RBM 中的数学

周志阳

2014 年 2 月 18 日

目 录

受限波尔兹曼机 (Restricted Boltzmann Machine, RBM) 是一种可用**随机神经网络** (stochastic neural network) 来解释的**概率图模型** (probabilistic graphical model). 它由 Smolensky 于 1986 年在**波尔兹曼机** (Boltzmann Machine, BM) 的基础上提出, 所谓“随机”, 是指这种网络中的神经元是随机神经元, 其输出只有两种状态 (未激活、激活), 一般用二进制的 0 和 1 来表示, 而状态的具体取值则根据概率统计法则来决定 ([3]).

随着计算机计算能力的迅速提高和快速算法的不断发展, RBM 在各种相关机器学习算法中已经变得实际可行. 尤其是, 在 Hinton 于 2006 年提出了以 RBM 为基本构成模块的 DBN (Deep Belief Network) 模型之后, 机器学习界更是掀起了一股研究 RBM 理论及应用的热潮.

RBM 模型涉及的知识点较多, 本文主要针对 RBM 的网络结构、数学模型以及快速训练算法进行介绍, 且重点放在一些数学公式的推导及具体算法的描述上.

§1 预备知识

本节的预备知识相对独立, 仅供参考, 读者可以先跳过本节, 阅读过程中碰到相关问题再回过头来查阅.

§1.1 sigmoid 函数

sigmoid 函数是神经网络中常用的激活函数之一, 其定义为

$$sigmoid(x) = \frac{1}{1+e^{-x}}, \tag{1.1}$$

该函数的定义域为 $(-\infty, +\infty)$, 值域为 $(0, 1)$. 图 1 给出了 sigmoid 函数的图像.

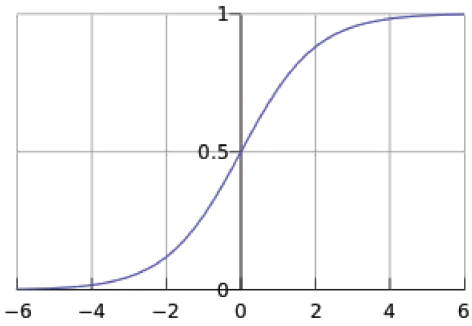

图 1 sigmoid 函数的图像

§1.2 Bayes 定理

贝叶斯定理是英国数学家**贝叶斯** (Thomas Bayes) 提出来的, 用来描述两个条件概率之间的关系. 若记 $P(A)$, $P(B)$ 分别表示事件 A 和事件 B 发生的概率, $P(A|B)$ 表示事件 B 发生的情况下事件 A 发生的概率, $P(A, B)$ 表示事件 A, B 同时发生的概率, 则有

$$P(A|B) = \frac{P(A,B)}{P(B)}, \tag{1.2}$$

$$P(B|A) = \frac{P(A,B)}{P(A)}, \tag{1.3}$$

利用 (1.2) 和 (1.3), 进一步可得

$$P(A|B) = P(A)\frac{P(B|A)}{P(B)}, \tag{1.4}$$

这就是**贝叶斯公式**.

在 (1.4) 中, 我们把 $P(A)$ 称为**"先验概率"** (Prior probability), 即在事件 B 发生之前, 我们对事件 A 发生概率的一个判断. $P(A|B)$ 称为**"后验概率"** (Posterior probability), 即在事件 B 发生之后, 我们对事件 A 发生概率的重新评估. $\frac{P(B|A)}{P(B)}$ 称为**"可能性函数"** (Likelyhood), 这是一个调整因子, 使得预估概率更接近真实概率 ([6]).

§1.3 二分图

二分图 (bipartite graph) 又称二部图、双分图或偶图, 是图论中的一种特殊模型. 设 $G = (V, E)$ 是一个无向图, 如果顶点 V 可分割为两个互不相交的子集 V_1 和 V_2, 并且图中的每条边 (i, j) 所关联的两个顶点 i 和 j 分别属于这两个不同的顶点集, 即 $i \in V_1, j \in V_2$, 或者 $i \in V_2, j \in V_1$, 则称图 G 为一个二分图.

§1.4 MCMC 方法 ([2])

最早的**蒙特卡罗方法**是由物理学家发明的, 旨在于通过随机化的方法计算积分. 假设给定函数 $h(x)$, 我们想计算如下的积分

$$\int_a^b h(x)dx. \tag{1.5}$$

如果我们无法通过数学推导直接求出解析解, 那么, 为了避免对区间 (a, b) 上所有的 x 值进行枚举 (多数情况下这也是不可能的), 我们可以将 $h(x)$ 分解为某个函数 $f(x)$ 和一个定义在 (a, b) 上的概率密度函数 $p(x)$ 的乘积. 这样整个积分就可以写成

$$\int_a^b h(x)dx = \int_a^b f(x)p(x)dx = \mathbb{E}_{p(x)}[f(x)]. \tag{1.6}$$

这样一来, 原积分就等同于 $f(x)$ 在 $p(x)$ 这个分布上的均值. 这时, 如果我们从分布 $p(x)$ 上采集大量的样本点 $x_1, x_2, \cdots, x_n$, 这些样本符合分布 $p(x)$, 即对 $\forall\, i$, 有

$$\frac{x_i}{\sum_{i=1}^{n} x_i} \approx p(x_i), \tag{1.7}$$

那么, 我们就可以通过这些样本来逼近这个均值

$$\int_a^b h(x)dx = \mathbb{E}_{p(x)}[f(x)] \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i). \tag{1.8}$$

这就是蒙特卡罗方法的**基本思想**. 近年来, 随着**随机化模型**的流行, 蒙特卡罗方法在机器学习领域有着越来越广泛的应用.

假如我们现在已经定义好分布 $p(x)$, 那么, 蒙特卡罗方法的一个核心问题是: **如何从这个分布上采集样本?**

一般来讲, 对于经典的分布, 例如对于均匀分布和正态分布等, 都已有比较成熟的算法可以快速地直接生成该分布下的无偏 (unbiased) 样本. 然而, 对于任意的分布, 我们并不能做到这一点. 那么, 如何在任意分布下采样呢? 这就是**马尔可夫链蒙特卡罗方法** (Markov chain Monte Carlo, MCMC) 需要解决的问题.

简单来说, **MCMC 的基本思想**就是利用马尔可夫链来产生指定分布下的样本. 为此, 先简单介绍一下马尔可夫链的基本知识.

§1.4.1 马尔可夫链

设 X_t 表示随机变量 X 在离散时间 t 时刻的取值. 若该变量随时间变化的转移概率仅仅依赖于它的当前取值, 即

$$P(X_{t+1}=s_j|X_0=s_{i_0},X_1=s_{i_1},\cdots,X_t=s_i)=P(X_{t+1}=s_j|X_t=s_i), \tag{1.9}$$

则称这个变量为**马尔可夫变量**, 其中 $s_{i_0},s_{i_1},\cdots,s_i,s_j\in\Omega$ 为随机变量 X 可能的状态. 这个性质称为**马尔可夫性质**, 具有马尔可夫性质的随机过程称为**马尔可夫过程**.

由 (1.9) 可知, 对于一个马尔可夫随机变量, 我们只需知道其当前的取值, 就足以充分预测其未来的变化趋势. 而所谓的**马尔可夫链**就是指一段时间内随机变量 X 的取值序列 $(X_0,X_1,\cdots,X_m)$, 它们符合条件 (1.9).

