第十二章　深度信念网络

12.1　玻尔兹曼机

玻尔兹曼机（Boltzmann machine）是一个特殊的概率无向图模型。玻尔兹曼机有以下三点性质。

1. 每个随机变量（节点）是二值的，我们用一个二值的随机向量$\mathbf{X}$来表示所有的变量。在应用时，所有变量一般可以分为两组：可见变量$\mathbf{V}$和隐变量$\mathbf{H}$。

2. 所有变量之间是全连接的。每个变量X_i的取值依赖于所有其它变量$\mathbf{X}_{\setminus i}$。为了简单起见，假设两个变量之间的相互影响（$X_i \to X_j$和$X_j \to X_i$）是对称的，权重相等。这样，玻尔兹曼机可以看做是一个无向图。

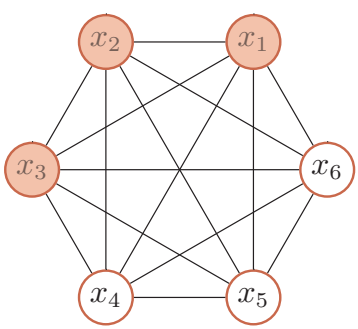

图 12.1: 一个有六个变量的玻尔兹曼机。

3. 整个能量函数定义为

$$E(\mathbf{X}=\mathbf{x}) = -\left(\sum_{i<j} w_{ij}\, x_i\, x_j + \sum_i b_i\, x_i\right), \tag{12.1}$$

其中，w_{ij} 是两个变量 x_i 和 x_j 之间的连接权重，$x_i \in \{0,1\}$ 表示状态，b_i 是节点 i 的偏置。

图12.1给出了一个包含 3 个可见变量和 3 个隐变量的玻尔兹曼机。

在玻尔兹曼机中，每个变量 X_i 可以解释为是否接受一个基本假设 [Ackley et al., 1985]，其取值为 1 或 0 分别表示系统接受或拒绝该假设。变量之间连接的权重为可正可负的实数，代表了两个假设之间的弱约束关系。一个正的权重表示两个假设可以相互支持。也就是说，如果一个假设被接受，另一个也很可能被接受。相反，一个负的权重表示两个假设不能同时被接受。

变量 $\mathbf{X}$ 的联合概率由玻尔兹曼分布得到，即

这也是玻尔兹曼机名称的由来。

$$P(\mathbf{X}=\mathbf{x}) = \frac{1}{Z}\exp\left(\frac{-E(\mathbf{x})}{T}\right), \tag{12.2}$$

为了简单起见，这里我们把玻尔兹曼常数 k 吸收到温度 T 中。

玻尔兹曼机可以用来解决两类问题。一类是搜索问题。当给定变量之间的连接权重，需要找到一组二值向量，使得整个网络的能量最低。另一类是学习问题。当给一组定部分变量的观测值时，计算一组最优的权重。

12.1.1 能量最小化

动态系统是数学上的一个概念，用一个函数来描述一个空间中所有点随时间的变化情况，比如钟摆晃动、水的流动等。

玻尔兹曼机可以看做是一个*随机动力系统*（Stochastic Dynamical System），每个变量的状态都以一定的概率受到其它变量的影响。

要寻找一个变量使得整个网络的能量最小，一个简单（但是低效）的做法是选择一个变量，在其它变量保持不变的情况下，将这个变量设为会导致整个网络能量更低的状态。

因为整个网络的连接是对称的，一个变量 X_i 的两个状态 0（关闭）和 1（打开）之间的能量差异（Energy Gap）为

$$\Delta E_i(\mathbf{x}_{\setminus i}) = E(x_i=0,\mathbf{x}_{\setminus i}) - E(x_i=1,\mathbf{x}_{\setminus i}) \tag{12.3}$$

$$= \sum_j w_{ij}\, x_j + b_i. \tag{12.4}$$

如果能量差异 $\Delta E_i(\mathbf{x}_{\setminus i})$ 大于一定的阈值（比如0），我们就设 $X_i = 1$，否则就设 $X_i = 0$。这种简单、确定性的方法在运行一定时间之后总是可以收敛到一个解。但是这个解是局部最优的，不是全局最优。为了跳出局部最优，就必须允许“偶尔”可以将一个变量设置为使得能量变高的状态。这样我们就需要引入一定的随机性，我们以 $p_i = \sigma\left(\frac{\Delta E_i(\mathbf{x}_{\setminus i})}{T}\right)$ 的概率将变量 X_i 设为1，否则设为0。

p_i 和 X_i 的自身的历史状态无关。

玻尔兹曼机动态运行过程中，随机的选择一个变量 X_i，然后根据上面的概率以一定的随机性设置其状态。在固定温度 T 的情况下，玻尔兹曼机动态运行足够时间之后，系统会达到热平衡。此时，任何全局状态的概率服从玻尔兹曼分布。热平衡时的状态分布也只与系统能量有关，与初始状态无关。

要使得玻尔兹曼机达到热平衡，温度 T 的选择十分关键。当系统温度非常高 $T \to \infty$ 时，$p_i \to 0.5$，即每个变量状态的改变十分容易，每一种网络状态都是一样的，而从很快可以达到热平衡。当系统温度非常低 $T \to 0$ 时，如果 $\Delta E_i(\mathbf{x}_{\setminus i}) > 0$ 则 $p_i \to 1$，如果 $\Delta E_i(\mathbf{x}_{\setminus i}) < 0$ 则 $p_i \to 0$。因此，当 $T \to 0$ 时，随机性方法又变成了确定性的方法。

