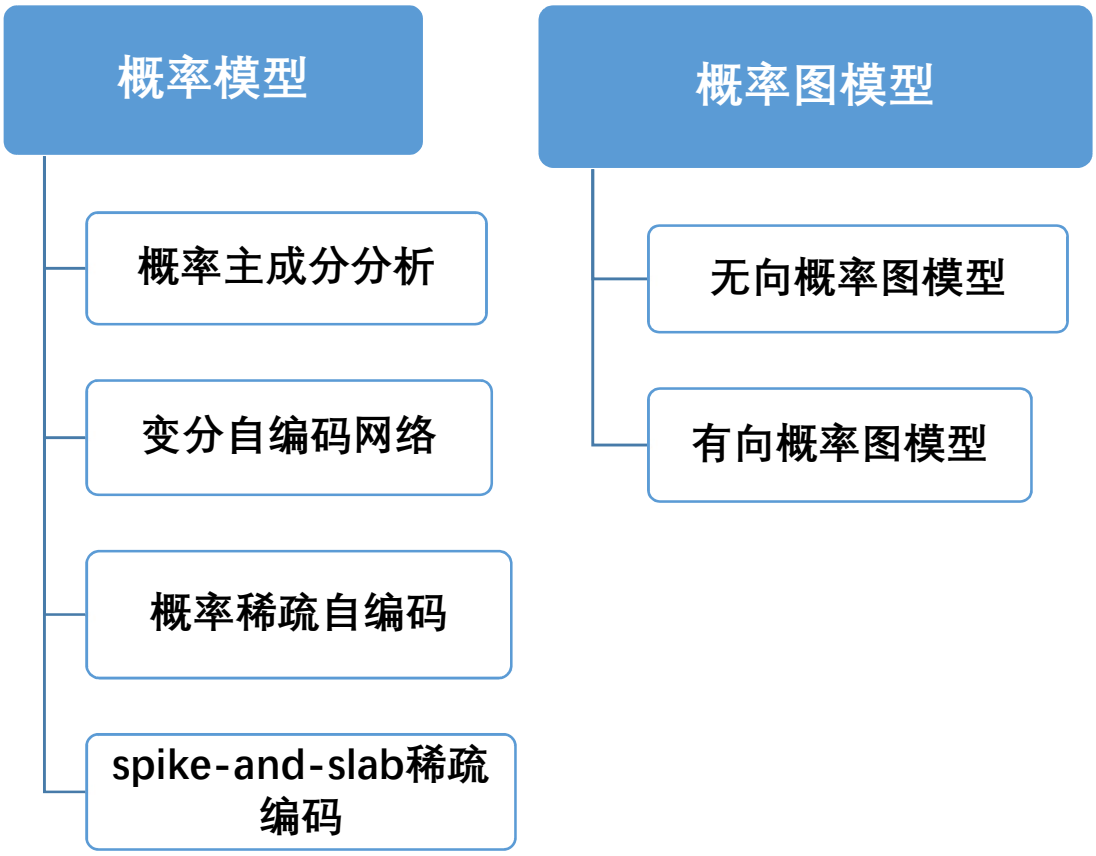
2.3 单层与多层数据再表达的概率学习模型



从概率建模的角度来看，数据再表达学习，即特征学习的问题可以解释为：用尽可能用少的隐随机变量的集合来试图恢复描述所有观察数据的分布。

我们用 $p(t,x)$ 表示隐随机变量 x 和观测数据（可见变量）t 的联合概率分布。提取的特征的值，即数据再表达，则被认为是推断过程的结果，以确定描述给定样本数据的隐随机变量的概率分布，即 $p(x|t)$，通常称之为后验概率。

学习就是对带正则化的训练数据进行最大似然估计，从而求出隐含随机变量的概率分布模型参数的取值。为了推断隐含随机变量的分布，概率模型又可以演化成概率图模型，而概率图模型又有两种模型，即有向图和无向图模型。由于它们在联合分布 $p(t,x)$ 的参数化方面不同，使得模型的推断性能、学习和推断过程中计算效率都有较大的不同。

概率图模型是包括数据再表达学习（表示学习）在内的机器学习的一个重要方法。

2.3.1 数据再表达学习的概率模型

数据再表达学习的概率模型学习的隐藏随机变量模型分别参数化条件概率 $p(x|t)$ 和先验概率 $p(x)$，以构造联合概率分布。这种数据再表达算法主要包括：

- **概率主成分分析（PCA）（Roweis[28]，Tipping 和 Bishop[29]）**
- **自编码的概率形式——变分自编码网络[27]**
- **概率稀疏自编码（Olshausen 和 Field[30]）**
- **spike-and-slab 稀疏编码（Goodfellow 等[32]）。**