一般来说, 一个马尔可夫链可通过其对应的转移概率来定义. 所谓的转移概率, 是指随机变量从一个时刻到下一个时刻, 从状态 s_i 转移到另一个状态 s_j 的概率, 即

$$P(i\to j):=P_{i,j}=P(X_{t+1}=s_j|X_t=s_i). \tag{1.10}$$

若记 $\pi_k^{(t)}$ 表示随机变量 X 在时刻 t 取值 s_k 的概率, 则 X 在时刻 $t+1$ 取值为 s_i 的概率为

$$\begin{aligned}\pi_i^{(t+1)} &= P(X_{t+1}=s_i)\\ &= \sum_k P(X_{t+1}=s_i|X_t=s_k)\cdot P(X_t=s_k)\\ &= \sum_k P_{k,i}\cdot\pi_k^{(t)}.\end{aligned} \tag{1.11}$$

设状态的数目为 n, 则根据 (1.11), 有

$$(\pi_1^{(t+1)},\cdots,\pi_n^{(t+1)})=(\pi_1^{(t)},\cdots,\pi_n^{(t)})\begin{bmatrix}P_{1,1}&P_{1,2}&\cdots&P_{1,n}\\P_{2,1}&P_{2,2}&\cdots&P_{2,n}\\&&\cdots&\\P_{n,1}&P_{n,2}&\cdots&P_{n,n}\end{bmatrix},$$

上式也可写成矩阵向量形式

$$\pi^{(t+1)}=\pi^{(t)}\cdot\mathbf{P}, \tag{1.12}$$

其中 $\pi^{(r)}=(\pi_1^{(r)},\cdots,\pi_n^{(r)})$, $r=t,t+1$ 为行向量, $\mathbf{P}=(P_{i,j})_{n\times n}$ 为**转移概率矩阵**.

如果存在某个取值, 从它出发转移回自身所需要的转移次数总是整数 $d(>1)$ 的倍数, 那么这个马尔可夫过程就具有**周期性** (Periodic). 如果任意两个取值之间总是能以非零的概率相互转移, 那么该马尔可夫过程就称为**不可约** (Irreducible) ("不可约" 是指每一个状态都可来自任意的其它状态). 如果一个马尔可夫过程既没有周期性, 又不可约, 则称它是**各态遍历的** (Ergodic).

对于各态遍历的马尔可夫过程, 不论 $\pi^{(0)}$ 取何值, 随着转移次数的增多, 随机变量的取值分布最终都会收敛于**唯一的**平稳分布 π^*, 即

$$\lim_{t\to\infty} \pi^{(0)}\mathbf{P}^t = \pi^*, \tag{1.13}$$

且这个平稳分布 π^* 满足

$$\pi^*\mathbf{P} = \pi^*. \tag{1.14}$$

这意味着, 不论 $\pi^{(0)}$ 取何值, 经过足够多次转移后, 随机变量各取值总会不断接近于该过程的平稳分布. 这为 MCMC 建立了理论基础: **如果我们想在某个分布下采样, 只需要模拟以其为平稳分布的马尔科夫过程, 经过足够多次转移之后, 我们的样本分布就会充分接近于该平稳分布.** 这意味着我们近似地采集到了目标分布下的样本.

§1.4.2 正则分布

假设一个物理系统具备一定的自由度 (例如, 一滴水珠里的水分子在空间中可以任意排列), 那么, 系统所处的状态 (所有水分子的位置) 就具备一定的随机性. 假设系统处于状态 i 的概率为 p_i, 则显然有

$$p_i \geq 0, \text{ 且 } \sum_i p_i = 1. \tag{1.15}$$

根据系统的物理性质, 不同的状态可能会使系统具备不同的能量. 我们用 E_i 来表示系统处于状态 i 时的能量. **统计力学**的一个基本结论是: 当系统与外界达到热平衡时, 系统处于状态 i 的概率 p_i 具有以下形式

$$p_i = \frac{1}{Z_T} e^{-\frac{E_i}{T}}, \tag{1.16}$$

其中

$$Z_T = \sum_i e^{-\frac{E_i}{T}} \tag{1.17}$$

被称为**归一化常数** (Normalizing Constant), T 为正数, 表示系统所处的温度. (1.16) 这种概率分布的形式叫做**正则分布**.

从这个分布形式我们可以看到, 同一温度下, 能量越小的状态具有越大的概率. 另一方面, 当温度 T 升高时, 概率分布会对能量越来越不敏感, 并逐渐趋近于**均匀分布**. 特别是当 $T \to \infty$ 时, 整个分布完全退化为均匀分布, 这时系统的状态变得完全随机. 以水滴作为例子, 此时所有的水分子将不再受整个系统能量的限制, 而四散开来, 宏观上看到的就是蒸发.

在机器学习领域, 人们通常根据需求自定**能量函数**, 然后借鉴物理规律去实现其他的功能, 而正则分布就是沟通两者的桥梁.

§1.4.3 Metropolis-Hastings 采样

在 MCMC 算法中, 为了在一个指定的分布上采样, 我们首先从系统的任意一个状态出发, 然后模拟马尔可夫过程, 不断进行状态转移. 根据马尔可夫的性质, 在经过足够的转移次数之后, 我们所处的状态即符合目标分布, 这时, 该状态就可以作为一个采集到的样本. 而算法的关键就是, **设计出合理的状态转移过程**. Metropolis-Hastings 是一种非常重要的 MCMC 采样算法, 并且对于设计状态转移过程建立了**统一的框架**.

在 Metropolis-Hastings 算法中, 假设我们想从分布 $\pi(\cdot)$ 上采集样本, 那么, 我们引入另一组**转移提议分布** (proposal density) $Q(.;i)$. 这个分布的作用是, 根据我们的当前状态 i, 提议转移之后的状态. 每次状态转移时, 我们首先利用 $Q(.;i)$ 提议出下一步的状态, 假设为 j, 然后, 我们以下面的概率接受这个状态

$$\alpha(i \to j) = \min\left\{1, \frac{\pi(j)Q(i;j)}{\pi(i)Q(j;i)}\right\}, \tag{1.18}$$

(1.18) 中, $\alpha(i \to j)$ 被称作**接受概率**.

为了模拟接受新状态 j 的过程, 我们可以首先产生一个 $[0,1]$ 之间的均匀分布的随机数 r, 然后, 如果 $r < \alpha(i \to j)$, 则采用状态 j 作为新状态, 否则维持状态 i 不变. $Q(j;i)$ 表示从状态 i 提议转移到状态 j 的概率. 一般来说, $Q(.,.)$ 可以选择为一些比较简单的概率分布.

下面我们简单分析 Metropolis-Hastings 算法的正确性.

显然, 对于上面的例子, 状态 i 转移到状态 j 的转移概率为

$$P(i \to j) = \alpha(i \to j)Q(i;j), \tag{1.19}$$

于是很容易验证: 对于任意的状态 i, j, 成立如下的所谓**细致平衡条件** (detailed balance)

$$\begin{aligned}
\pi(i)P(i \to j) &= \pi(i)\alpha(i \to j)Q(i;j) \\
&= \pi(i)\min\left\{1, \frac{\pi(j)Q(i;j)}{\pi(i)Q(j;i)}\right\}Q(i;j) \\
&= \min\{\pi(i)Q(i;j), \pi(j)Q(i;j)\} \\
&= \pi(j)\min\left\{\frac{\pi(i)Q(i;j)}{\pi(j)Q(i;j)}, 1\right\}Q(i;j) \\
&= \pi(j)\alpha(j \to i)Q(i;j) \\
&= \pi(j)P(j \to i).
\end{aligned}$$

当细致平衡条件满足时, 就可以证明状态分布 $\pi(\cdot)$ 在转移 $P(i \to j)$ 下保持不变, 即

$$\sum_j \pi(j)P(j \to i) = \sum_j \pi(i)P(i \to j) = \pi(i)\sum_j P(i \to j) = \pi(i).$$

若该马尔可夫过程满足各态遍历性, 那么, 根据稳定分布的唯一性, 我们就知道在转移 $\mathbf{P}$ 下, 状态的分布最终会收敛于 $\pi(\cdot)$, 也就是我们要采样的分布.

§1.4.4 Gibbs 采样

Gibbs 采样 (Gibbs Sampling) 是 Metropolis-Hastings 的特殊形式. 它应用于系统具有多个变量, 并且对于变量间的条件分布我们能够直接采样的情况下. Metropolis-Hastings 作为一个通用的框架, 它的缺点就在于它过于灵活. 转移提议分布 $Q(.;.)$ 如果选择得不好, 很有可能每次提议都被拒绝 (即 $\alpha(i \to j)$ 总是很小), 于是造成状态迟迟维持不变, 影响马尔可夫链收敛的速度.

在 Gibbs 采样中, 我们假设系统由 m 个变量构成, 不妨定义系统状态 $X \equiv (x_1, x_2, \cdots, x_m)$, 并且对于任一变量 x_i, 我们能够直接从条件分布 $P(x_i|x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_m)$ 中为其采样.