一个比较好的折中方法是让系统刚开始在一个比较高的温度下运行，然后逐渐降低，直到系统在一个比较低的温度下达到热平衡。这样我们就能够得到一个能量全局最小的分布。这个过程被称为模拟退火（Simulated Annealing）[Kirkpatrick et al., 1983]。

模拟退火是一种寻找全局最优的近似方法，其名字来自冶金学的专有名词“退火”，即将材料加热后再以一定的速度退火冷却，可以减少晶格中的缺陷。固体中的内部粒子会停留在使内能有局部最小值的位置，加热时能量变大，粒子会变得无序并随机移动。退火冷却时速度较慢，使得粒子在每个温度都达到平衡态。最后在常温时，粒子以很大的概率达到内能比原先更低的位置。可以证明，模拟退火算法所得解依概率收敛到全局最优解。

在上面的模拟退火过程中，我们以 $p_i = \sigma\left(\frac{\Delta E_i(\mathbf{x}_{\setminus i})}{T}\right)$ 的概率将变量 X_i 设为1。这个概率是变量 X_i 的条件概率。

定理 12.1 – 玻尔兹曼机中变量的条件概率： 在玻尔兹曼机中，当

给定其它变量 $\mathbf{x}_{\setminus i}$ 时，$p(x_i = 1|\mathbf{x}_{\setminus i})$ 的条件概率为

$$p(x_i = 1|\mathbf{x}_{\setminus i}) = \sigma\left(\frac{\Delta E_i(\mathbf{x}_{\setminus i})}{T}\right). \tag{12.5}$$

其中，σ 为logistic sigmoid 函数。

证明. 根据玻尔兹曼分布的性质，见公式(11.47)，可得

$$\begin{aligned}
\Delta E_i(\mathbf{x}_{\setminus i}) &= -T \ln P(X_i = 0, \mathbf{x}_{\setminus i}) - (-T \ln p(x_i = 1, \mathbf{x}_{\setminus i})) && (12.6)\\
&= T \ln \frac{p(x_i = 1, \mathbf{x}_{\setminus i})}{P(X_i = 0, \mathbf{x}_{\setminus i})} && (12.7)\\
&= T \ln \frac{p(x_i = 1|\mathbf{x}_{\setminus i})}{P(X_i = 0|\mathbf{x}_{\setminus i})} && (12.8)\\
&= T \ln \frac{p(x_i = 1, |\mathbf{x}_{\setminus i})}{1 - p(x_i = 1|\mathbf{x}_{\setminus i})}, && (12.9)
\end{aligned}$$

进而得到，

$$\begin{aligned}
p(x_i = 1|\mathbf{x}_{\setminus i}) &= \frac{1}{1 + \exp\left(-\frac{\Delta E_i(\mathbf{x}_{\setminus i})}{T}\right)} && (12.10)\\
&= \sigma\left(\frac{\Delta E_i(\mathbf{x}_{\setminus i})}{T}\right) && (12.11)\\
&= \sigma\left(\frac{1}{T}\sum_j w_{ij}\, x_j + b_i\right). && (12.12)
\end{aligned}$$

□

12.1.2 参数学习

玻尔兹曼机中的变量分为可见变量 $\mathbf{v} \in \{0, 1\}^m$ 和不可见变量 $\mathbf{h} \in \{0, 1\}^n$。

给定一组可见向量 $\hat{\mathbf{v}}^{(1)}, \hat{\mathbf{v}}^{(2)}, \cdots, \hat{\mathbf{v}}^{(N)}$ 作为训练集，我们要调整玻尔兹曼机的分布使得训练集中所有样本的（对数）似然函数最大。而这个分布由权重参数 W 决定，所以我们需要更新权重 W。训练集的对数似然函数定义为

$$\mathcal{LL}(\mathcal{W}) = \frac{1}{N}\sum_{i=1}^{N} \log p(\hat{\mathbf{v}}^{(n)}) \tag{12.13}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\log\sum_{\mathbf{h}} p(\hat{\mathbf{v}}^{(n)},\mathbf{h}) \tag{12.14}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\log\frac{\sum_{\mathbf{h}}\exp(E(\hat{\mathbf{v}}^{(n)},\mathbf{h}))}{\sum_{\mathbf{v},\mathbf{h}}\exp(E(\mathbf{v},\mathbf{h}))}, \tag{12.15}$$

