

《神经网络与深度学习》



深度信念网络 Deep Belief Network

<https://nndl.github.io/>

主要内容

- ▶ 玻尔兹曼机
- ▶ 受限玻尔兹曼机
- ▶ 深度信念网络



玻尔兹曼机 Boltzmann Machine

玻尔兹曼机 (Boltzmann Machine)

▶ 玻尔兹曼机是一个特殊的概率无向图模型。

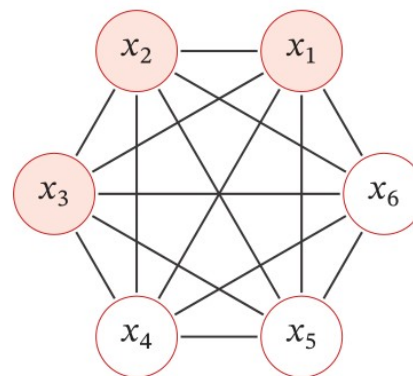
- ▶ 每个随机变量是二值的
- ▶ 所有变量之间是全连接的
- ▶ 整个能量函数定义为

$$\begin{aligned} E(\mathbf{x}) &\triangleq E(\mathbf{X} = \mathbf{x}) \\ &= - \left(\sum_{i < j} w_{ij} x_i x_j + \sum_i b_i x_i \right) \end{aligned}$$

▶ $p(\mathbf{x})$ 为玻尔兹曼分布

$$p(\mathbf{x}) = \frac{1}{Z} \exp \left(\frac{-E(\mathbf{x})}{T} \right)$$

▶ T 表示温度



包含六个变量的玻尔兹曼机

两个基本问题:

1. 推断 $p(\mathbf{h}|\mathbf{v})$
2. 参数学习 $\mathbf{W}$

玻尔兹曼机的推断

▶ 如何推断?

$$p(\mathbf{x}) = \frac{1}{Z} \exp\left(\frac{-E(\mathbf{x})}{T}\right)$$

▶ 配分函数 Z 通常难以计算

▶ 近似采样--Gibbs 采样

▶ 全条件概率

定理 12.1 – 玻尔兹曼机中变量的全条件概率: 对于玻尔兹曼机中的一个变量 X_i , 当给定其他变量 $\mathbf{x}_{\setminus i}$ 时, 全条件概率 $p(x_i|\mathbf{x}_{\setminus i})$ 为

$$p(x_i = 1|\mathbf{x}_{\setminus i}) = \sigma\left(\frac{\sum_j w_{ij} x_j + b_i}{T}\right), \quad (12.5)$$

$$p(x_i = 0|\mathbf{x}_{\setminus i}) = 1 - p(x_i = 1|\mathbf{x}_{\setminus i}), \quad (12.6)$$

其中 $\sigma(\cdot)$ 为 Logistic 函数.

吉布斯采样



吉布斯采样 玻尔兹曼机的吉布斯采样过程为:随机选择一个变量 X_i , 然后根据其全条件概率 $p(x_i|\mathbf{x}_{\setminus i})$ 来设置其状态, 即以 $p(x_i = 1|\mathbf{x}_{\setminus i})$ 的概率将变量 X_i 设为 1, 否则为 0. 在固定温度 T 的情况下, 在运行足够时间之后, 玻尔兹曼机会达到**热平衡**. 此时, 任何全局状态的概率服从玻尔兹曼分布 $p(\mathbf{x})$, 只与系统的能量有关, 与初始状态无关.

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

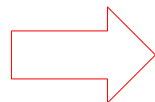
►要使得玻尔兹曼机达到热平衡, 其收敛速度和温度 T 相关.