Gibbs 采样算法的流程如下:

算法 1.1 (Gibbs 采样算法)

1. 初始化系统状态为 $X^{(0)}$.
2. 初始化时间 $t \leftarrow 0$.
3. 对每个变量 x_i, $i \in \{1, 2, \cdots, m\}$, 按以下条件概率对其采样
$$P(x_i^{(t+1)}|x_1^{(t+1)}, \cdots, x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, \cdots, x_m^{(t)}).$$
4. $t \leftarrow t + 1$.
5. 若 t 少于足够的转移次数, 则返回第 3 步.
6. 返回 $X^{(t)}$ 作为采集到的样本.

和 Metropolis-Hastings 采样相比, Gibbs 采样的一大特点是, 不存在接受概率 α, 因此, 状态转移总是能够实行. 所以它往往比 Metropolis-Hastings 具有**更快的收敛速度**.

§2 RBM 的网络结构

RBM 包含两个层 — **可见层** (visible layer) 和**隐藏层** (hidden layer). 神经元之间的连接具有如下特点: **层内无连接, 层间全连接**. 这里, **层内**既包括可见层内的神经元也包括隐藏层内的神经元; 而**全连接**指的是, 可见层中的每一个神经元与隐藏层中的所有神经元都有连接, 同时, 隐藏层中的每一个神经元也与可见层中的所有神经元都有连接. 由此可知, RBM 对应的图 (神经元当作顶点神经元之间的连接当作边) 是一个**二分图**.

可见层单元用来描述观察数据的一个方面或一个特征, 例如, 对于黑白图片而言, 一个可见单元可能描述的就是图片某处是否为白色. 而隐藏层单元的意义一般来说并不明确, 也无法认为指定, 它们用来获取可见层单元对应变量之间的依赖关系, 因此有时也叫做特征提取层.

注意, RBM 和 BM 的不同之处在于层内神经元的连接上, 前者要求层内神经元之间没有连接, 而后者允许层内神经元之间有连接. 正是由于这个限制, RBM 具有如下**性质**:

性质 2.1 *当给定可见层神经元的状态时, 各隐藏层神经元的激活条件独立; 反之当给定隐藏层神经元的状态时, 可见层神经元的激活也条件独立.*

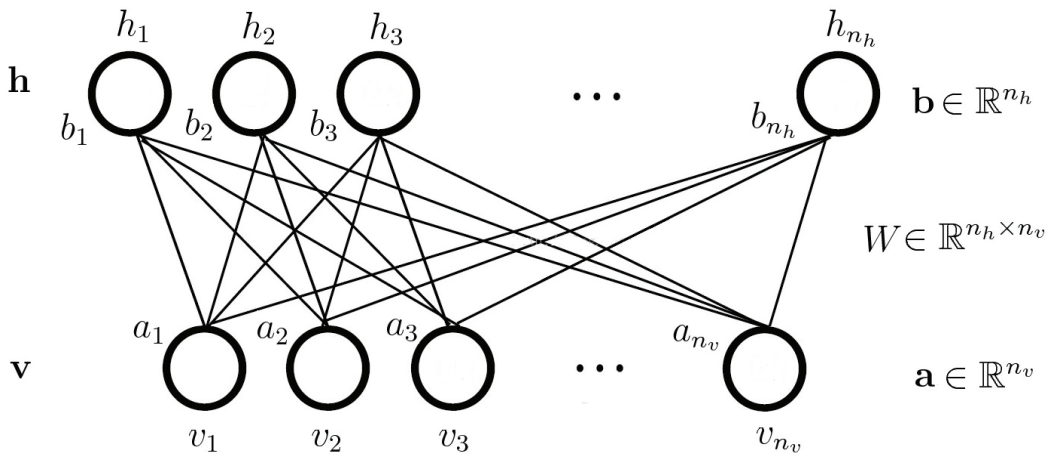

图 2 RBM 网络结构示意图

图 2 给出了一个 RBM 网络结构的示意图, 其中

- n_v, n_h: 分别表示可见层和隐藏层中包含**神经元的数目**. 下标 v, h 代表 visible 和 hidden.
- $\mathbf{v} = (v_1, v_2, \cdots, v_{n_v})^T$: 可见层的**状态向量**, v_i 表示可见层中第 i 个神经元的状态.
- $\mathbf{h} = (h_1, h_2, \cdots, h_{n_h})^T$: 隐藏层的**状态向量**, h_j 表示隐藏层中第 j 个神经元的状态.
- $\mathbf{a} = (a_1, a_2, \cdots, a_{n_v})^T \in \mathbb{R}^{n_v}$: 可见层的**偏置向量**, a_i 表示可见层中第 i 个神经元的偏置.
- $\mathbf{b} = (b_1, b_2, \cdots, b_{n_h})^T \in \mathbb{R}^{n_h}$: 隐藏层的**偏置向量**, b_j 表示隐藏层中第 j 个神经元的偏置.

- $W=(w_{i,j})\in\mathbb{R}^{n_h\times n_v}$: 隐藏层和可见层之间的**权值矩阵**, $w_{i,j}$ 表示隐藏层中第 i 个神经元与可见层中第 j 个神经元之间的连接权重.

注 2.1 本文我们取 $W\in\mathbb{R}^{n_h\times n_v}$, 采用了一般神经网络中连接权值矩阵的记法. 也有文献取 $W\in\mathbb{R}^{n_v\times n_h}$, 此时, $w_{i,j}$ 表示可见层中第 i 个神经元与隐藏层中第 j 个神经元之间的连接权重. 两者本质上没什么不同, 在相应的数学公式中注意指标匹配就好.

记 $\theta=(W,\mathbf{a},\mathbf{b})$ 表示 RBM 中的参数, 可将其视为把 $W,\mathbf{a},\mathbf{b}$ 中的所有分量拼接起来得到的长向量.

此外, 为讨论方便起见, 我们假定 RBM 中所有神经元均为**二值的**, 即对 $\forall\, i,j$, 有 $v_i,h_j\in\{0,1\}$. 这样的 RBM 也称为 Binary RBM.

§3 能量函数和概率分布

RBM 模型是一个基于能量的模型 (Energy Based Model, EBM), 因此, 我们首先为其定义一个能量函数, 并利用该能量函数引入一系列相关的概率分布函数.

对于一组给定的状态 $(\mathbf{v}, \mathbf{h})$, 可定义如下**能量函数**

$$E_\theta(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{n_v} a_i v_i - \sum_{j=1}^{n_h} b_j h_j - \sum_{i=1}^{n_v}\sum_{j=1}^{n_h} h_j w_{j,i} v_i, \tag{3.20}$$

上式为分量形式, 也可紧凑地将其写成如下矩阵 - 向量形式

$$E_\theta(\mathbf{v}, \mathbf{h}) = -\mathbf{a}^T\mathbf{v} - \mathbf{b}^T\mathbf{h} - \mathbf{h}^T W\mathbf{v}. \tag{3.21}$$

利用 (3.20) 定义的能量函数, 可以给出状态 $(\mathbf{v}, \mathbf{h})$ 的**联合概率分布**

$$P_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{Z_\theta} e^{-E_\theta(\mathbf{v},\mathbf{h})}, \tag{3.22}$$

其中

$$Z_\theta = \sum_{\mathbf{v},\mathbf{h}} e^{-E_\theta(\mathbf{v},\mathbf{h})} \tag{3.23}$$

为**归一化因子**, 也称为**配分函数** (partition function).

对于一个实际问题, 我们最关心的是关于观测数据 $\mathbf{v}$ 的概率分布 $P_\theta(\mathbf{v})$, 它对应 $P_\theta(\mathbf{v}, \mathbf{h})$ 的边缘分布, 也称为**似然函数** (likelihood function), 具体为

$$P_\theta(\mathbf{v}) = \sum_{\mathbf{h}} P_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{Z_\theta}\sum_{\mathbf{h}} e^{-E_\theta(\mathbf{v},\mathbf{h})}, \tag{3.24}$$

类似地, 我们有

$$P_\theta(\mathbf{h}) = \sum_{\mathbf{v}} P_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{Z_\theta}\sum_{\mathbf{v}} e^{-E_\theta(\mathbf{v},\mathbf{h})}. \tag{3.25}$$

注 3.1 *由 (3.24) 和 (3.25) 可见, 要确定 $P_\theta(\mathbf{v})$ 或 $P_\theta(\mathbf{h})$, 均需计算 Z_θ, 而根据 Z_θ 的定义式 (3.23), $\sum_{\mathbf{v},\mathbf{h}}$ 中包含 $2^{n_v+n_h}$ 个项, 由此可知, 若直接计算, 其计算复杂度是很大的.*

为记号上简单起见, 在不引起混淆的前提下, 我们忽略下标 θ, 即把 P_θ, Z_θ, E_θ 分别简记为 P, Z, E.