其中，每个训练向量$p(\hat{\mathbf{v}}^{(n)})$的对数似然对参数$w_{ij}$的导数为

$$\frac{\partial \log p(\hat{\mathbf{v}}^{(n)})}{\partial w_{ij}} = \frac{\partial \log\sum_{\mathbf{h}} p(\hat{\mathbf{v}}^{(n)},\mathbf{h})}{\partial w_i j} \tag{12.16}$$

$$= \frac{\partial \log\sum_{\mathbf{h}}\exp(E(\hat{\mathbf{v}}^{(n)},\mathbf{h})) - \log\sum_{\mathbf{v},\mathbf{h}}\exp(E(\mathbf{v},\mathbf{h}))}{\partial w_{ij}} \tag{12.17}$$

$$= \sum_{\mathbf{h}}\frac{\exp(E(\hat{\mathbf{v}}^{(n)},\mathbf{h}))}{\sum_{\mathbf{h}}\exp(E(\hat{\mathbf{v}}^{(n)},\mathbf{h}))}x_i x_j - \sum_{\mathbf{v},\mathbf{h}}\frac{\exp(E(\mathbf{v},\mathbf{h}))}{\sum_{\mathbf{v},\mathbf{h}}\exp(E(\mathbf{v},\mathbf{h}))}x_i x_j \tag{12.18}$$

$$= \sum_{\mathbf{h}} p(\mathbf{h}|\hat{\mathbf{v}}^{(n)})x_i x_j - \sum_{\mathbf{v},\mathbf{h}} p(\mathbf{v},\mathbf{h})x_i x_j \tag{12.19}$$

$$= \mathbb{E}_{p(\mathbf{h}|\hat{\mathbf{v}}^{(n)})}[x_i x_j] - \mathbb{E}_{p(\mathbf{x})}[x_i x_j], \tag{12.20}$$

其中，$p(\mathbf{h}|\hat{\mathbf{v}}^{(n)})$和$p(\mathbf{v},\mathbf{h})$在当前参数$W$下计算的条件概率和联合概率。

采用梯度上升法，参数w_{ij}的更新公式为：

$$w_{ij} \leftarrow w_{ij} + \alpha\left(\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{p(\mathbf{h}|\hat{\mathbf{v}}^{(n)})}[x_i x_j] - \mathbb{E}_{p(\mathbf{x})}[x_i x_j]\right), \tag{12.21}$$

其中，$\alpha > 0$为学习率。

但是，我们很难精确计算这个梯度。公式(12.20)中，第一项为$x_i x_j$在给定$\mathbf{V} = \hat{\mathbf{v}}^{(n)}$时的期望，第二项为没有任何限制时的期望。因为涉及到计算配分函数，该两个分布很难计算。对于一个D维的随机向量$\mathbf{X}$，其取值空间大小为2^D。当D比较大时，这两项的计算会十分耗时。因此，只能通过一些采样方法（如Gibbs采样）来进行近似求解。

对于第一项，将可见变量固定，通过模拟退火的方法使得网络达到热平衡状态（温度$T = 1$）。对于每一个连接，采样$x_i x_j$的值。在训练集上所有的训练向量上重复此过程，得到$x_i x_j$的近似期望$\langle x_i x_j\rangle_{\texttt{data}}$。

对于第二项，不设任何限制，通过模拟退火的方法使得网络达到热平衡状态（温度$T = 1$）。对于每一个连接，采样$x_i x_j$的值。在训练集上所有的训练向量上重复此过程，得到$x_i x_j$的近似期望$\langle x_i x_j\rangle_{\texttt{model}}$。

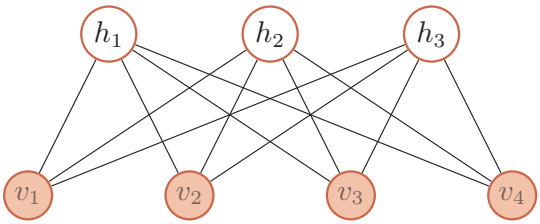

图 12.2: 一个有 7 个变量的受限玻尔兹曼机。

这样，权重 w_{ij} 可以用下面公式近似地更新

$$w_{ij} \leftarrow w_{ij} + \alpha \left(\langle x_i x_j \rangle_{\texttt{data}} - \langle x_i x_j \rangle_{\texttt{model}} \right). \tag{12.22}$$

这个更新方法的一个特点是仅仅使用了局部信息。也就是说，虽然我们优化目标是整个网络的能量最低，但是每个权重的更新只依赖于它连接的相关变量的状态。

玻尔兹曼机可以用在监督学习和无监督学习中。在监督学习中，可见变量又可以分为输入和输出变量，隐变量则隐式地描述了可见变量之间复杂的约束关系。在无监督学习中，隐变量可以看做是可见变量的内部特征表示。玻尔兹曼机也可以看做是一种随机型的神经网络，是 Hopfield 神经网络的扩展，并且可以生成的相应的 Hopfield 神经网络。在没有时间限制时，玻尔兹曼机还可以用来解决复杂的组合优化问题。

全连接的玻尔兹曼机虽然只在理论上十分有趣，但是由于其复杂性，目前为止并没有被广泛使用。虽然基于采样的方法在很大程度提高了学习效率，但是每更新一次权重，就需要网络重新达到热平衡状态，这个过程依然比较低效，需要很长时间。在实际应用中，使用比较广泛的一种带限制的版本，也就是受限玻尔兹曼机。

12.2 受限玻尔兹曼机

受限玻尔兹曼机因其结构最初簧风琴，2000 年后受限玻兹曼机的名称才变得流行。

受限玻尔兹曼机（Restricted Boltzmann Machines，RBM）是一个二分图结构的无向图模型，如图12.2所示。在受限玻尔兹曼机中，变量可以为两组，分别为隐藏层和可见层（或输入层）。同一层中的节点之间没有连接，一个层中的节点与另一层中的所有节点连接。节点变量的取值为 0 或 1。