下面就它们的单层模式和多层模式分别进行介绍。

一. 概率主成分分析（PPCA）

前面已经介绍了基本的 PCA 方法及其改进，它在数据处理领域是一种非常常见的方法。但是上述方法并没有建立在一个概率模型上。下面我们将介绍对该类方法的进一步改进：

- **如何通过对与隐变量模型中的参数做极大似然估计来确定一组观测数据向量的主轴。**
- **我们还将对期望最大化算法，即 EM 算法[28]的流程进行阐述，并且将之用于 PCA 算法，这样在空间和时间上的效率非常高，对于有缺失信息的样本有着较好的效果。**
- **最后简单介绍一种 PCA 的变体 SPCA（敏感主成分分析），它在数据空间中定义了更好的概率密度模型。**

对于 d 维可观测变量 t 和 q 维的隐藏变量 x，$q<d$，令 $t=Wx+\mu+\varepsilon$，其中 $d\times q$ 的矩阵 W 是这两组变量之间的一个连接矩阵（未知，需要确定），μ 是用以对 t 进行中心化，x 服从高斯分布 $N(0,I)$，误差项 ε 服从高斯分布 $N(0,\sigma^2 I)$，假设因子 x 与误差项 ε 是独立的。则当 x 固定时我们有 $t|x$ 服从高斯分布 $N(Wx+\mu,\sigma^2 I)$。

假设可观测数据 t 的样本为：$t_1,t_2,\cdots,t_N$，其中 N 是可观测数据 t 的样本个数。类似于上一节因子分析中的计算，我们得出可观测变量 t 也服从高斯分布 $N(\mu,C)$，其中

$$C=WW^T+\sigma^2 I$$

那么对应的极大似然函数为：

$$L=-\frac{N}{2}\{d\ln(2\pi)+\ln|C|+tr(C^{-1}R_t)\},\qquad (2.28)$$

$$R_t = \frac{1}{N}\sum_{n=1}^{N}(t_n - \mu e)(t_n - \mu e)^T \qquad (2.29)$$

其中矩阵 R_t 是观测样本的协方差矩阵，μ 是样本的均值。这里用到了矩阵迹的循环不变性。

由 $\frac{\partial L}{\partial W} = 0$，求解得

$$W_{ML} = U_q(\Lambda_q - \sigma^2 I)^{1/2} R \qquad (2.30)$$

其中 W 的下标是表示这是由极大似然估计解出来了的，有时也用 $\hat{W}$ 表示，$d \times q$ 的矩阵 U_q 是由 $d \times d$ 的矩阵 S 的主值特征向量构成，对角阵 Λ 是与之相对应的特征值构成的矩阵，R 是任意一个正交矩阵，可以取为单位矩阵。其它特征向量的组合（非主值特征向量）是似然函数的鞍点，这样便给出了一个从隐藏变量所在空间到可观测变量的一个投影。还可以解得误差项的方差

$$\sigma_{ML}^2 = \frac{1}{d-q}\sum_{j=q+1}^{d}\lambda_j$$

这是对投影过程中损失掉的那些维度里损失的值的一个平均。

我们将概率主成分分析与主成分分析做一个对比后发现，二者的映射关系其实只差了一个 $\sigma^2 I$，或者说**主成分分析是概率主成分分析中误差项方差为 0 的特殊情况**。在主成分分析中我们用 q 个主向量去近似的我们的数据，即把其余非主成分向量的数据看作噪音丢掉。$\sigma_{ML}^2 = \frac{1}{d-q}\sum_{j=q+1}^{d}\lambda_j$ 正好表达了这个观点， 即方差等于其它非主成分空间的方差的平均值，也就是把噪音平均分配到每个方向上。它可以直观给出观测数据在主成分空间上方差的组成成分，一方面来自噪音 σ^2，另一方面来自隐变量空间 $\sigma_0^2 = \lambda - \sigma^2$。 假设 u 是我们主成分空间的一个特征向量，那么该方向的方差可以表示为

$$\begin{aligned} u^T C u &= u^T(W^T W + \sigma^2 I)u \\ &= (u^T U_M (L_M - \sigma^2) U_M^T u) + \sigma^2 \\ &= (u\sum_{q+1}^{d}\lambda_q u_m u_m^T u - \sigma^2) + \sigma^2 = (\lambda - \sigma^2) + \sigma^2 = \lambda \end{aligned} \qquad (2.31)$$