接下来, 考虑当给定可见层 (或隐藏层) 上所有神经元的状态时, 隐藏层 (或可见层) 上某个神经单元被激活 (即取值为 1) 的概率, 即计算 $P(h_k = 1|\mathbf{v})$ (或 $P(v_k = 1|\mathbf{h})$), **它们将在下文中推导梯度公式时用到**. 由于两者的推导完全类似, 这里仅以 $P(h_k = 1|\mathbf{v})$ 为例来给出推导过程.

为使得推导过程在记号上简单起见, 记

$$\mathbf{h}_{-k} = (h_1, h_2, \cdots, h_{k-1}, h_{k+1}, \cdots, h_{n_h})^T \tag{3.26}$$

表示在 $\mathbf{h}$ 中挖掉分量 h_k 后得到的向量, 并引入

$$\alpha_k(\mathbf{v}) = b_k + \sum_{i=1}^{n_v} w_{k,i} v_i, \tag{3.27}$$

以及

$$\beta(\mathbf{v}, h_{-k}) = \sum_{i=1}^{n_v} a_i v_i + \sum_{\substack{j=1 \\ j\neq k}}^{n_h} b_j h_j + \sum_{i=1}^{n_v} \sum_{\substack{j=1 \\ j\neq k}}^{n_h} h_j w_{j,i} v_i, \tag{3.28}$$

容易验证, 利用上述两个式子, 有

$$E(\mathbf{v}, \mathbf{h}) = -\beta(\mathbf{v}, h_{-k}) - h_k \alpha_k(\mathbf{v}), \tag{3.29}$$

事实上, $-h_k\alpha_k(\mathbf{v})$ 和 $-\beta(\mathbf{v}, h_{-k})$ 分别对应 $E(\mathbf{v}, \mathbf{h})$ 的定义式 (3.20) 中下标 $j = k$ 和 $j \neq k$ 的部分.

下面给出 $P(h_k = 1|\mathbf{v})$ 的推导过程.

$$\begin{aligned}
& P(h_k = 1 \mid \mathbf{v}) \\
= \ & P(h_k = 1 \mid h_{-k}, \mathbf{v}) \quad \text{(等价式)} \\
= \ & \frac{P(h_k = 1, h_{-k}, \mathbf{v})}{P(h_{-k}, \mathbf{v})} \quad \text{(Bayes 公式)} \\
= \ & \frac{P(h_k = 1, h_{-k}, \mathbf{v})}{P(h_k = 1, h_{-k}, \mathbf{v}) + P(h_k = 0, h_{-k}, \mathbf{v})} \quad (h_k = 0 \text{ 或 } 1) \\
= \ & \frac{\frac{1}{Z} e^{-E(h_k=1, h_{-k}, \mathbf{v})}}{\frac{1}{Z} e^{-E(h_k=1, h_{-k}, \mathbf{v})} + \frac{1}{Z} e^{-E(h_k=0, h_{-k}, \mathbf{v})}} \quad \text{(利用联合概率公式 (3.22))} \\
= \ & \frac{e^{-E(h_k=1, h_{-k}, \mathbf{v})}}{e^{-E(h_k=1, h_{-k}, \mathbf{v})} + e^{-E(h_k=0, h_{-k}, \mathbf{v})}} \quad \text{(分子分母约掉 } \tfrac{1}{Z}\text{)} \\
= \ & \frac{1}{1 + e^{-E(h_k=0, h_{-k}, \mathbf{v}) \ + \ E(h_k=1, h_{-k}, \mathbf{v})}} \quad \text{(分子分母同乘 } e^{E(h_k=1, h_{-k}, \mathbf{v})}\text{)} \\
= \ & \frac{1}{1 + e^{[\beta(\mathbf{v}, h_{-k}) + \mathbf{0}\cdot\alpha_k(\mathbf{v})] \ + \ [-\beta(\mathbf{v}, h_{-k}) - \mathbf{1}\cdot\alpha_k(\mathbf{v})]}} \quad \text{(利用公式 (3.29))} \\
= \ & \frac{1}{1 + e^{-\alpha_k(\mathbf{v})}} \quad \text{(化简)} \\
= \ & sigmoid(\alpha_k(\mathbf{v})) \quad \text{(sigmoid 函数定义)} \\
= \ & sigmoid(b_k + \sum_{i=1}^{n_v} w_{k,i} v_i) \quad \text{(定义式 (3.27))}
\end{aligned}$$

经过上述推导, 我们得到了公式

$$P(h_k = 1 \mid \mathbf{v}) = sigmoid(b_k + \sum_{i=1}^{n_v} w_{k,i} v_i), \tag{3.30}$$

通过完全类似的推导, 也可得公式

$$P(v_k = 1 \mid \mathbf{h}) = sigmoid(a_k + \sum_{j=1}^{n_h} w_{j,k} h_j). \tag{3.31}$$

最后, 利用上述记号, 我们给出性质 2.1 (见 §2) 的数学表示如下

$$P(\mathbf{h}|\mathbf{v}) = \prod_{j=1}^{n_h} P(h_j|\mathbf{v}); \quad P(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{n_v} P(v_i|\mathbf{h}). \tag{3.32}$$

注 3.2 关于 **"专家乘积"** 模型 *(a product of experts model)*

RBM 的相关文献中经常会提到 "专家乘积" 的概念, 简单来说, 就是指 $P(\mathbf{v})$ 能表示成如下的连乘形式

$$\begin{aligned}
P(\mathbf{v}) &= \sum_{\mathbf{h}} P(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \\
&= \frac{1}{Z}\sum_{\mathbf{h}} e^{\sum_{i=1}^{n_v} a_i v_i + \sum_{j=1}^{n_h} b_j h_j + \sum_{i=1}^{n_v}\sum_{j=1}^{n_h} h_j w_{j,i} v_i} \\
&= \frac{1}{Z}\sum_{h_1}\sum_{h_2}\cdots\sum_{h_{n_h}} e^{\sum_{i=1}^{n_v} a_i v_i + \sum_{j=1}^{n_h} b_j h_j + \sum_{i=1}^{n_v}\sum_{j=1}^{n_h} h_j w_{j,i} v_i} \\
&= \frac{1}{Z}\sum_{h_1}\sum_{h_2}\cdots\sum_{h_{n_h}} e^{\sum_{i=1}^{n_v} a_i v_i} e^{\sum_{j=1}^{n_h} h_j\cdot(b_j + \sum_{i=1}^{n_v} w_{j,i} v_i)} \\
&= \frac{1}{Z} e^{\sum_{i=1}^{n_v} a_i v_i} \sum_{h_1}\sum_{h_2}\cdots\sum_{h_{n_h}}\prod_{j=1}^{n_h} e^{h_j\cdot(b_j + \sum_{i=1}^{n_v} w_{j,i} v_i)} \\
&= \frac{1}{Z} e^{\sum_{i=1}^{n_v} a_i v_i} \sum_{h_1} e^{h_1\cdot(b_1 + \sum_{i=1}^{n_v} w_{1,i} v_i)} \sum_{h_2} e^{h_2\cdot(b_2 + \sum_{i=1}^{n_v} w_{2,i} v_i)} \cdots \sum_{h_{n_h}} e^{h_{n_h}\cdot(b_{n_h} + \sum_{i=1}^{n_v} w_{n_h,i} v_i)} \\
&= \frac{1}{Z} e^{\sum_{i=1}^{n_v} a_i v_i} \prod_{j=1}^{n_h}\sum_{h_j} e^{h_j\cdot(b_j + \sum_{i=1}^{n_v} w_{j,i} v_i)} \\
&= \frac{1}{Z}\prod_{i=1}^{n_v} e^{a_i v_i} \prod_{j=1}^{n_h}\sum_{h_j} e^{h_j\cdot(b_j + \sum_{i=1}^{n_v} w_{j,i} v_i)} \\
&= \frac{1}{Z}\prod_{i=1}^{n_v} e^{a_i v_i} \prod_{j=1}^{n_h}(e^{0\cdot(b_j + \sum_{i=1}^{n_v} w_{j,i} v_i)} + e^{1\cdot(b_j + \sum_{i=1}^{n_v} w_{j,i} v_i)}) \\
&= \frac{1}{Z}\prod_{i=1}^{n_v} e^{a_i v_i} \prod_{j=1}^{n_h}(1 + e^{b_j + \sum_{i=1}^{n_v} w_{j,i} v_i})
\end{aligned}$$

具体可参考文献 *[4]* 和 *[5]*.