和两层的全连接神经网络的结构相同。

假设一个受限玻尔兹曼机由 m 个可见层节点和 n 个隐层节点组成，其定义如下：

- 可见层节点：$\mathbf{v} = [v_1, \cdots, v_m]^{\mathrm{T}}$
- 隐藏层节点：$\mathbf{h} = [h_1, \cdots, h_n]^{\mathrm{T}}$
- 权重矩阵：$W = \{w_{i,j}\} \in \mathbb{R}^{n \times m}$，其中每个元素为隐层单元 h_i 和可见层单元 v_j 之间边的权重
- 偏置：每个可见层单元 v_i 有偏置 a_i，对每个隐层单元 h_j 有偏置 b_j

受限玻尔兹曼机的能量函数定义为

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j h_j w_{i,j} v_i \tag{12.23}$$

$$= -\mathbf{a}^{\mathrm{T}}\mathbf{v} - \mathbf{b}^{\mathrm{T}}\mathbf{h} - \mathbf{h}^{\mathrm{T}} W \mathbf{v}, \tag{12.24}$$

随机向量 $(\mathbf{v}, \mathbf{h})$ 的联合概率为

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h})) \tag{12.25}$$

$$= \frac{1}{Z} \exp(\mathbf{a}^{\mathrm{T}}\mathbf{v}) \exp(\mathbf{b}^{\mathrm{T}}\mathbf{h}) \exp(\mathbf{h}^{\mathrm{T}} W \mathbf{v}), \tag{12.26}$$

其中，$Z = \sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))$ 为配分函数。

定理 12.2 – 受限玻尔兹曼机中变量的条件概率： 在受限玻尔兹曼机中，每个可见变量和隐变量的条件概率为

$$p(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_j w_{i,j} h_j\right), \tag{12.27}$$

$$p(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_i w_{i,j} v_i)\right), \tag{12.28}$$

其中，σ 为logistic sigmoid 函数。

证明.（1）我们先证明 $p(v_i = 1|\mathbf{h})$。

可见层变量 $\mathbf{v}$ 的边际概率为

$$P(\mathbf{v}) = \sum_{\mathbf{h}} P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h})) \tag{12.29}$$

$$= \frac{1}{Z} \sum_{\mathbf{h}} \exp\left(\mathbf{a}^{\mathrm{T}}\mathbf{v} + \sum_j b_j h_j + \sum_i \sum_j h_j w_{i,j} v_i\right) \tag{12.30}$$

$$= \frac{\exp(\mathbf{a}^{\mathrm{T}}\mathbf{v})}{Z} \sum_{\mathbf{h}} \exp\left(\sum_j h_j (b_j + \sum_i w_{i,j} v_i)\right) \tag{12.31}$$

$$= \frac{\exp(\mathbf{a}^{\mathrm{T}}\mathbf{v})}{Z} \sum_{\mathbf{h}} \prod_j \exp\left(h_j (b_j + \sum_i w_{i,j} v_i)\right) \tag{12.32}$$

$$= \frac{\exp(\mathbf{a}^{\mathrm{T}}\mathbf{v})}{Z} \sum_{h_1} \sum_{h_2} \cdots \sum_{h_n} \prod_j \exp\left(h_j (b_j + \sum_i w_{i,j} v_i)\right) \tag{12.33}$$

$$= \frac{\exp(\mathbf{a}^{\mathrm{T}}\mathbf{v})}{Z} \prod_j \sum_{h_j} \exp\left(h_j (b_j + \sum_i w_{i,j} v_i)\right) \tag{12.34}$$

将 h_j 为 0 或 1 的取值代入计算。

$$= \frac{\exp(\mathbf{a}^{\mathrm{T}}\mathbf{v})}{Z} \prod_j \left(1 + \exp(b_j + \sum_i w_{i,j} v_i)\right). \tag{12.35}$$

固定 $h_j = 1$ 时，$p(h_j = 1, \mathbf{v})$ 的边际概率为

$$p(h_j = 1, \mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}, h_j = 1} \exp\left(-E(\mathbf{v}, \mathbf{h})\right) \tag{12.36}$$

$$= \frac{\exp(\mathbf{a}^{\mathrm{T}}\mathbf{v})}{Z} \prod_{k, k \neq j} \left(1 + \exp(b_k + \sum_i w_{i,k} v_i)\right) \exp(b_j + \sum_i w_{i,j} v_i). \tag{12.37}$$

由公式12.35和 12.37，可以计算隐藏单元 h_j 的条件概率为：

$$p(h_j = 1 | \mathbf{v}) = \frac{p(h_i = 1, \mathbf{v})}{p(\mathbf{v})} \tag{12.38}$$

$$= \frac{\exp(b_j + \sum_i w_{i,j} v_i)}{1 + \exp(b_j + \sum_i w_{i,j} v_i)} \tag{12.39}$$

$$= \sigma\left(b_j + \sum_i w_{i,j} v_i\right), \tag{12.40}$$

其中，$\sigma(\cdot)$ 为logitic sigmoid 函数。

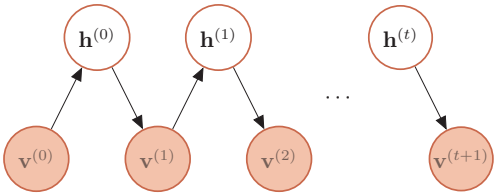

图 12.3: 受限玻尔兹曼机的采样过程

（2）同理，条件概率 $p(v_i = 1|\mathbf{h})$ 为

$$p(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_j w_{i,j} h_j\right). \tag{12.41}$$