上式最后一个等号表达了主成分分析的方差正好是由隐空间的方差 $\lambda - \sigma^2$ 和误差项（噪音）的方差 σ^2 两部分组成。

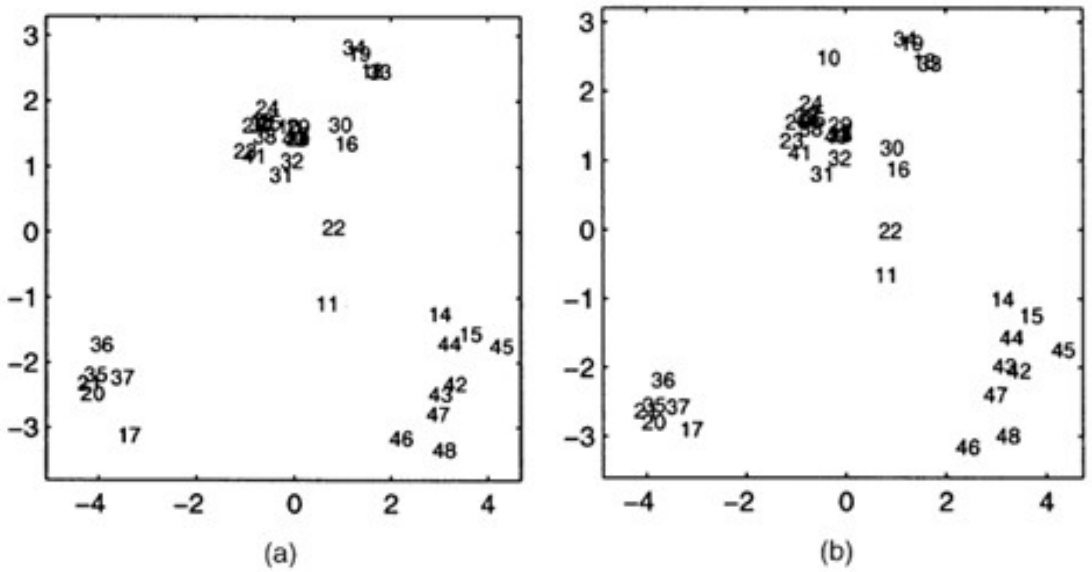


图 2.17 对一组烟草花叶数据完整数据应用 PCA 和有缺失的数据应用 PPCA

二. PCA 和 SPCA（sensible PCA 敏感主成分分析）中的 EM 算法

PCA 模型是对数据再表达中数据降维再表达的非常好的方法，但是存在有几个缺点：

（1）原始 PCA 找到主成分方向这个操作在高维数据或大量数据点中存在问题。例如当 n 和 p 为几百或几千时，考虑尝试在 p 维空间中对 n 个向量的样本协方差矩阵进行对角化。在计算复杂性和数据可达性方面都可能出现困难。

（2）PCA 标准方法的另一个缺点是如何适当的处理扔掉的数据这一点并不明显。事实上很多方法都存在这个问题，因此必须使用各种 ad-hoc 插值方法丢弃或完成不完整的点。然而 PCA 的 EM 算法保留了其他 EM 算法的所有好处，可以在每次迭代中直接估计丢失信息的最大似然值。

（3）PCA 模型本身存在一个严重的缺陷，它与用于计算其参数的技术无关：它没有在输入空间中定义适当的概率模型。如果我们对某些数据执行 PCA 然后询问模型是否适合新数据，则使用的唯一标准是新数据从其投影到主子空间的平方距离。远离训练数据但仍然在主子空间附近的数据点将被分配高“伪似然”或低误差。同样，不可能从 PCA 模型生成非常好的数据。

在本节中，我们介绍了一种称为敏感主成分分析（SPCA）的新模型，这是对 PC A 的一个改进算法，它在数据空间中定义了一个适当的协方差结构。它的参数也可以用 EM 算法学习。