§4 对数似然函数

给定训练样本后, 训练一个 RBM 意味着调整参数 θ, 以拟合给定的训练样本, 即, 使得在该参数下由相应 RBM 表示的概率分布尽可能地与训练数据相符合. 下面从数学上来进行描述.

假定训练样本集合为

$$\mathbf{S} = \{\mathbf{v}^1, \mathbf{v}^2, \cdots, \mathbf{v}^{n_s}\}, \tag{4.33}$$

其中 n_s 为训练样本的数目, $\mathbf{v}^i = (v_1^i, v_2^i, \cdots, v_{n_v}^i)^T$, $i = 1, 2, \cdots, n_s$, 它们是**独立同分布** (independent and identically distributed, i.i.d) 的. 训练 RBM 的目标就是最大化似然度

$$\mathcal{L}_{\theta,\mathbf{S}} = \prod_{i=1}^{n_s} P(\mathbf{v}^i). \tag{4.34}$$

注意, 连乘式 $\prod_{i=1}^{n_s} P(\mathbf{v}^i)$ 处理起来比较麻烦, 由函数 $\ln x$ 的严格单调性可知, 最大化 $\mathcal{L}_{\theta,\mathbf{S}}$ 与最大化 $\ln \mathcal{L}_{\theta,\mathbf{S}}$ 是**等价的**, 因此, 训练 RBM 的目标就变成了最大化

$$\ln \mathcal{L}_{\theta,\mathbf{S}} = \ln \prod_{i=1}^{n_s} P(\mathbf{v}^i) = \sum_{i=1}^{n_s} \ln P(\mathbf{v}^i). \tag{4.35}$$

为简洁起见, 下文我们省略下标 θ, 将 $\mathcal{L}_{\theta,\mathbf{S}}$ 简记为 $\mathcal{L}_{\mathbf{S}}$.

§5 梯度计算公式

最大化 (4.35) 常用的数值方法是**梯度上升** (gradient ascent) 法, 通过迭代的方式来进行逼近, 迭代格式为

$$\theta := \theta + \eta \frac{\partial \ln \mathcal{L}_{\mathbf{S}}}{\partial \theta}. \tag{5.36}$$

其中 $\eta > 0$ 为**学习率**. 由 (5.36) 可见, 关键是梯度 $\frac{\partial \ln \mathcal{L}_{\mathbf{S}}}{\partial \theta}$ ($\ln \mathcal{L}_{\mathbf{S}}$ 关于各个参数的偏导数) 的计算. 为此, 我们首先考虑**只含单个训练样本** $\mathbf{v}$ 的情形下梯度的计算, 此时

$$\begin{aligned}
\ln \mathcal{L}_{\mathbf{S}} &= \ln P(\mathbf{v}) \\
&= \ln \left(\frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \right) \\
&= \ln \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} - \ln Z \\
&= \ln \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} - \ln \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})},
\end{aligned} \tag{5.37}$$

注 5.1 *公式 (5.37) 及以下的推导公式中将涉及两个 $\mathbf{v}$: 一个表示特定的单个训练样本 $\mathbf{v}$, 另一个是 $\sum_{\mathbf{v},\mathbf{h}}$ 中任意的 $\mathbf{v}$, 为对两者区分起见, 特将前者标为红色, 从而避免引入新的记号.*

由 (5.37), 有

$$\begin{aligned}
&\frac{\partial \ln P(\mathbf{v})}{\partial \theta} \\
=\ & \frac{\partial}{\partial \theta} \left(\ln \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \right) - \frac{\partial}{\partial \theta} \left(\ln \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \right) \\
=\ & -\frac{1}{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} + \frac{1}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}} \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \\
=\ & -\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} + \sum_{\mathbf{v},\mathbf{h}} P(\mathbf{v},\mathbf{h}) \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta},
\end{aligned} \tag{5.38}$$

其中最后一个等式中的第一项用到了

$$\frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}} = \frac{\frac{e^{-E(\mathbf{v},\mathbf{h})}}{Z}}{\frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}{Z}} = \frac{\frac{e^{-E(\mathbf{v},\mathbf{h})}}{Z}}{\sum_{\mathbf{h}} \frac{e^{-E(\mathbf{v},\mathbf{h})}}{Z}} = \frac{P(\mathbf{v},\mathbf{h})}{P(\mathbf{v})} = P(\mathbf{h}|\mathbf{v}).$$

注意, (5.38) 右端的两项分别对应两个**期望** (expectation):

- 第一项 $\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta}$ 对应能量梯度函数 $\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta}$ 在条件分布 $P(\mathbf{h}|\mathbf{v})$ 下的期望;
- 第二项 $\sum_{\mathbf{v},\mathbf{h}} P(\mathbf{v},\mathbf{h}) \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta}$ 对应能量梯度函数 $\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta}$ 在联合分布 $P(\mathbf{v},\mathbf{h})$ 下的期望.

因此, 文献中也常将 (5.38) 写成

$$\frac{\partial \ln P(\mathbf{v})}{\partial \theta} = -\mathbb{E}_{P(\mathbf{h}|\mathbf{v})}\left[\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta}\right] + \mathbb{E}_{P(\mathbf{v},\mathbf{h})}\left[\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta}\right] \tag{5.39}$$

或者

$$\frac{\partial \ln P(\mathbf{v})}{\partial \theta} = -\left\langle \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \right\rangle_{P(\mathbf{h}|\mathbf{v})} + \left\langle \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \right\rangle_{P(\mathbf{v},\mathbf{h})} \tag{5.40}$$

的形式, 其中 $\mathbb{E}_P[\cdot]$ 和 $<\cdot>_P$ 表示求关于分布 P 的数学期望.

注 5.2 *$P(\mathbf{h}|\mathbf{v})$ 表示的是在可见单元限定为已知的训练样本 v 时隐藏层的概率分布, 因此 (5.38) 中的第一项比较容易计算; 但 $P(\mathbf{v},\mathbf{h})$ 表示的是可见层单元和隐藏层单元的联合分布, 其中涉及归一化因子 Z, 因此该分布很难获取, 导致我们无法直接计算 (5.38) 中的第二项, 只能通过一些采样方法获取其近似值 ([3]). 这些在下面的推导中可以看到.*

接下来, 我们讨论这两个期望的计算. 注意, 由于

$$\begin{aligned}\sum_{\mathbf{v},\mathbf{h}} P(\mathbf{v},\mathbf{h})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} &= \sum_{\mathbf{v}}\sum_{\mathbf{h}} P(\mathbf{v})P(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \\ &= \sum_{\mathbf{v}} P(\mathbf{v}) \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta},\end{aligned} \tag{5.41}$$

因此, 只需讨论 $\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta}$ 的计算即可. 下面按照 θ 取 $w_{i,j}$, a_i 和 b_i 分别进行介绍.