模拟退火

- ▶ 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。
- ▶ 当系统温度非常高 $T \rightarrow \infty$ 时， $p_i \rightarrow 0.5$ ，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 $T \rightarrow 0$ 时，如果 $\Delta E_i(x \setminus i) > 0$ 则 $p_i \rightarrow 1$ ，如果 $\Delta E_i(x \setminus i) < 0$ 则 $p_i \rightarrow 0$ 。

$$p(x_i = 1 | \mathbf{x}_{\setminus i}) = \frac{1}{1 + \exp\left(-\frac{\Delta E_i(\mathbf{x}_{\setminus i})}{T}\right)} \\ = \sigma\left(\frac{\sum_j w_{ij} x_j + b_i}{T}\right)$$

随机性方法

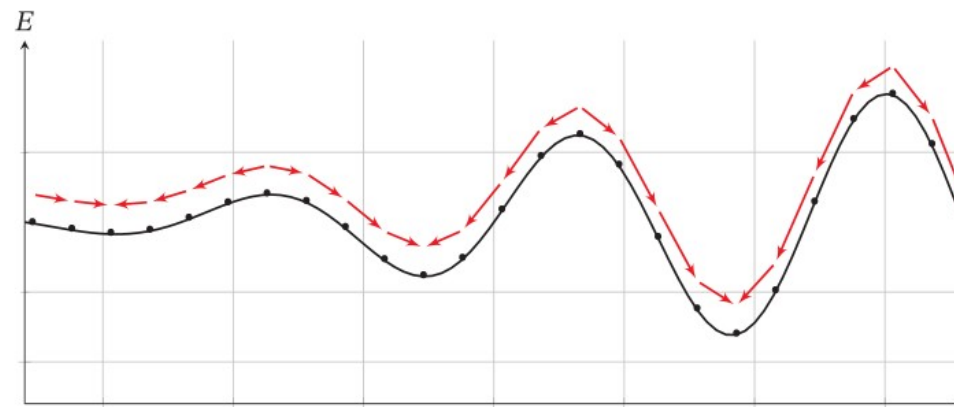


$$x_i = \begin{cases} 1 & \text{if } \sum_j w_{ij} x_j + b_i \geq 0, \\ 0 & \text{otherwise,} \end{cases}$$

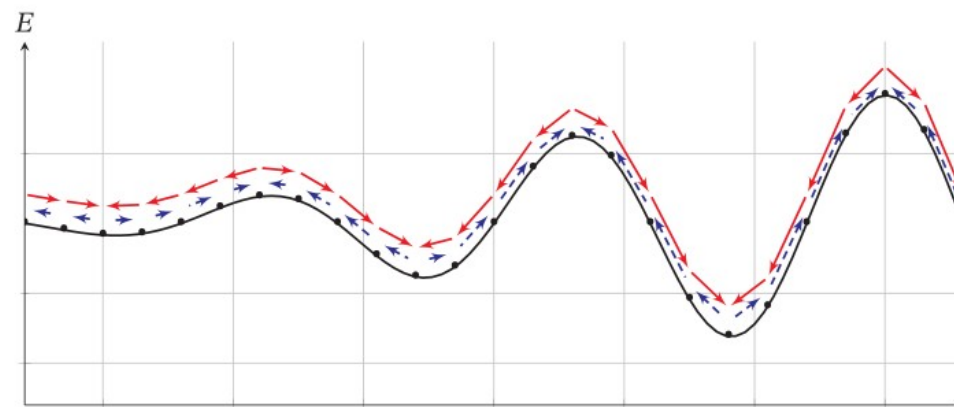
确定性方法

Hopfield 网络

Hopfield 网络和玻尔兹曼的能量变化对比



(a) Hopfield 网络



(b) 玻尔兹曼机

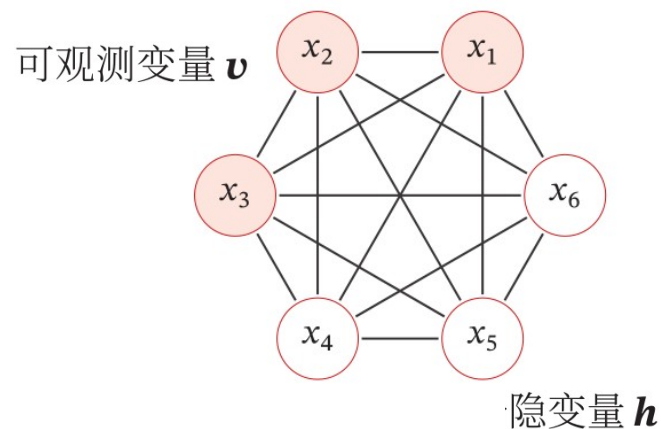
玻尔兹曼机的参数学习

► 最大似然估计

$$\mathcal{L}(\mathcal{D}; \mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_{n=1}^N \log p(\hat{\mathbf{v}}^{(n)}; \mathbf{W}, \mathbf{b})$$