□

公式12.40和12.41也可以写为向量形式。

$$p(\mathbf{h} = \underline{1}|\mathbf{v}) = \sigma\left(\mathbf{b} + W^{\mathrm{T}} v_i\right) \tag{12.42}$$

$$p(\mathbf{v} = \underline{1}|\mathbf{h}) = \sigma\left(\mathbf{a} + W h_j\right). \tag{12.43}$$

从上面定理可知，在给定可见变量时，隐变量之间相互条件独立的。同样，在给定隐变量时，可见变量之间也相互条件独立。因此，受限玻尔兹曼机可以并行地对所有的可见变量（或所有的隐变量）同时进行采样，而从可以更快地达到热平衡状态。

采样 受限玻尔兹曼机的采样过程如下：

- 随机初始化一个可见向量 $\hat{\mathbf{v}}_0$, 计算隐层节点的概率，并从中采样一个隐向量 $\mathbf{h}_0$；
- 基于 $\mathbf{h}_0$，计算可见变量概率，并从中采样一个可见向量 $\mathbf{v}_1$；
- 重复 t 次后，我们获得了 $(\mathbf{h}_t, \mathbf{v}_t)$
- 当 $t \to \infty$ 时, $(\mathbf{h}_t, \mathbf{v}_t)$ 的采样服从 $P(V, H)$ 分布。

图12.3也给出了上述过程的示例。

12.2.1 参数学习

和玻尔兹曼机一样，受限玻尔兹曼机通过最大化似然函数来找到最优的参数W。给定一组训练样本$\mathcal{V}=\hat{\mathbf{v}}^{(1)},\hat{\mathbf{v}}^{(2)},\cdots,\hat{\mathbf{v}}^{(N)}$，优化目标为

$$\mathcal{LL}(W)=\frac{1}{N}\sum_{n=1}^{N}\log p(\hat{\mathbf{v}}^{(n)}). \tag{12.44}$$

采用随机梯度上升法，对于样本$\hat{\mathbf{v}}^{(n)}$，目标函数关于参数w_{ij}的梯度为：

$$\frac{\partial\mathcal{LL}(W)}{\partial w_{ij}}=\sum_{h_j}p(h_j|\hat{\mathbf{v}}^{(n)})v_ih_j-\sum_{\mathbf{h},\mathbf{v}}p(\mathbf{v},\mathbf{h})v_ih_j, \tag{12.45}$$

将$h_j=0$或1的值代入上式，得到

$$\frac{\partial\mathcal{LL}(W)}{\partial w_{ij}}=p(h_j=1|\hat{\mathbf{v}}^{(n)})\hat{v}_i-\sum_{\mathbf{v}}p(\mathbf{v})p(h_j=1|\mathbf{v})v_i, \tag{12.46}$$

同理，

$$\frac{\partial\mathcal{LL}(W)}{\partial a_i}=\hat{v}_i-\sum_{\mathbf{v}}p(\mathbf{v})v_i, \tag{12.47}$$

$$\frac{\partial\mathcal{LL}(W)}{\partial b_j}=p(h_j=1|\hat{\mathbf{v}}^{(n)})-\sum_{\mathbf{v}}p(h_j=1|\mathbf{v}), \tag{12.48}$$

上述三个参数梯度的公式中，都需要计算$p(\mathbf{v})$，这就涉及到配分函数的计算，因此也需要通过Gibbs采样的方法来近似计算。根据受限玻尔兹曼机的条件独立性，可以按可见变量和隐变量两组轮流进行分批采样，如图12.3中所示。虽然比一般的玻尔兹曼机速度有很大提高，但一般还是需要通过很多步采样才可以采集到符合真实分布的样本。这就使得受限玻尔兹曼机的训练效率仍然不高。

对比散度学习算法 由于受限玻尔兹曼机的特殊结构，因此可以使用一种比Gibbs采样更有效的学习算法，即**对比散度** (Contrastive Divergence)[Hinton, 2002]。

与Gibbs采样不同，对比散度算法仅需k步Gibbs采样。为了提高效率，对比散度算法用一个训练样本作为可见变量的初始值。然后，轮流进行Gibbs采样，不需要等到收敛，只需要k步就足够了。这就是CD-k算法。通常，$k=1$就可以学得很好。

因为目标是$p(v)\approx p_{train}(v)$

对比散度的流程如算法12.1所示。

算法 12.1: 单步对比散度算法

输入: 训练集: $\hat{\mathbf{v}}^{(n)}, n = 1, \cdots, N$;

学习率：α

初始化：$W \leftarrow 0$, $\mathbf{a} \leftarrow 0$, $\mathbf{b} \leftarrow 0$;

for $t = 1 \cdots T$ **do**

  for $n = 1 \cdots N$ **do**

    选取一个样本 $\hat{\mathbf{v}}^{(n)}$，用公式 (12.40) 计算 $p(\mathbf{h} = \underline{1}|\hat{\mathbf{v}}^{(n)})$，并根据这个分布采集一个隐向量 $\mathbf{h}$;