(1) $\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{i,j}}$ 的计算

$$\begin{aligned}&\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{i,j}} \\ =\ & -\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) h_i v_j \quad (\text{利用 } \tfrac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{i,j}} = -h_i v_j) \\ =\ & -\sum_{\mathbf{h}} \prod_{k=1}^{n_h} P(h_k|\mathbf{v}) h_i v_j \quad (\text{利用 (3.32) 式}) \\ =\ & -\sum_{\mathbf{h}} P(h_i|\mathbf{v}) P(h_{-i}|\mathbf{v}) h_i v_j \quad (\text{独立性, } h_i \text{ 的定义见 (3.26) 式}) \\ =\ & -\sum_{h_i}\sum_{h_{-i}} P(h_i|\mathbf{v}) P(h_{-i}|\mathbf{v}) h_i v_j \quad (\text{对} \textstyle\sum_{\mathbf{h}} \text{进行拆分}) \\ =\ & -\sum_{h_i} P(h_i|\mathbf{v}) h_i v_j \sum_{h_{-i}} P(h_{-i}|\mathbf{v}) \quad (\text{求和号下的项进行分组}) \\ =\ & -\sum_{h_i} P(h_i|\mathbf{v}) h_i v_j \quad (\because \textstyle\sum_{h_{-i}} P(h_{-i}|\mathbf{v}) = 1) \\ =\ & -(\ P(h_i=0|\mathbf{v})\cdot 0 \cdot v_j + P(h_i=1|\mathbf{v})\cdot 1 \cdot v_j\ ) \quad (\because\ h_i = 0 \text{ 或 } 1) \\ =\ & -P(h_i=1|\mathbf{v}) v_j.\end{aligned} \tag{5.42}$$

(2) $\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial a_i}$ 的计算

$$\begin{aligned}
&\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial a_i} \\
=\ & -\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) v_i \quad (\text{利用 } \tfrac{\partial E(\mathbf{v},\mathbf{h})}{\partial a_i} = -v_i) \\
=\ & -v_i \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) \quad (\text{提出 } v_i) \\
=\ & -v_i \quad (\because \textstyle\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) = 1)
\end{aligned} \tag{5.43}$$

(3) $\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial b_i}$ 的计算 (与 $\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{i,j}}$ 的计算完全类似)

$$\begin{aligned}
&\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial b_i} \\
=\ & -\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) h_i \quad (\text{利用 } \tfrac{\partial E(\mathbf{v},\mathbf{h})}{\partial b_i} = -h_i) \\
=\ & -\sum_{\mathbf{h}} \prod_{k=1}^{n_h} P(h_k|\mathbf{v}) h_i \quad (\text{利用 (3.32) 式}) \\
=\ & -\sum_{\mathbf{h}} P(h_i|\mathbf{v}) P(h_{-i}|\mathbf{v}) h_i \quad (\text{独立性, } h_i \text{ 的定义见 (3.26) 式}) \\
=\ & -\sum_{h_i}\sum_{h_{-i}} P(h_i|\mathbf{v}) P(h_{-i}|\mathbf{v}) h_i \quad (\text{对 } \textstyle\sum_{\mathbf{h}} \text{ 进行拆分}) \\
=\ & -\sum_{h_i} P(h_i|\mathbf{v}) h_i \sum_{h_{-i}} P(h_{-i}|\mathbf{v}) \quad (\text{求和号下的项进行分组}) \\
=\ & -\sum_{h_i} P(h_i|\mathbf{v}) h_i \quad (\because \textstyle\sum_{h_{-i}} P(h_{-i}|\mathbf{v}) = 1) \\
=\ & -\left(P(h_i = 0|\mathbf{v}) \cdot 0 + P(h_i = 1|\mathbf{v}) \cdot 1 \right) \quad (\because\ h_i = 0 \text{ 或 } 1) \\
=\ & -P(h_i = 1|\mathbf{v}).
\end{aligned} \tag{5.44}$$

利用 (5.42)、(5.43) 和 (5.44), 并结合 (5.41), (5.38) 可进一步具体写为

$$\begin{aligned}
\frac{\partial \ln P(\mathbf{v})}{\partial w_{i,j}} &= -\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{i,j}} + \sum_{\mathbf{v},\mathbf{h}} P(\mathbf{v},\mathbf{h})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{i,j}} \\
&= P(h_i = 1|\mathbf{v}) v_j - \sum_{\mathbf{v}} P(\mathbf{v}) P(h_i = 1|\mathbf{v}) v_j,
\end{aligned} \tag{5.45}$$

$$\begin{aligned}
\frac{\partial \ln P(\mathbf{v})}{\partial a_i} &= -\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial a_i} + \sum_{\mathbf{v},\mathbf{h}} P(\mathbf{v},\mathbf{h})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial a_i} \\
&= v_i - \sum_{\mathbf{v}} P(\mathbf{v}) v_i,
\end{aligned} \tag{5.46}$$

$$\begin{aligned}\frac{\partial \ln P(\mathbf{v})}{\partial b_i} &= -\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial b_i} + \sum_{\mathbf{v},\mathbf{h}} P(\mathbf{v},\mathbf{h})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial b_i} \\ &= P(h_i=1|\mathbf{v}) - \sum_{\mathbf{v}} P(\mathbf{v})P(h_i=1|\mathbf{v}),\end{aligned} \tag{5.47}$$

其中 $P(h_i=1|\mathbf{v})$ 的计算公式见 (3.30).

至此, 我们已经给出了**只含单个训练样本**情形下各偏导数的计算公式. 对于**含多个训练样本**的情形 (即 $\mathbf{S}=\{\mathbf{v}^1,\mathbf{v}^2,\cdots,\mathbf{v}^{n_s}\}$), 利用 (4.35), 有

$$\frac{\partial \ln \mathcal{L}_{\mathbf{S}}}{\partial \theta} = \sum_{m=1}^{n_s} \frac{\partial \ln P(\mathbf{v}^m)}{\partial \theta} \tag{5.48}$$

从而, 可得如下公式

$$\frac{\partial \ln \mathcal{L}_{\mathbf{S}}}{\partial w_{i,j}} = \sum_{m=1}^{n_s}\left[P(h_i=1|\mathbf{v}^m)v_j^m - \sum_{\mathbf{v}} P(\mathbf{v})P(h_i=1|\mathbf{v})v_j\right], \tag{5.49}$$

$$\frac{\partial \ln \mathcal{L}_{\mathbf{S}}}{\partial a_i} = \sum_{m=1}^{n_s}\left[v_i^m - \sum_{\mathbf{v}} P(\mathbf{v})v_i\right], \tag{5.50}$$

$$\frac{\partial \ln \mathcal{L}_{\mathbf{S}}}{\partial b_i} = \sum_{m=1}^{n_s}\left[P(h_i=1|\mathbf{v}^m) - \sum_{\mathbf{v}} P(\mathbf{v})P(h_i=1|\mathbf{v})\right]. \tag{5.51}$$

注 5.3 *公式 (5.49), (5.50) 和 (5.51) 的求和号下的每一项都是独立的, 因此实际计算时, 只需分别针对各个训练样本进行计算, 然后作累加即可.*

上述三个公式中关于 $\sum_{\mathbf{v}}$ 的计算复杂度是 $O(2^{n_v+n_h})$ 的 (见注 3.1), 因此通常采用 MCMC 方法如 Gibbs 采样方法 (见 §1 预备知识中关于 MCMC 以及算法 1.1 的介绍) 进行采样, 并用样本对 $\sum_{\mathbf{v}}$ 这个项进行估计. 然而, 每次执行 MCMC 采样, 都需要进行足够次数的状态转移才能保证采集到的样本符合目标分布, 并且, 我们需要采集大量的样本才能足够精确地进行. 这些需求极大地加重了 RBM 训练的复杂度. 因此, 这种方法虽然理论可行, 但从效率上看是不可取的 ([2]). 下一节我们将介绍一种效率更高的 RBM 训练算法.

§6 对比散度算法

通过常规的 MCMC 采样来估计公式 (5.45), (5.46) 和 (5.47) 中 $\sum_{\mathbf{v}}$ 对应的项之所以十分缓慢, 最大的原因在于需要经过许多步的状态转移才能保证采集到的样本符合目标分布. 然而, **既然我们的目标是让 RBM 拟合训练样本的分布, 那么, 是否可以考虑让 MCMC 的状态以训练样本作为起点呢? 这样一来, 也许这些状态只需要很少次的状态转移就可以抵达 RBM 的分布了** ([2]). 基于这个想法, Hinton 教授于 2002 年发明了**对比散度** (**C**ontrastive **D**ivergence, **CD**) 算法 ([4]), 该方法目前已经成为了训练 RBM 的标准算法.

k 步 CD 算法 (简记为 CD-k) 的步骤很简单, 具体可描述为: 对 $\forall\ \mathbf{v} \in \mathbf{S}$, 取初始值 $\mathbf{v}^{(0)} := \mathbf{v}$, 然后执行 k 步 Gibbs 采样, 其中第 t 步 ($t = 1, 2, \cdots, k$) 先后执行

- 利用 $P(\mathbf{h}|\mathbf{v}^{(t-1)})$ 采样出 $\mathbf{h}^{(t-1)}$;
- 利用 $P(\mathbf{v}|\mathbf{h}^{(t-1)})$ 采样出 $\mathbf{v}^{(t)}$,

采样过程参见算法 6.1 中的函数 sample_h_given_v 和 sample_v_given_h.