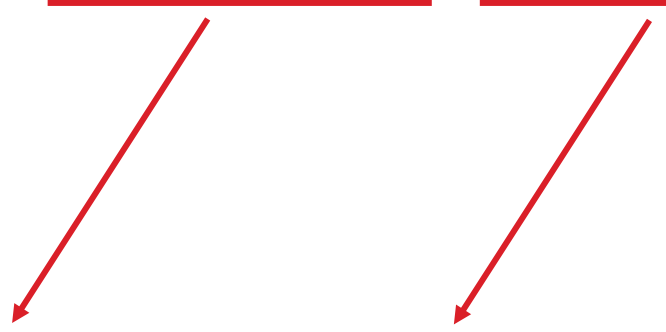
► 采用梯度上升法

$$\frac{\partial \mathcal{L}(\mathcal{D}; \mathbf{W}, \mathbf{b})}{\partial w_{ij}} = \mathbb{E}_{\hat{p}(\mathbf{v})} \mathbb{E}_{p(\mathbf{h}|\mathbf{v})} [x_i x_j] - \mathbb{E}_{p(\mathbf{v}, \mathbf{h})} [x_i x_j],$$
$$\frac{\partial \mathcal{L}(\mathcal{D}; \mathbf{W}, \mathbf{b})}{\partial b_i} = \mathbb{E}_{\hat{p}(\mathbf{v})} \mathbb{E}_{p(\mathbf{h}|\mathbf{v})} [x_i] - \mathbb{E}_{p(\mathbf{v}, \mathbf{h})} [x_i],$$



玻尔兹曼机的参数学习

- ▶ 基于 Gibbs 采样来进行近似求解

$$\frac{\mathbb{E}_{\hat{p}(\mathbf{v})} \mathbb{E}_{p(\mathbf{h}|\mathbf{v})}[x_i x_j]}{\quad} - \frac{\mathbb{E}_{p(\mathbf{v}, \mathbf{h})}[x_i x_j]}{\quad}$$

$$w_{ij} \leftarrow w_{ij} + \alpha \left(\underline{\langle x_i x_j \rangle_{\text{data}}} - \underline{\langle x_i x_j \rangle_{\text{model}}} \right)$$



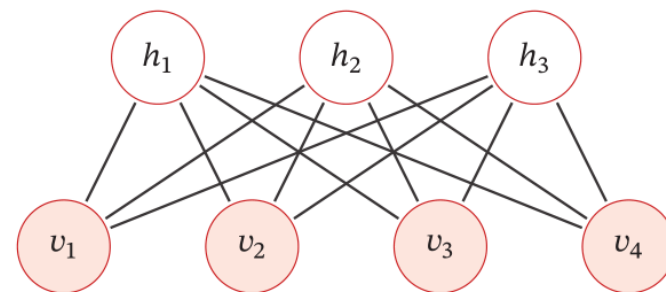
受限玻尔兹曼机

Restricted Boltzmann Machines

受限玻尔兹曼机（Restricted Boltzmann Machines，RBM）

- ▶ 受限玻尔兹曼机是一个二分图结构的无向图模型。
 - ▶ 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）。
 - ▶ 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。

$$\begin{aligned} E(\mathbf{v}, \mathbf{h}) &= -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{ij} h_j \\ &= -\mathbf{a}^\top \mathbf{v} - \mathbf{b}^\top \mathbf{h} - \mathbf{v}^\top \mathbf{W} \mathbf{h}. \end{aligned}$$



$$\begin{aligned} p(\mathbf{v}, \mathbf{h}) &= \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h})) \\ &= \frac{1}{Z} \exp(\mathbf{a}^\top \mathbf{v}) \exp(\mathbf{b}^\top \mathbf{h}) \exp(\mathbf{v}^\top \mathbf{W} \mathbf{h}), \end{aligned}$$

全条件概率

- ▶ 从无向图的性质可知，
 - ▶ 在给定可观测变量时，隐变量之间互相条件独立。
 - ▶ 在给定隐变量时，可观测变量之间也互相条件独立。
- ▶ 全条件概率