    计算**正向梯度** $\hat{\mathbf{v}}^{(n)}\mathbf{h}^{\mathrm{T}}$;

    根据 $\mathbf{h}$，用公式 (12.41) 计算 $p(\mathbf{v} = \underline{1}|\mathbf{h})$，并根据这个分布采集重构的可见变量 $\mathbf{v}'$;

    根据 $\mathbf{v}'$，重新计算 $p(\mathbf{h} = \underline{1}|\mathbf{v}')$ 并采样一个 $\mathbf{h}'$;

    计算**反向梯度** $\mathbf{v}'\mathbf{h}'^{\mathrm{T}}$;

    更新参数：

    $W \leftarrow W + \alpha(\hat{\mathbf{v}}^{(n)}\mathbf{h}^{\mathrm{T}} - \mathbf{v}'\mathbf{h}'^{\mathrm{T}})$;

    $\mathbf{a} \leftarrow \mathbf{a} + \alpha(\hat{\mathbf{v}}^{(n)} - \mathbf{v}')$;

    $\mathbf{b} \leftarrow \mathbf{b} + \alpha(\mathbf{h} - \mathbf{h}')$;

  end

end

输出: $W, \mathbf{a}, \mathbf{b}$

受限玻尔兹曼机是一个生成模型，使用隐变量来描述输入数据的分布。同时受限玻尔兹曼机也是一个无监督模型，不需要数据的标签信息。

12.2.2 受限玻尔兹曼机的类型

在具体的不同任务中，需要处理的数据类型不一定都是二值的，也可能是连续值。为了能够处理这些数据，就需要根据输入或输出的数据类型来设计新的能量函数。

一般来说，常见的受限玻尔兹曼机有以下三种：

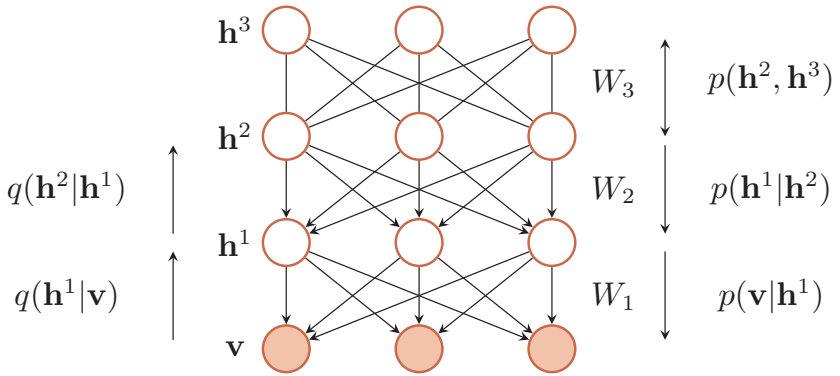

图 12.4: 一个有 4 层结构的深度信念网络。

“贝努力-贝努力”受限玻尔兹曼机 “贝努力-贝努力”受限玻尔兹曼机（Bernoulli-Bernoulli RBM, BB-RBM）就是上面介绍的可见变量和隐变量都为二值类型的受限玻尔兹曼机。

“高斯-贝努力”受限玻尔兹曼机 “高斯-贝努力”受限玻尔兹曼机（Gaussian-Bernoulli RBM, GB-RBM），其能量函数定义为

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i \frac{(v_i - \mu_i)^2}{2\sigma_i^2} - \sum_j b_j h_j - \sum_i \sum_j \frac{v_i}{\sigma_i} w_{i,j} h_j, \tag{12.49}$$

其中，每个可见变量 v_i 服从 (μ_i, σ_i) 的高斯分布。

“贝努力-高斯”受限玻尔兹曼机 “贝努力-高斯”受限玻尔兹曼机（Bernoulli-Gaussian RBM, BG-RBM），其能量函数定义为

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j \frac{(h_j - \mu_j)^2}{2\sigma_j^2} - \sum_i \sum_j v_i w_{i,j} \frac{h_j}{\sigma_j}, \tag{12.50}$$

其中，每个可见变量 v_i 服从 (μ_i, σ_i) 的高斯分布。

12.3 深度信念网络

深度信念网络（deep belief network，DBN）是深度的有向的概率图模型，其图结构由多层的节点构成。每层节点的内部没有连接，相邻两层的节点之间

和全连接的神经网络结构相同。

为全连接。网络的最底层为可见变量，其它层节点都为隐变量。最顶部的两层间的连接是无向的，其他层之间有连接上下的有向连接。图12.4给出了一个深度信念网络的示例。对一个有 L 层隐变量的深度信念网络，最底层（第0层）为可见变量 $\mathbf{v} = \mathbf{h}^{(0)}$，其余每层变量为 $\mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(L)}$。除了最顶上两层外，每一层变量 $\mathbf{h}^{(l)}$ 依赖于其上面一层 $\mathbf{h}^{(l+1)}$，即

$$p(\mathbf{h}^{(l)}|\mathbf{h}^{(l+1)}, \cdots, \mathbf{h}^{(L)}) = p(\mathbf{h}^{(l)}|\mathbf{h}^{(l+1)}), \tag{12.51}$$

