep Phase，从上到下（Top-down）采样获得样本，此时采样并不困难，因为没有训练数据，并且顶层的节点是互相独立的，换句话说，从上到下采样得到的样本，是虚拟样本；有了样本，那么就可以学习权重R_{ji} 了，即求反向权重R。其实它的目的就是想通过反向边的构造，去近似后验分布$p(h|v)$，醒眠算法是一个精度不算高的算法，但是效率很高，很有启发意义。我们接下来，就看一下，醒眠算法每一步的目标函数与细节。

从上到下，作为一个生成过程，那么概率分布很容易就能写出来，就是所有节点的联合概率分布，即$p_\theta(v,h)$，这里用θ 代指正向权重W。从下到上，是一个后验过程，那么目标函数为$q_\phi(h|v)$，这

里用ϕ代指R。第一步的Wake Phase的目标函数，就有：

$$E_{q_\phi(h|v)}[\log p_\theta(v,h)] \tag{77}$$

从目标函数可以看出，从下到上采样，然后从上到下去学习。那么采用MLE方法，有：

$$\hat{\theta} = \arg\max_\theta E_{q_\phi(h|v)}[\log p_\theta(h,v)], \text{ with } \phi \text{ fixed.} \tag{78}$$

我们可以和之前学习EM算法的公式来比较，发现这里的目标函数就是ELBO。反过来说Sleep Phase的目标函数，有如下的表达：

$$\hat{\phi} = \arg\max_\phi E_{p_\theta(h,v)}[\log q_\phi(h|v)] \tag{79}$$

把这个期望，写成积分的形式，有：

$$\begin{aligned}
\hat{\phi} &= \arg\max_\phi \int p_\theta(v)p_\theta(h|v)\log q_\phi(h|v)dh &(80)\\
&= \arg\max_\phi \int p_\theta(h|v)\log q_\phi(h|v)dh &(81)\\
&= \arg\max_\phi \int p_\theta(h|v)\log(\frac{q_\phi(h|v)}{p_\theta(h|v)}p_\theta(h|v))dh &(82)\\
&= \arg\max_\phi \int p_\theta(h|v)\log(\frac{q_\phi(h|v)}{p_\theta(h|v)})dh &(83)\\
&= \arg\min_\phi KL(p_\theta(h|v), q_\theta(h|v)) &(84)
\end{aligned}$$

第一个等号，使用了贝叶斯规则，把联合概率分布写成了条件概率分布与边缘概率的乘积；第二个与第四个等号，都是去掉了与ϕ无关的项。我们继续可以与EM 算法比较，发现这个KL散度，是不一样的，与EM算法对应的正好相反。醒眠算法两个阶段的目标函数是不一样的，即Wake Phase对应的最大化ELBO，等价于最小化$KL(q_\theta(h|v), p_\theta(h|v))$。目标函数的不一致，导致算法并不能保证精确性，因为从上到下的采样过程，都是纯粹虚拟的样本，就像睡着了一样，所以称为Sleep Phase。

4 深度信念网络

深度信念网络（Deep Belief Network）是Hinton在2006年发表的工作。在同时期来说，比最流行的SVM要好，但是希望是巨大的，把深度学习的连接主义推上了历史舞台。今天这个模型已经被淘汰了，但是作为学习机器学习的阶段，还是很有重要意义的。DBN 的概率图如图（4）所示： 我们可以看出，模型分为两部分，是一个复合模型。我们给出所有节点的联合概率分布：

$$p(v,h^1,h^2,h^3) = p(v|h^1,h^2,h^3)p(h^1,h^2,h^3) \tag{85}$$

根据图的性质，可以化简为：

$$\begin{aligned}
&= p(v|h^3)p(h^1,h^2,h^3) &(86)\\
&= p(v|h^1)p(h^1|h^2)p(h^2,h^3) &(87)\\
&= \prod_i p(v_i|h^1)\prod_j p(h_j^1|h^2)p(h^2,h^3) &(88)
\end{aligned}$$