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，每个可观测变量和隐变量的条件概率为

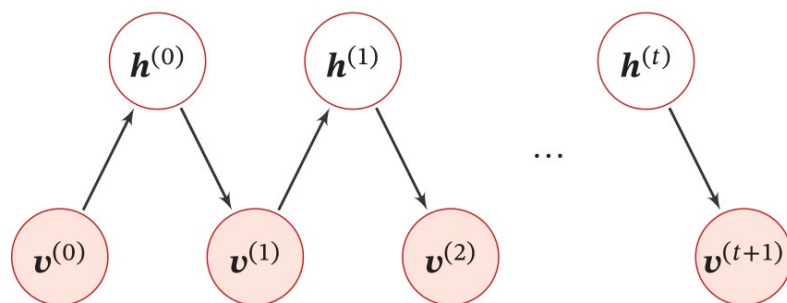
$$p(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_j w_{ij}h_j\right), \quad (12.33)$$

$$p(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_i w_{ij}v_i\right), \quad (12.34)$$

其中 σ 为 Logistic 函数.

吉布斯采样

- 受限玻尔兹曼机可以并行地对所有的可观测变量（或所有的隐变量）同时进行采样，从而可以更快地达到热平衡状态。



$$p(\mathbf{h} = 1|\mathbf{v}) = \sigma(\mathbf{W}^\top \mathbf{v} + \mathbf{b})$$

$$p(\mathbf{v} = 1|\mathbf{h}) = \sigma(\mathbf{W}\mathbf{h} + \mathbf{a}).$$

参数学习

- ▶ 采用梯度上升法时，参数 W, a, b 可以用下面公式近似地更新

$$w_{ij} \leftarrow w_{ij} + \alpha (\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}),$$

$$a_i \leftarrow a_i + \alpha (\langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}}),$$

$$b_j \leftarrow b_j + \alpha (\langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{model}}),$$

- ▶ 根据受限玻尔兹曼机的条件独立性，可以对可观测变量和隐变量进行分组轮流采样。

对比散度算法 Contrastive Divergence

算法 12.1 单步对比散度算法

输入: 训练集 $\{\mathbf{v}^{(n)}\}_{n=1}^N$, 学习率 α

```
1 初始化:  $\mathbf{W} \leftarrow 0, \mathbf{a} \leftarrow 0, \mathbf{b} \leftarrow 0$ ;  
2 for  $t = 1 \cdots T$  do  
3   for  $n = 1 \cdots N$  do  
4     选取一个样本  $\mathbf{v}^{(n)}$ , 用公式 (12.46) 计算  $p(\mathbf{h} = \mathbf{1}|\mathbf{v}^{(n)})$ , 并根据这个  
       分布采集一个隐向量  $\mathbf{h}$ ;  
5     计算正向梯度  $\mathbf{v}^{(n)}\mathbf{h}^\top$ ;  
6     根据  $\mathbf{h}$ , 用公式 (12.47) 计算  $p(\mathbf{v} = \mathbf{1}|\mathbf{h})$ , 并根据这个分布采集重构的  
       可见变量  $\mathbf{v}'$ ;  
7     根据  $\mathbf{v}'$ , 重新计算  $p(\mathbf{h} = \mathbf{1}|\mathbf{v}')$  并采样一个  $\mathbf{h}'$ ;  
8     计算反向梯度  $\mathbf{v}'\mathbf{h}'^\top$ ;  
       // 更新参数  
9      $\mathbf{W} \leftarrow \mathbf{W} + \alpha(\mathbf{v}^{(n)}\mathbf{h}^\top - \mathbf{v}'\mathbf{h}'^\top)$ ;  
10     $\mathbf{a} \leftarrow \mathbf{a} + \alpha(\mathbf{v}^{(n)} - \mathbf{v}')$ ;  
11     $\mathbf{b} \leftarrow \mathbf{b} + \alpha(\mathbf{h} - \mathbf{h}')$ ;  
12  end  
13 end  
   输出:  $\mathbf{W}, \mathbf{a}, \mathbf{b}$ 
```