其中，$l = \{0, \cdots, L-2\}$。

顶部的两层是一个无向图，可以看做是一个受限玻尔兹曼机，用来产生 $p(\mathbf{h}^{(L-1)})$ 的先验分布。

深度信念网络中所有变量的联合概率可以分解为

$$p(\mathbf{v}, \mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(L)}) = p(\mathbf{v}|\mathbf{h}^{(1)}) \left(\prod_{l=1}^{L-2} p(\mathbf{h}^{(l)}|\mathbf{h}^{(l+1)}) \right) p(\mathbf{h}^{(L-1)}, \mathbf{h}^{(L)}) \tag{12.52}$$

$$= \left(\prod_{l=0}^{L-2} p(\mathbf{h}^{(l)}|\mathbf{h}^{(l+1)}) \right) p(\mathbf{h}^{(L-1)}, \mathbf{h}^{(L)}), \tag{12.53}$$

其中，$p(\mathbf{h}^{(l)}|\mathbf{h}^{(l+1)})$ 为sigmoid 型条件概率分布为

$$p(\mathbf{h}^{(l)}|\mathbf{h}^{(l+1)}) = \sigma\left(\mathbf{b}^{(l+1)} + W^{(l+1)}\mathbf{h}^{(l+1)}\right), \tag{12.54}$$

其中，$\sigma(\cdot)$ 为按位计算的logistic sigmoid 函数，$\mathbf{b}^{(l+1)}$ 为偏置参数，$W^{(l+1)}$ 为权重参数。

生成模型 深度信念网络是一个生成模型，可以用来生成符合特定分布的样本。隐变量用来描述在可见变量之间的高阶相关性。假如训练数据服从分布 $p(\mathbf{v})$，通过训练得到一个深度信念网络。在生成样本时，首先在最顶两层进行足够多次的吉布斯采样，生成 $\mathbf{h}^{(L-1)}$，然后依次计算下一层隐变量的分布。因为在给定上一层变量取值时，下一层的变量是条件独立的，因为可以独立采样。这样，我们可以从第 $L-1$ 层开始，自顶向下进行逐层采样，最终得到可见层的样本。

12.3.1 深度信念网络的训练

深度信念网络最直接的训练方式可以通过最大似然方法使得可见层变量 $\mathbf{v}$ 的分布 $p(\mathbf{v})$ 在训练集合上的似然达到最大。但在深度信念网络中，隐变量 $\mathbf{h}$ 之

间的关系十分复杂，由于“贡献度分配问题”，很难直接学习。即使对于简单的单层信念网络 $p(v=1|\mathbf{h})=\sigma\left(b+\mathbf{w}^{\mathrm{T}}\mathbf{h}\right)$，在已知可见变量时，其隐变量的联合后验概率 $p(\mathbf{h}|v)$ 不再相互独立，因此很难精确估计所有隐变量的后验概率。早期深度信念网络的后验概率一般通过蒙特卡罗方法或变分方法来近似估计，但是效率比较低，而导致其参数学习比较困难。

为了有效地训练深度信念网络，我们将每一层的 sigmoid 信念网络转换为受限玻尔兹曼机。这样做的好处是隐变量的后验概率是相互独立的，从而可以很容易地进行采样。这样，深度信念网络可以看作是由多个受限玻尔兹曼机从下到上进行堆叠，每一层受限玻尔兹曼机的隐层作为上一层受限玻尔兹曼机的可见层。进一步地，深度信念网络可以采用逐层训练的方式来快速训练，即从最底层开始，每次只训练一层，直到最后一层 [Hinton et al., 2006]。

“逐层训练”是能够有效训练深度模型的最早的方法。

深度信念网络的训练过程可以分为**预训练**和**精调**两个阶段。先通过逐层预训练将模型的参数初始化为较优的值，再通过传统学习方法对参数进行精调。

预训练 在预训练阶段，我们采用逐层训练的方式，将深度信念网络的训练简化为对多个受限玻尔兹曼机的训练。

算法12.2给出一种深度信念网络的逐层预训练方法。大量的实践表明，预训练可以产生非常好的参数初始值，从而极大地降低了模型的学习难度。

精调 经过预训练之后，再结合具体的任务（监督学习或重构），通过传统的全局学习算法对网络进行精调（Fine-Tuning），使模型收敛到更好的局部最优点。

除了顶层的受限玻尔兹曼机，其它层之间的权重被分成向上的**认知权重**（“Recognition”Weights）R 和向下的**生成权重**（“Generative”Weights）W。认知权重用来进行后验概率计算，而生成权重用来进行定义模型。认知权重的初始值 $R^{(l)}=W^{(l)^{\mathrm{T}}}$。