第二个等号是因为h^1与h^3无关，第三个等号是因为上层的被观测后，下层的节点相互独立。接下来我们把Sigmoid网络的函数表示写出来，有：

$$p(v_i|h^1) = sigmoid(w_{:,i}^{1^T} h^1 + b_i^0) \tag{89}$$

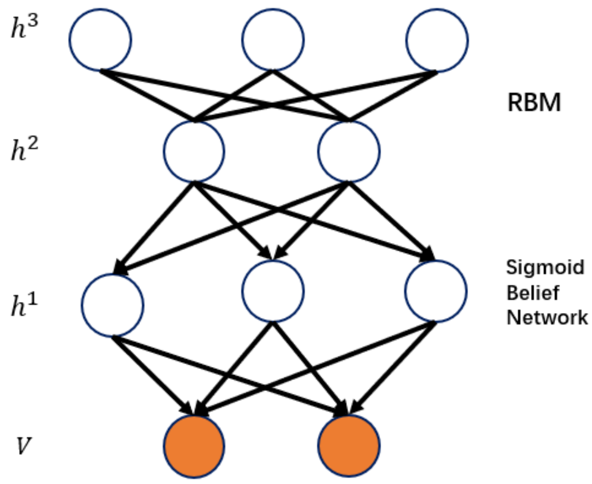

Figure 4: 深度信念网络对应的概率图模型。

$w_{:,i}$表示一个列向量，即第i列的元素组成的向量。下面写出第二层的条件概率：

$$p(h_i^1|h^2) = sigmoid(w_{:,i}^{2^T} h^2 + b_i^1) \tag{90}$$

最后再给出RBM的概率：

$$p(h^2, h^3) = \frac{1}{Z}\exp(h^{3^T} W^3 h^2 + h^{2^T} b^2 + h^{3^T} b^3) \tag{91}$$

这个模型为什么这么设计？这是我们接下来要解决的问题。我们先看RBM（1），对于隐层来说，我们继续堆叠一个RBM，去学习第一个隐层的节点，我们有直觉认为，堆叠的要比一般的效果要好。理论上，可以无限的叠加，但是层数越多，计算量越大，所以要均衡效果与开销。我们看显层的的概率分布：

$$\begin{aligned} p(v) &= \sum_{h^1} p(v, h^1) && (92) \\ &= \sum_{h^1} p(h^1)p(v|h^1) && (93) \end{aligned}$$

这里假设固定住$p(v|h^1)$，所以受限的玻尔兹曼机，必须去掉向上的方向，因为向上的方向导致在学习过程中，会影响下面的权重，所以DBN可以看成是RBM的堆叠。那么从这个角度出发，分析为什么DBN 效果比RBM的效果要好。

首先给出对数似然函数：

$$\log p(v) = \log \sum_{h^1} p(v, h^1) \tag{94}$$

我们引入一个后验分布$q(h^1|v)$，上式可以写成：

$$\begin{aligned} &= \log \sum_{h^1} q(h^1|v)\frac{p(h^1, v)}{q(h^1|v)} && (95) \\ &= \log E_{q(h^1|v)}[\frac{p(h^1, v)}{q(h^1|v)}] && (96) \\ &\geq E_{q(h^1|v)}[\log \frac{p(h^1, v)}{q(h^1|v)}] && (97) \\ &= \sum_{h^1} q(h^1|v)[\log p(h^1, v) - \log q(h^1|v)] && (98) \end{aligned}$$

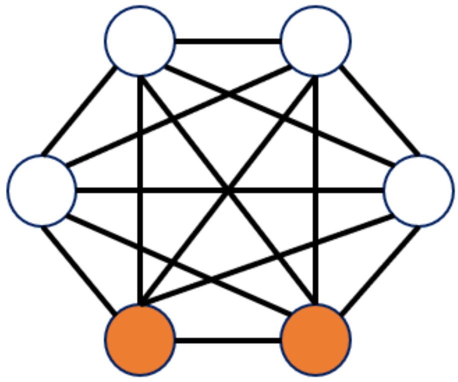

Figure 5: 玻尔兹曼机的概率图表示。

那么DBN加了一层之后，那么$\log p(h^1 v)$就可以写成：

$$\log p(h^1, v) = \log p(h^1) + \log p(v|h^1) \tag{99}$$

把这个等式带入到公式（98），有：

$$= \sum_{h^1} q(h^1|v) \log p(h^1) + Const. \tag{100}$$

然后DBN对$p(h^1)$再次进行建模，即又加了一层，所以ELBO，又增加了。换句话说，DBN 的效果比RBM的效果要好。

我们可以对第一个隐层取近似，有：

$$p(h^1|v) \simeq q(h^1|v) = \prod_i q(h_i^1|v) = \prod_i sigmoid(w_{i,:}^1 + b_i^1) \tag{101}$$