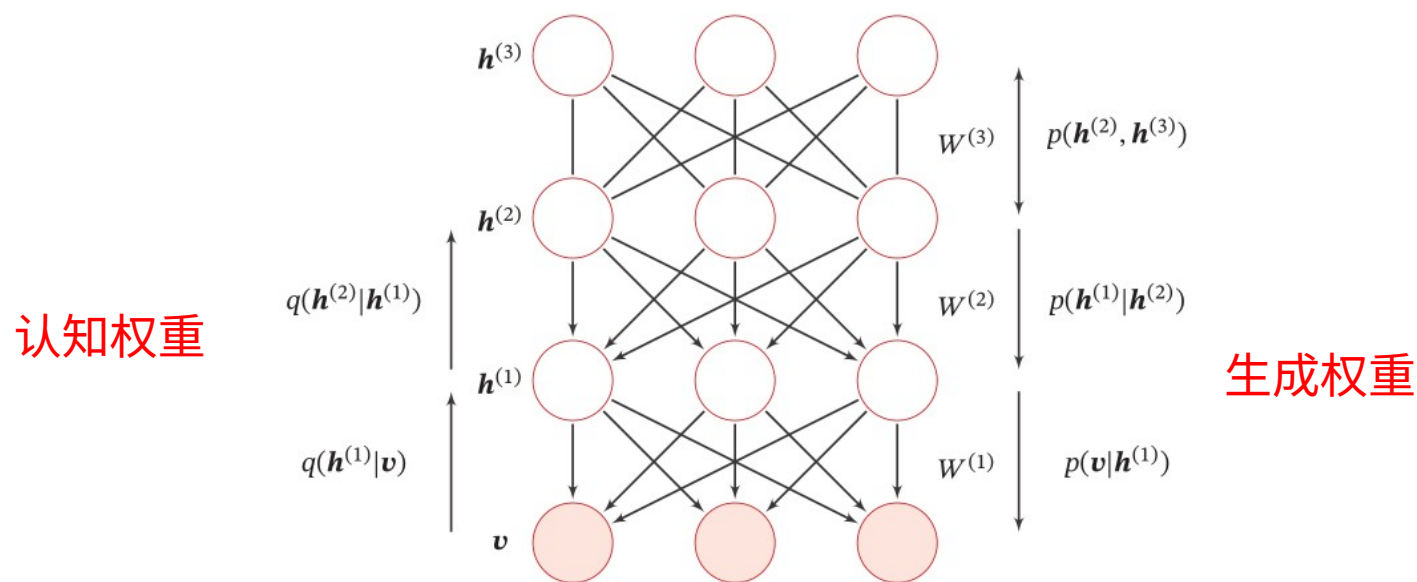
吉布斯采样, 不需要等到收敛, 只需要 K 步



深度信念网络 Deep Belief Network

深度信念网络 (Deep Belief Network)