接着, 利用 k 步 Gibbs 采样后得到的 $\mathbf{v}^{(k)}$ 来近似估计公式 (5.45), (5.46) 和 (5.47) 中 $\sum_{\mathbf{v}}$ 对应的期望项 (或均值项), 具体为

$$\frac{\partial \ln P(\mathbf{v})}{\partial w_{i,j}} \approx P(h_i = 1|\mathbf{v}^{(0)})v_j^{(0)} - P(h_i = 1|\mathbf{v}^{(k)})v_j^{(k)}, \tag{6.52}$$

$$\frac{\partial \ln P(\mathbf{v})}{\partial a_i} \approx v_i^{(0)} - v_i^{(k)}, \tag{6.53}$$

$$\frac{\partial \ln P(\mathbf{v})}{\partial b_i} \approx P(h_i = 1|\mathbf{v}^{(0)}) - P(h_i = 1|\mathbf{v}^{(k)}). \tag{6.54}$$

事实上, 上述近似过程也可以看成是利用

$$CD_k(\theta, \mathbf{v}) = -\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}^{(0)})\frac{\partial E(\mathbf{v}^{(0)}, \mathbf{h})}{\partial \theta} + \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}^{(k)})\frac{\partial E(\mathbf{v}^{(k)}, \mathbf{h})}{\partial \theta} \tag{6.55}$$

来近似 (5.38) 式

$$\frac{\partial \ln P(\mathbf{v})}{\partial \theta} = -\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}^{(0)})\frac{\partial E(\mathbf{v}^{(0)}, \mathbf{h})}{\partial \theta} + \sum_{\mathbf{v},\mathbf{h}} P(\mathbf{v}, \mathbf{h})\frac{\partial E(\mathbf{v}, \mathbf{h})}{\partial \theta}$$

的过程.

至此, 关于 $\mathcal{L}_{\mathbf{S}}$ 的梯度计算公式都变得具体可算了. 综上, 我们给出一个 CD-k 算法的完整描述. 仍然假定训练样本集合为 $\mathbf{S}$, CD-k 算法的**目标**是获取一次梯度上升迭代中偏导数 $\frac{\partial \mathcal{L}_{\mathbf{S}}}{\partial W}$, $\frac{\partial \mathcal{L}_{\mathbf{S}}}{\partial \mathbf{a}}$ 和 $\frac{\partial \mathcal{L}_{\mathbf{S}}}{\partial \mathbf{b}}$ (其分量为 $\frac{\partial \mathcal{L}_{\mathbf{S}}}{\partial w_{i,j}}$, $\frac{\partial \mathcal{L}_{\mathbf{S}}}{\partial a_i}$ 和 $\frac{\partial \mathcal{L}_{\mathbf{S}}}{\partial b_i}$) 的近似值, 以下分别将其简记为 ΔW, $\Delta \mathbf{a}$ 和

$\Delta\mathbf{b}$ (其分量为 $\Delta w_{i,j}$, Δa_i 和 Δb_i). 此外, 记 RBM(W, $\mathbf{a}$, $\mathbf{b}$) 表示以 W, $\mathbf{a}$, $\mathbf{b}$ 为参数的 RBM 网络. 模块 CDK 的描述如下.

算法 6.1 (CDK (k, $\mathbf{S}$, RBM(W, $\mathbf{a}$, $\mathbf{b}$); ΔW, $\Delta\mathbf{a}$, $\Delta\mathbf{b}$))

- **输入**: k, $\mathbf{S}$, RBM(W, $\mathbf{a}$, $\mathbf{b}$)
- **输出**: ΔW, $\Delta\mathbf{a}$, $\Delta\mathbf{b}$

Step 1 初始化: $\Delta W = 0$, $\Delta\mathbf{a} = 0$, $\Delta\mathbf{b} = 0$.

Step 2 通过对 $\mathbf{S}$ 中的样本循环, 生成 ΔW, $\Delta\mathbf{a}$, $\Delta\mathbf{b}$.

FORALL *the* $\mathbf{v} \in \mathbf{S}$ **DO**

{

$\mathbf{v}^{(0)} := \mathbf{v}$

FOR $t = 0, 1, \cdots, k-1$ **DO**

{

- $\mathbf{h}^{(t)}$ = *sample_h_given_v* ($\mathbf{v}^{(t)}$, RBM(W, $\mathbf{a}$, $\mathbf{b}$));
- $\mathbf{v}^{(t+1)}$ = *sample_v_given_h* ($\mathbf{h}^{(t)}$, RBM(W, $\mathbf{a}$, $\mathbf{b}$)).

}

FOR $i = 1, 2, \cdots, n_v$; $j = 1, 2, \cdots, n_h$ **DO**

{

- $\Delta w_{i,j} = \Delta w_{i,j} + \left[P(h_i = 1|\mathbf{v}^{(0)})v_j^{(0)} - P(h_i = 1|\mathbf{v}^{(k)})v_j^{(k)}\right]$;
- $\Delta a_i = \Delta a_i + \left[v_i^{(0)} - v_i^{(k)}\right]$;
- $\Delta b_i = \Delta b_i + \left[P(h_i = 1|\mathbf{v}^{(0)}) - P(h_i = 1|\mathbf{v}^{(k)})\right]$.

}

}

算法 6.1 中包含两个函数

(1) $\mathbf{h}$ = sample_h_given_v($\mathbf{v}$, RBM(W, $\mathbf{a}$, $\mathbf{b}$))

(2) $\mathbf{v}$ = sample_v_given_h($\mathbf{h}$, RBM(W, $\mathbf{a}$, $\mathbf{b}$))

前者表示已知可见层的 $\mathbf{v}$, 利用 RBM(W, $\mathbf{a}$, $\mathbf{b}$) 网络采样出隐藏层的 $\mathbf{h}$; 后者表示已知隐藏层的 $\mathbf{h}$, 利用 RBM(W, $\mathbf{a}$, $\mathbf{b}$) 网络采样出可见层的 $\mathbf{v}$. 下面对这两个函数进行介绍, 首先介绍第一个函数 sample_h_given_v.

记 $p_j^{\mathbf{v}} = P(h_j = 1|\mathbf{v}), j = 1, 2, \cdots, n_h$, 则 sample_h_given_v 中的计算可写成

FOR $j = 1, 2, \cdots, n_h$ **DO**

{

1. 产生 $[0, 1]$ 上的随机数 r_j.
2. $h_j = \begin{cases} 1, & if\ r_j < p_j^{\mathbf{v}}; \\ 0, & otherwise. \end{cases}$

}

注 6.1 *关于上述操作的理解*

给定可见层上的 $\mathbf{v}$ 时, 隐藏层上第 j 个单元取值为 1 的概率为 $p_j^{\mathbf{v}}$, 那么, 如何根据这个值来确定 h_j 的取值为 0 还是 1 呢?

注意, 区间 $[0, 1]$ 上随机产生的 r_j 落在子区间 $[0, p_j^{\mathbf{v}})$ (即 $r_j < p_j^{\mathbf{v}}$) 的概率也是 $p_j^{\mathbf{v}}$ (这是因为子区间 $[0, p_j^{\mathbf{v}})$ 在 $[0, 1]$ 上所占的长度比例为 $p_j^{\mathbf{v}}$). 既然两个事件的概率是相等的, 我们就可以利用后者来模拟前者, 若 r_j 落在子区间 $[0, p_j^{\mathbf{v}})$, 则认为 h_j 应该取 1, 否则取 0.

第二个函数是完全类似的, 若记 $p_i^{\mathbf{h}} = P(v_i = 1|\mathbf{h}), i = 1, 2, \cdots, n_v$, 则 sample_v_given_h 中的计算可写成

FOR $i = 1, 2, \cdots, n_v$ **DO**

{

1. 产生 $[0, 1]$ 上的随机数 r_i.
2. $v_i = \begin{cases} 1, & if\ r_i < p_i^{\mathbf{h}}; \\ 0, & otherwise. \end{cases}$

}

§7 RBM 训练算法

上一节重点介绍了 CD-k 算法, 在此基础上, 本节我们再给出一个完整的 RBM 训练算法描述.