关于学习分为两步：

- 预训练（Pre-training，堆叠RBM）；
- 微调（Fine-tuning）：醒眠算法或者BP。

里面的细节，建议看Hinton的论文《A fast learning algorithm for Deep Belief Network》。

5 玻尔兹曼机

玻尔兹曼机是一般模型，概率图的表示（5）如下： 模型的节点分为两类，一类是可观测节点$V = \{0,1\}^D$，隐藏节点$h = \{0,1\}^P$。这里的D和P分别表示节点的个数。模型的参数分为三种：

1. $L = [L_{ij}]_{D\times D}$表示可观测节点之间连接的权重；
2. $J = [J_{ij}]_{P\times P}$表示隐藏节点之间连接的权重；
3. $W = [W_{ij}]_{D\times P}$表示的是可见节点与隐藏节点之间连接的权重。

模型的联合概率分布为：

$$p(v,h) = \frac{1}{Z}\exp(-E(v,h)) \tag{102}$$

$$E(v,h) = -(v^T W h + \frac{1}{2}v^T L v + \frac{1}{2}h^T J h) \tag{103}$$

那么还是按照流程，写出对数似然函数，并求得参数梯度。假设有N个可见节点。先写出对数似然函数：

$$\ell(\theta) = \frac{1}{N}\sum_{v\in V}\log p(v) \tag{104}$$

假定参数用θ代替，那么对对数似然函数求梯度，有：

$$\frac{\partial}{\partial\theta}\frac{1}{N}\sum_{v\in V}\log p(v) = \frac{1}{N}\sum_{v\in V}\frac{\partial\log p(v)}{\partial\theta} \tag{105}$$

那么我们把里面的偏导数，之前我们已经推导过了，这里直接写出来：

$$\frac{\partial\log p(v)}{\partial\theta} = \sum_{v,h}p(v,h)\frac{\partial E(v,h)}{\partial\theta} - \sum_{h}p(h|v)\frac{\partial E(v,h)}{\partial\theta} \tag{106}$$

我们这里看其中对W的梯度：

$$\frac{\partial\log p(v)}{\partial W} = \sum_{v,h}p(v,h)(-Vh^T) + \sum_{h}p(h|v)(Vh^T) \tag{107}$$

把这部分带入到整体的参数求导式子当中，就得到了最终的梯度，值得注意的是，第一项的求和出现了重复，这里进行一个化简，去掉一个对V的求和号，这里就不在重复了。直接给出结果：

$$E_{p_{data}}[Vh^T] - E_{p_{model}}[Vh^T] \tag{108}$$

这里的$p_{data} \sim p_{data}(v,h) = p_{data}(v)p_{model}(h|v), p_{model} = p_{model}(v,h)$，第一项称为**正相**（positive phase），第二项称为**负相**（negative phase）。有了这些，就可以使用随机梯度上升更新参数了。那么这里的期望怎么计算，参看《深度学习》里的公式。有：

$$\begin{aligned} p(v_i = 1|h, V_{-i}) &= \sigma(\sum_{j=1}^{P} w_{ij}h_j + \sum_{k=1\imath}^{D} L_{ik}v_k) \qquad (109) \\ p(h_j = 1|h_{-j}) &= \sigma(\sum_{i=1}^{D} w_{ij}v_i + \sum_{m=1\jmath}^{P} J_{im}h_m) \qquad (110) \end{aligned}$$

很明显，有了这些条件概率，离散型随机变量，首选MCMC采样方法，逼近期望值。[1]

下面我们采用另一种近似方法——变分推断，绕过MCMC采样步骤。ELBO的表达式如下：

$$ELBO = \log p_\theta(v) - KL(q_\phi, p_\theta) = \sum_{h} q_\phi(h|v)\log p_\theta(v,h) + H[q] \tag{111}$$