- ▶ 深度信念网络是深度的概率有向图模型
 - ▶ 和全连接的神经网络结构相同。
 - ▶ 顶部的两层为一个无向图，可以看做是一个受限玻尔兹曼机。



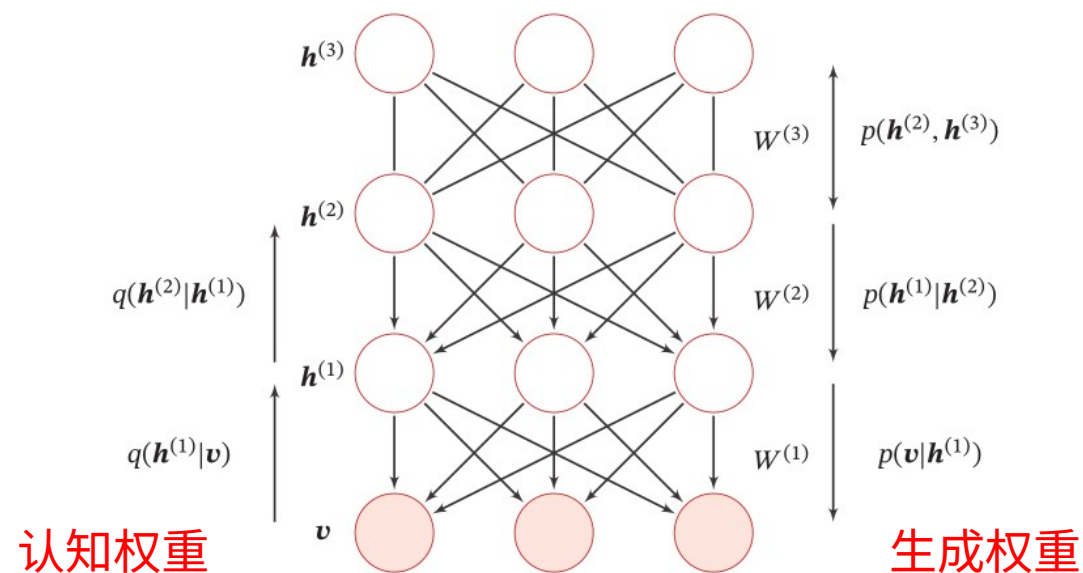
局部条件概率

- 深度信念网络中所有变量的联合概率可以分解为

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

- 局部条件概率

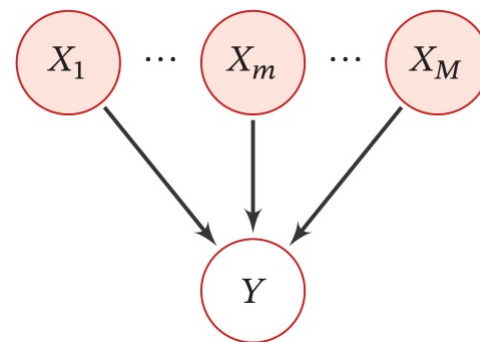
$$p(\mathbf{h}^{(l)}|\mathbf{h}^{(l+1)}) = \sigma(\mathbf{a}^{(l)} + \mathbf{W}^{(l+1)}\mathbf{h}^{(l+1)})$$



参数学习

- ▶ 深度信念网络的参数学习是一种含隐变量的参数学习
 - ▶ 需要EM或变分方法
 - ▶ 隐变量 $\mathbf{h}$ 之间的关系十分复杂，很难直接学习。
- ▶ 在已知可观测变量时，其隐变量的联合后验概率 $p(\mathbf{h}|\mathbf{v})$ 不再相互独立，因此很难精确估计所有隐变量的后验概率。