算法 7.1 (RBM 训练算法)

Step 1 初始化 (Initialization)

(1) 给定训练样本集合 $\mathbf{S}$ ($|\mathbf{S}| = n_s$).

(2) 给定训练周期 J, 学习率 η 以及 $CD\text{-}k$ 算法参数 k.

(3) 指定可见层和隐藏层的单元数目 n_v, n_h.

(4) 初始化偏置向量 $\mathbf{a}$, $\mathbf{b}$ 和权值矩阵 W.

Step 2 训练 (Training)

FOR $iter = 1, 2, \cdots, J$ **DO**

{

1. 调用 $CDK(k, \mathbf{S}, \text{RBM}(W, \mathbf{a}, \mathbf{b}); \Delta W, \Delta\mathbf{a}, \Delta\mathbf{b})$, 生成 ΔW, $\Delta\mathbf{a}$, $\Delta\mathbf{b}$.
2. 刷新参数: $W = W + \eta(\frac{1}{n_s}\Delta W)$, $\mathbf{a} = \mathbf{a} + \eta(\frac{1}{n_s}\Delta\mathbf{a})$, $\mathbf{b} = \mathbf{b} + \eta(\frac{1}{n_s}\Delta\mathbf{b})$.

}

接下来, 我们讨论算法 7.1 中一些重要参数的选取及注意事项 ([3]).

1. **小批量数据**

算法 7.1 中 Step 2 的 FOR 循环中, 我们利用训练样本集合 $\mathbf{S}$, 通过对各样本偏导数的累加一次性生成 ΔW, $\Delta\mathbf{a}$, $\Delta\mathbf{b}$. 实际应用中, 我们常用的做法是: 事先将 $\mathbf{S}$ 分成包含几十或几百个样本的**小批量数据** (mini-batches), 然后逐个计算. 这样效率更高, 因为它适合并行, 尤其是可以充分利用图形处理器 GPU 或 Matlab 中矩阵之间相乘运算的优势.

具体来说, 假定 $\mathbf{S} = \bigcup_{i=1}^{K} S_i$, 其中各 S_i 之间无交集, 且 $|S_1| = |S_2| = \cdots = |S_K| = n_{block}$ (当然全部相等是理想情况, 一般情况下对最后一个子集规模会小一些), 则算法 7.1 中 Step 2 的循环体可以改写为

FOR $l = 1, 2, \cdots, K$ **DO**

{

1. 调用 CDK$(k, S_l, RBM(W, \mathbf{a}, \mathbf{b}); \Delta W, \Delta\mathbf{a}, \Delta\mathbf{b})$, 生成 ΔW, $\Delta\mathbf{a}$, $\Delta\mathbf{b}$.
2. 刷新参数: $W = W + \eta(\frac{1}{n_{block}}\Delta W)$, $\mathbf{a} = \mathbf{a} + \eta(\frac{1}{n_{block}}\Delta\mathbf{a})$, $\mathbf{b} = \mathbf{b} + \eta(\frac{1}{n_{block}}\Delta\mathbf{b})$.

}

由此易见, 算法 7.1 中 Step 2 对应的是 $K = 1$ (亦即 $n_{block} = n_s$) 的情形. 特别地, 当 $n_{block} = 1$ 时, 参数更新则以**在线学习**的方式进行. 注意, 公式中的 $\frac{1}{n_{block}}$ 表示累加的是平均梯度, 加入这个因子, 还可以避免当小批量数据的容量发生改变时学习率 η 也要做出相应修改的问题.

2. **权重和偏置的初始值**

一般地, 权重矩阵 W 可初始化为来自正态分布 $N(0, 0.01)$ 的随机数. 隐藏层偏置 $\mathbf{b}$ 初始化为零. 可见层偏置 $\mathbf{a}$ 按如下公式初始化

$$a_i = \log \frac{p_i}{1 - p_i},$$

其中 p_i 表示训练样本中第 i 个特征处于激活状态 (即取值为 1) 的样本所占的比例.

3. **学习率和动量学习率**

学习率 η 较大时, 收敛速度更快, 但可能引起算法的不稳定, 而 η 较小时, 虽可避免不稳定情况的出现, 但收敛速度变慢. 为克服这一矛盾, 一种做法是在参数更新式中加入**动量项** (momentum), 使本次参数值修改的方向不完全由当前样本下似然函数的梯度方向决定, 而是采用上一次参数值修改方向与本次梯度方向的组合. 在某些情况下, 这可以避免算法过早地收敛到局部最优点. 具体格式可写为

$$\theta := \rho\theta + \eta \frac{\partial \ln \mathcal{L}_{\mathbf{S}}}{\partial \theta}. \tag{7.56}$$

其中 ρ 为**动量项学习率**.

4. **CD-k 算法中的参数 k**

CD-k 算法中的参数 k 通常取得很小, Hinton 发现, 在实际应用中, 甚至取 $k = 1$ 就能保证良好的学习效果了.

关于**可见层和隐藏层的单元数目 n_v, n_h 的选取**, 以及在目标函数中如何加入**正则化项**等, 这里不详细介绍, 具体可参见文 [3] 和 [7].

§8 RBM 的评估

对于一个已经学习得到或者正在学习中的 RBM, 应通过何种指标来评价其优劣呢? 显然, 最简单的指标是该 RBM 对训练数据的似然度, 即由 (4.34) 定义的 $L_{\mathbf{S}}$, 或等价的由 (maximamquality) 定义的 $\ln \mathcal{L}_{\mathbf{S}}$, 但其中均涉及到归一化因子 Z, 计算复杂度相当高. 因此, 只能采用近似方法来进行评估.

常用的近似方法是**重构误差**, 所谓重构误差 (reconstruction error), 就是以训练样本作为初始状态, 经过 RBM 的分布进行一次 Gibbs 转移后与原数据的差异量, 具体可这样计算

error = 0
FORALL $\mathbf{v} \in \mathbf{S}$ **DO**
{
1. $\mathbf{h}$ = sample_h_given_v($\mathbf{v}$, RBM(W, $\mathbf{a}$, $\mathbf{b}$))
2. $\tilde{\mathbf{v}}$ = sample_v_given_h($\mathbf{h}$, RBM(W, $\mathbf{a}$, $\mathbf{b}$))
3. $error = error + \|\mathbf{v} - \tilde{\mathbf{v}}\|$.

}

其中关于函数 sample_h_given_v 和 sample_v_given_h 的定义参见 §6, 范数 $\|\cdot\|$ 采用 1- 范数或 2- 范数 (以 $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$ 为例, $\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$, $\|\mathbf{x}\|_2 = (\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}}$).

重构误差能够在一定程度上反映 RBM 对训练样本的似然度, 不过并不完全可靠. 但总的来说, 其计算相当简单, 因此在实践中非常有用.

另外还有一种叫做**退火式重要性抽样** (**A**nnealed **I**mportance **S**ampling, AIS), 它通过引入一个更简单的辅助分布, 近似计算出归一化因子, 从而直接算出 RBM 对训练数据的似然度, 具体做法请参见文 [2].

参考文献

[1] Asja Fischer, Christian Igel. An Introduction to Restricted Boltzmann Machines. Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications Lecture Notes in Computer Science Volume 7441, 2012, pp 14-36.

[2] 胡洋. 基于马尔可夫链蒙特卡罗方法的 RBM 学习算法改进. 硕士学位论文, 上海交通大学, 2012.

[3] 张春霞, 姬楠楠, 王冠伟. 受限玻尔兹曼机简介 [J]. 2013.

[4] Hinton G.E.: Training products of experts by minimizing contrastive divergence. Neural Computation 14, 1771-1800 (2002)

[5] Welling, M.: Product of experts. Scholarpedia 2(10), 3879 (2007)

[6] 阮一峰的博客 http://www.ruanyifeng.com/blog/2011/08/bayesian_inference_part_one.html

[7] Hinton G. A practical guide to training restricted Boltzmann machines[J]. Momentum, 2010, 9(1): 926.

[8] 《深度学习读书笔记之 RBM》(http://blog.csdn.net/mytestmy/article/details/9150213)