下面首先对 EM 算法进行介绍。

EM 算法

假设你是一个餐厅的厨师，两位顾客点了同样一个菜，你就一次炒完。装盘时你不需要用天平一分为二，只需要先大体上分成两份，然后观察，从多的一份中匀出一些，放到少的一份中，再观察，再匀出一些，直到感觉一样多为止。这就是 EM 算法的基本思路。

EM 算法也称期望最大化（Expectation-Maximum,简称 EM）算法，它是一个基础算法，是很多机器学习领域算法的基础。

一个最直观了解 EM 算法思路的是 K-Means 算法。在 K-Means 聚类时，每个聚类簇的质心是隐含数据。我们会假设 K 个初始化质心，即 EM 算法的 E 步；然后计算得到每个样本最近的质心，并把样本聚类到最近的这个质心，即 EM 算法的 M 步。重复这个 E 步和 M 步，直到质心不再变化为止，这样就完成了 K-Means 聚类。K-Means 算法是比较简单的 EM 算法。

1. EM 算法要解决的问题

我们经常会从样本观察数据中，找出样本的模型参数。最常用的方法就是极大化模型分布的对数似然函数。

但是在一些情况下，我们得到的观察数据有未观察到的隐含数据，不知道隐含数据及其模型参数，因而无法直接用极大化对数似然函数得到模型分布的参数。这就是 EM 算法要解决的问题。

EM 算法要解决的问题是：在部分已知的相关变量 X 和部分“无法观测的潜在变量 Y ”的概率模型中，求取参数 θ 的最大似然估计 $\ell(\theta)$，使得变量 X 出现的可能性最大。

在开始状态下，不管是潜在变量 Y 还是参数 θ 都是未知的，并且因为存在潜在变量 Y ，直接最大化 $\ell(\theta)$ 求出 θ 比较困难。但是，我们能够明确的是：

- **知道 Y 的信息就可以得到 θ 的信息；**
- **知道 θ 的信息就可以得到 Y 的信息。**

EM 算法解决这个问题的思路是使用启发式的迭代方法。既然我们不能直接最大化 $\ell(\theta)$ 求出模型分布参数 θ ，那可以首先建立 $\ell(\theta)$ 的下界，即先猜想隐含数据（EM 算法的 E 步）；然后最大化下界，即基于观察数据和猜测的隐含数据一起来最大化对数似然函数，求解我们的模型参数（EM 算法的 M 步）。由于我们之前的隐藏数据是猜测的，所以此时得到的模型参数一般还不是我们想要的结果。我们基于当前得到的模型参数，继续猜测隐含数据（EM 算法的 E 步），然后继续最大化对数似然函数，求解我们的模型参数（EM 算法的 M 步）。以此类推，不断的迭代下去，直到模型分布参数基本无变化，算法收敛，找到合适的模型参数。

从上面的描述可以看出，EM 算法是迭代求解最大值的算法，同时算法在每一次迭代时分为两步，E 步和 M 步。逐步迭代更新隐含数据和模型分布参数，直到收敛，即得到我们需要的模型参数。

2. EM 算法的推导

给定样本集合 $Z=(X,Y)=\{(x_i,y_i):x_i\in R^n,y_i\in R^n(i=1,2,\cdots,m)\}$，包括 m 个独立的可观测数据 $\{x_i:x_i\in R^n(i=1,2,\cdots,m)\}$ 和 m 个无法可观测的数据 $\{y_i:y_i\in\Theta(i=1,2,\cdots,m)\}$，即潜在变量，$\Theta$ 为潜在变量所在的参数空间。$Z=(X,Y)$ 和 X 分别称为不完全数据集和完全数据集。

$x_i \in R^n$ 表示样本数据 $X = \{x_i : x_i \in R^n (i = 1, 2, \cdots, m)\}$ 的第 i 个数据，用 $x_{ij}(j = 1, 2, \cdots, n)$ 或者 $x_i^{(j)}(j = 1, 2, \cdots, n)$ 表示 i 个数据 $x_i \in R^n$ 的第 j 个分量，即：

$$x_i = (x_{i1}, x_{i2}, \cdots, x_{in})^T \quad \text{或者} \quad x = (x_i^{(1)}, x_i^{(2)}, \cdots, x_i^{(n)})^T$$

假设 $Z = (X, Y)$ 的联合概率密度函数为 $P(X, Y \mid \theta)$，其中 θ 为需要估计的参数。

对 θ 可以使用最大似然方法估计，从服从同一分布 $P(X, Y \mid \theta)$ 的概率模型中观测到数据集合 $\{x_i : x_i \in R^n (i = 1, 2, \cdots, m)\}$ 的似然函数为：

$$L(\theta) = L(x_1, x_2, \cdots, x_m) = \prod_{i=1}^{m} p(x_i \mid \theta) \qquad (2.32)$$