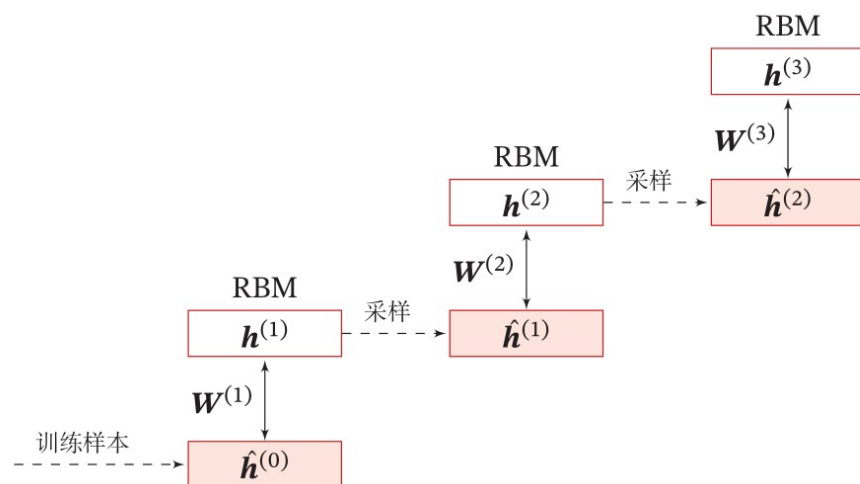
- ▶ 只有一层的简单 Sigmoid 信念网络



$$p(v = 1|\mathbf{h}) = \sigma(b + \mathbf{w}^\top \mathbf{h})$$

训练深度信念网络 - 逐层训练

- ▶ 逐层训练是能够有效训练深度模型的最早的方法。



算法 12.2 深度信念网络的逐层预训练方法

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

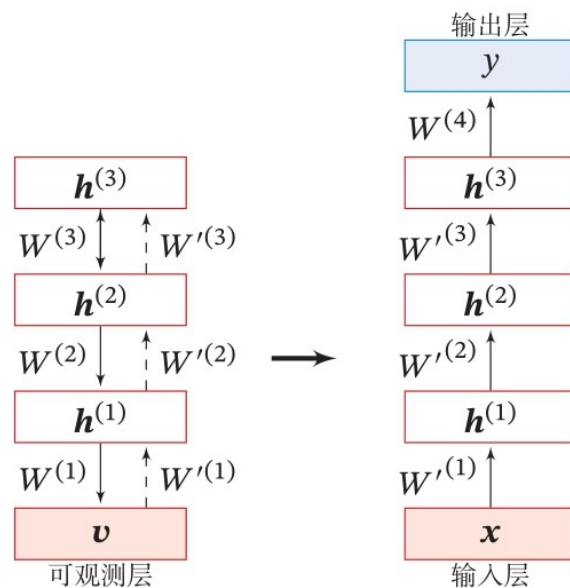
```
1 for  $l = 1 \dots L$  do
2   初始化:  $\mathbf{W}^{(l)} \leftarrow 0, \mathbf{a}^{(l)} \leftarrow 0, \mathbf{b}^{(l)} \leftarrow 0$ ;
3   从训练集中采样  $\hat{\mathbf{h}}^{(0)}$ ;
4   for  $i = 1 \dots l - 1$  do
5     根据分布  $p(\mathbf{h}^{(i)} | \hat{\mathbf{h}}^{(i-1)})$  采样  $\hat{\mathbf{h}}^{(i)}$ ;
6   end
7   将  $\hat{\mathbf{h}}^{(l-1)}$  作为训练样本, 充分训练第  $l$  层受限玻尔兹曼机, 得到参数  $\mathbf{W}^{(l)}, \mathbf{a}^{(l)}, \mathbf{b}^{(l)}$ ;
8 end
```

输出: $\{\mathbf{W}^{(l)}, \mathbf{a}^{(l)}, \mathbf{b}^{(l)}\}, 1 \leq l \leq L$

训练深度信念网络 - 精调 (Fine-Tuning)

► 作为判别模型的精调

► 深度信念网络作为神经网络的预训练模型



训练深度信念网络 - 精调 (Fine-Tuning)

► 作为生成模型的精调

► Contrastive Wake-Sleep 算法

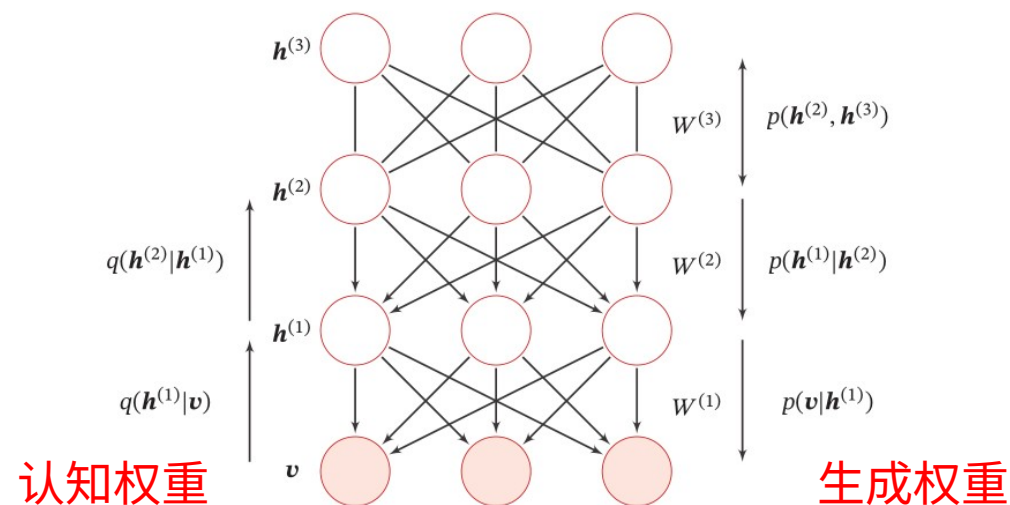
- Wake 阶段：认知过程，通过外界输入（可见变量）和向上认知权重，计算每一层隐变量的后验概率并采样。然后，修改下行的生成权重使得下一层的变量的后验概率最大

□ “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。

□ “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。





谢 谢

<https://nndl.github.io/>