深度信念网络一般采样 Contrastive Wake-Sleep 算法进行精调，其算法过程是：

- 随机初始化权重；
- Wake 阶段：认知过程，通过外界输入（可见变量）和向上认知权重，计算每一层隐变量的后验概率并采样。然后，修改下行的生成权重使得下一

算法 12.2: 深度信念网络的逐层训练方法

输入: 训练集: $\hat{\mathbf{v}}^{(n)}, n = 1, \cdots, N$;
学习率：α;
深度信念网络层数:L;
第 l 层权重：$W^{(l)}$;
第 l 层偏置 $\mathbf{a}^{(l)}$;
第 l 层偏置 $\mathbf{b}^{(l)}$;

for $l = 1 \cdots L$ **do**
  初始化：$W^{(l)} \leftarrow 0,\ \mathbf{a}^{(l)} \leftarrow 0,\ \mathbf{b}^{(l)} \leftarrow 0$;
  从训练集中采样 $\mathbf{h}^{(0)} = \hat{\mathbf{v}}$;
  for $i = 1 \cdots l-1$ **do**
    根据分布 $q(\mathbf{h}^{(i)}|\mathbf{h}^{(i-1)})$ 采样 $\mathbf{h}^{(i)}$;
  end
  将 $\mathbf{h}^{(l-1)}$ 作为训练样本，充分训练第 l 个受限玻尔兹曼机 $W^{(l)}, \mathbf{a}^{(l)}, \mathbf{b}^{(l)}$;
end

输出: $W_1, \cdots, W_L$

层的变量的后验概率最大。也就是“如果现实跟我想象的不一样，改变我的权重使得我想象的东西就是这样的”；

- Sleep 阶段：生成过程，通过顶层的采样和向下的生成权重，逐层计算每一层的后验概率并采样。然后，修改向上的认知权重使得上一层变量的后验概率最大。也就是“如果梦中的景象不是我脑中的相应概念，改变我的认知权重使得这种景象在我看来就是这个概念”；
- 交替进行 Wake 和 Sleep 过程，直到收敛。

12.3.2 作为深度神经网络的预训练

只需要向上的认知权重。

深度信念网络的一个应用是作为深度神经网络的预训练部分，提供神经网络的初始权重。在深度信念网络的最顶层再增加一层输出层，然后再使用反向传播算法对这些权重进行调优。

特别是在训练数据比较少时，预训练的作用非常大。因为不恰当的初始化

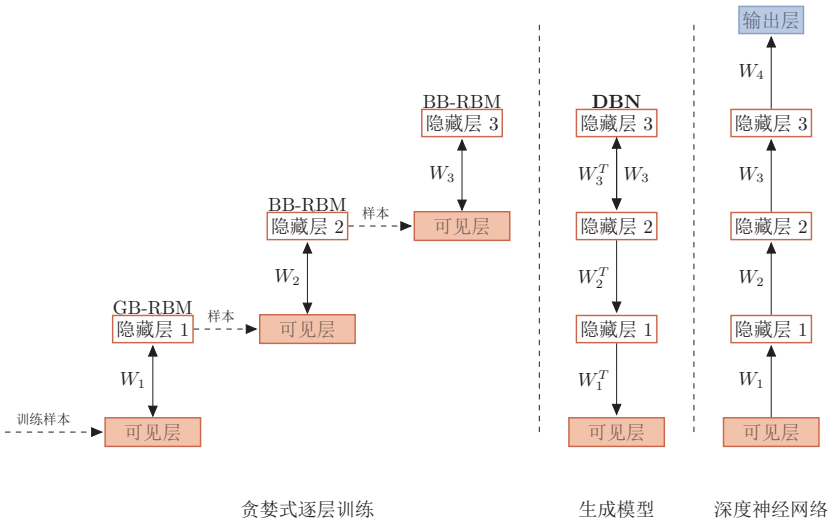

图 12.5: 一个有 4 层结构的深度信念网络。

权重会显著影响最终模型的性能，而预训练获得的权重在权值空间中比随机权重更接近最优的权重，避免了反向传播算法因随机初始化权值参数而容易陷入局部最优和训练时间长的缺点。这不仅提升了模型的性能，也加快了调优阶段的收敛速度 [Larochelle et al., 2007]。

除了深度信念网络之外，自编码器 [Bengio et al., 2007] 以及它的变体，比如稀疏自编码器 [Ranzato et al., 2006] 和去噪自编码器 [Vincent et al., 2008]，也可以用来作为深度神经网络的初始化。

12.3.3 卷积深度置信网络

12.4 图模型与神经网络的关系

图模型和神经网络有着类似的网络结构，但两者也有很大的不同。图模型的节点是随机变量，其图结构的主要功能是用来描述变量之间的依赖关系，一般是稀疏连接。使用图模型的好处是可以有效进行统计推断。而神经网络中的节点是神经元，是一个计算节点。如果将神经网络中每个神经元看做是一个二值随机变量，那神经网络就变成一个 sigmoid 信念网络。

图模型中的每个变量一般有着明确的解释，变量之间依赖关系一般是人工来定义。而神经网络中的神经元则没有直观的解释。

判别模型也可以用图模型来表示。

图模型一般是生成模型，可以用生成样本，也可以通过贝叶斯公式用来做分类。而神经网络是判别模型，直接用来分类。

图模型的参数学习的目标函数为似然函数或条件似然函数，若包含隐变量则通常通过EM算法来求解。而神经网络参数学习的目标为交叉熵或平方误差等损失函数。

12.5 总结和深入阅读

习题 12-1 证明Jensen不等式。

参考文献

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. Optimization by simulated annealing. *science*, 220(4598): 671–680, 1983.

Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480. ACM, 2007.

Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun. Efficient learning of sparse representations with an energy-based model. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, pages 1137–1144. MIT Press, 2006.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the International Conference on Machine Learning*, pages 1096–1103. ACM, 2008.