对$q_\phi(h|v)$取近似，有：

$$q_\phi(h|v) = \prod_{j=1}^{P} q_\phi(h_j|v) \tag{112}$$

因为是二值分布，所以有：

$$q_\phi(h_j = 1|v) = \phi_j \tag{113}$$

$$\phi = \{\phi_j\}_{j=1}^{P} \tag{114}$$

[1]这两个条件概率公式，我们再此不做推导，相对来说，有些复杂，具体的细节，参看《深度学习》的相关章节（20.1,20.4.1,20.4.2,20.4.3），也可以参看白板推导机器学习的玻尔兹曼机视频。

再把联合概率分布带入，有：

$$\hat{\theta}_j = \arg\max_{\phi} ELBO \tag{115}$$

$$= \arg\max_{\phi} \sum_h q_\phi(h|v)[-\log Z + V^T Wh + \frac{1}{2}V^T LV + \frac{1}{2}h^T Jh] + H[q] \tag{116}$$

$$= \arg\max_{\phi} \sum_h q_\phi(h|v)[-\log Z + \frac{1}{2}V^T LV] + \sum_h q_\phi(h|v)[V^T Wh + \frac{1}{2}h^T Jh] + H[q] \tag{117}$$

$$= \arg\max_{\phi} \sum_h q_\phi(h|v)V^T Wh + \frac{1}{2}q_\phi(h|v)h^T Jh + H[q] \tag{118}$$

最后一个等号，是因为在$\arg\max$时，都与ϕ无关。把这三项进行编号，分别计算：

$$[1] = \sum_h q_\phi(h|v) \sum_{i=1}^{D}\sum_{j=1}^{P} v_i w_{ij} h_j \tag{119}$$

$$= \sum_h \prod_{\hat{j}=1}^{P} q_\phi(h_{\hat{j}}|v) \sum_{i=1}^{D}\sum_{j=1}^{P} v_i w_{ij} h_j \tag{120}$$

我们看其中一项，有：

$$\sum_h \prod_{\hat{j}=1}^{P} q_\phi(h_{\hat{j}}|v) v_1 w_{12} h_2 = \sum_{h_2} q_\phi(h_2|v) v_1 w_{12} h_2 \sum_{h\backslash h_2} \prod_{\hat{j}=1\backslash 2}^{P} q_\phi(h_{\hat{j}}|v) \tag{121}$$

$$= \sum_{h_2} q_\phi(h_2|v) v_1 w_{12} h_2 \tag{122}$$

$$= q_\phi(h_2 = 1|v) v_1 w_{12} \tag{123}$$

$$= \phi_2 v_1 w_{12} \tag{124}$$

找到以上的规律，那么总的结果为：

$$[1] = \sum_{i=1}^{D}\sum_{\hat{j}=1}^{P} \phi_{\hat{j}} v_i w_{i\hat{j}} \tag{125}$$

那么第二项，第三项就不再推导了。直接给出结果，有：

$$[2] = \sum_{j=1}^{P}\sum_{m=1\backslash j}^{P} \phi_j \phi_m J_{jm} \tag{126}$$

$$[3] = -\sum_{j=1}^{P}[\phi_j \log\phi_j + (1-\phi_j)\log(1-\phi_j)] \tag{127}$$

然后让这三项分别对ϕ_j求偏导数，有：

$$\frac{\partial}{\partial\phi_j}[1] = \sum_{i=1}^{D} v_i W_{ij} \tag{128}$$

$$\frac{\partial}{\partial\phi_j}[2] = \sum_{m=1\backslash j}^{P} \phi_m J_{jm} \tag{129}$$

$$\frac{\partial}{\partial\phi_j}[3] = -\log\frac{\phi_j}{1-\phi_j} \tag{130}$$

最后令：

$$\frac{\partial}{\partial\phi_j}([1]+[2]+[3]) = 0 \tag{131}$$

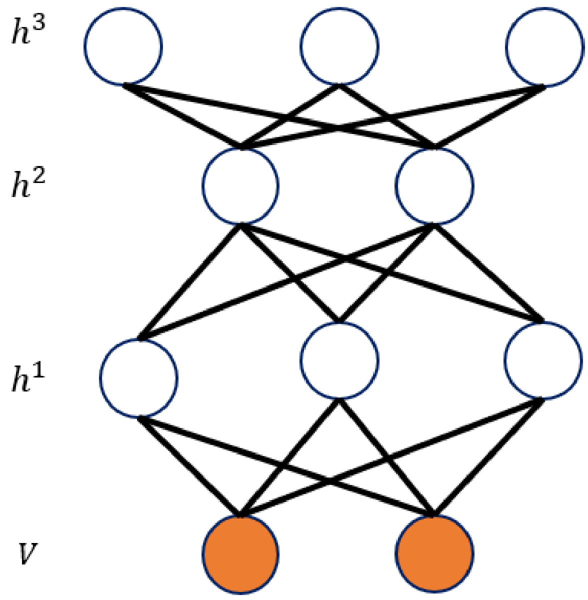

Figure 6: 深度玻尔兹曼机的概率图表示。

得到：

$$\phi_j = \sigma(\sum_{i=1}^{D} v_i W_{ij} + \sum_{m=1\backslash j}^{P} \phi_m J_{jm}) \tag{132}$$

这个方程，就是不动点方程，采用梯度上升，逐步求解。那么我们就不需要采样了，直接求后验概率就可以了。

6 深度玻尔兹曼机

在介绍深度玻尔兹曼机之前，我们一步一步的深入，历史上却是现有的BM（1983），后来提出RBM（1986），2002年提出对比散度（让采样算法更加高效）的算法，在2006提出了DBN，最后在2008年提出了DBM，即深度玻尔兹曼机。

深度玻尔兹曼机（Deep Boltzmann Machine）是多层的，概率图（6）如下： 对于深度玻尔兹曼机，训练方法也是分为两步：

- 预训练（Pre-training）
- 随机梯度上升（SGA）

我们这里，主要介绍预训练如何来做。因为是多个RBM的堆叠。对于单个受限的玻尔兹曼机，似然函数有：

$$\begin{aligned} p(V) &= \sum_{h^1} p(V, h^1) & (133) \\ &= \sum_{h^1} p(h^1; W^1) p(V|h^1; W^1) & (134) \end{aligned}$$

对于第二个RBM，我们需要对第一个RBM的隐藏层进行采样，作为第二个RBM的样本，即：

$$p(h^1|V; W^1) = \prod_{i=1}^{n} p(h_i^1|V; W^1) \tag{135}$$

采用吉布斯采样，一个一个采样，假设采样了N个样本。那么对于第二个RBM，我们同样有：

$$p(h^1;W^2)=\sum_{h^2}p(h^1,h^2;W^2) \tag{136}$$

换句话说，我们在第一个RBM的基础上，又添加了一层。那么我们要问，真正的$p(h^1)$ 与哪一层相关？很明显，既与W^1相关，又与W^2相关。即：

$$p(h^1)=\sum_{h^2,v}p(v,h^1,h^2) \tag{137}$$

一个直觉就是，采用$p(h^1;W^1)$和$p(h^2;W^2)$的算数平均数去近似$p(h^1;W^1,W^2)$。那么我们把这两个概率分别写出来：

$$\begin{aligned} p(h^1;W^1) &= \sum_{v}p(V,h^1;W^1)=\sum_{V}p(V)p(h^1|V;W^1) & (138)\\ p(h^2;W^2) &= \sum_{h^2}p(h^1,h^2;W^2)=\sum_{h^2}p(h^2)p(h^2|h^1;W^2) & (139)\end{aligned}$$

那么这两个式子采用MC采样近似，有：

$$\begin{aligned} p(h^1;W^1) &\simeq \frac{1}{N}\sum_{v\in V}p(h^1|V;W^1) & (140)\\ p(h^2;W^2) &\simeq \frac{1}{N}\sum_{h^2\in H}p(h^2|h^1;W^2) & (141)\end{aligned}$$

如果把这两个分布，简单相加，会存在重复计算的问题（double counting problem）。因为h^2的采样，同样来自于V，那么就意味着样本被重复使用了一次。这样就会出现一个问题，学习的效果非常不好，概率分布会非常尖锐。所以进一步的，只需要把系数减半就行了，换句话说，对于每个RBM，把所有层对应的系数，全都除以2，作为DBM的参数，第一层和最后一层，要考虑边界条件，需要做一些改造，把第一层的向下的权重，除以2，最后一层向上的权重，除以2，以此来保证，对于第一层来说，向上输入的样本是两倍权重的样本，向下输出的样本是单倍权重的样本。关于这些细节，参看文献[**?**]。

References