该似然函数反映了在不同的参数 θ 下，取得当前这个样本集的可能性。为了方便，定义对数似然函数 $\ell(\theta)$：

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{m} \log p(x_i \mid \theta) \qquad (2.33)$$

如果不存在潜在变量，对完全数据集 X 而言，直接可以得到最大似然估计：

$$\hat{\theta} = \arg\max \ell(\theta) = \arg\max \sum_{i=1}^{m} \log p(x_i \mid \theta) \qquad (2.34)$$

然而，完全数据集只是不完全数据集 $Z = (X, Y)$ 的可测部分，不完全数据的对数似然函数 $\ell(\theta)$ 的完整形式为：

$$\ell(\theta) = \sum_{i=1}^{m} \log \sum_{y_i \in \Theta} p(x_i, y_i \mid \theta) \qquad (2.35)$$

$\ell(\theta)$中不仅包含了可测变量，还包含了不可测变量

$$\{y_i : y_i \in \Theta(i=1,2,\cdots,m)\}$$

我们的目标是找到合适的参数θ和$\{y_i : y_i \in \Theta(i=1,2,\cdots,m)\}$，使得$\ell(\theta)$最大。所以，很难直接用最大似然方法对$\theta$进行估计。但是，如果我们知道了$\{y_i : y_i \in \Theta(i=1,2,\cdots,m)\}$的具体取值，仍然可以像一般的最大似然估计一样求解$\theta$。

显然，对于（2.35）式，即使知道了$y_1, y_2, \cdots, y_m$的具体取值，仍然难以通过对$\ell(\theta)$求导得到θ的最大似然估计量，因为（2.35）中包含一个和的对数$\log \sum_{y_i \in \Theta} p(x_i, y_i \mid \theta)$，求导非常复杂。通常**将“和的对数”变为“对数的和”，那么求导就简单了，这是 EM 算法的关键。**

定理 2.1：若X为离散随机变量，其分布满足：

$$P(X = x_i) = p_i, \sum_{i=1}^{m} p_i = 1$$

假设$Y = f(X)$是随机变量的连续函数，则

$$E(Y) = E\left[f(X)\right] = \sum_{i=1}^{m} f(x_i) p_i \qquad (2.36)$$

定理 2.2 Jensen 不等式：对于随机变量X，如果函数$Y = f(X)$是凸函数，则

$$E(Y) = E\left[f(X)\right] = \sum_{i=1}^{m} f(x_i) p_i \le f(E[x]) \qquad (2.37)$$

如果函数 $Y=f(X)$ 是凹函数，则

$$E(Y)=E\left[f(X)\right]=\sum_{i=1}^{m}f(x_i)p_i\geq f(E[x]) \qquad (2.38)$$

且等式 $E\left[f(X)\right]=f(E(X))$ 成立的充分必要条件为 X 为常数。

假设无法观测变量 $\{y_i:y_i\in\Theta(i=1,2,\cdots,m)\}$ 满足某种分布 Q_i $(i=1,2,\cdots,m)$，即 $Q_i(y_i)\geq 0,\sum_{y_i\in\Theta}Q_i(y_i)=1$。对于（2.35）式，由于 $\mathrm{y}=\log x$ 为凹函数，所以由 Jensen 不等式

$$\begin{aligned}\ell(\theta)&=\sum_{i=1}^{m}\log\sum_{y_i\in\Theta}p(x_i,y_i\mid\theta)=\sum_{i=1}^{m}\log\sum_{y_i\in\Theta}Q_i(y_i)\frac{p(x_i,y_i\mid\theta)}{Q_i(y_i)}\\&\geq\sum_{i=1}^{m}\sum_{y_i\in\Theta}Q_i(y_i)\log\frac{p(x_i,y_i\mid\theta)}{Q_i(y_i)}\end{aligned} \qquad (2.39)$$

其中

$$\log\sum_{y_i\in\Theta}Q_i(y_i)\frac{p(x_i,y_i\mid\theta)}{Q_i(y)}$$

就是

$$f\left(E\left[\frac{p(x_i,y_i\mid\theta)}{Q_i(y_i)}\right]\right),$$

而

$$\sum_{y_i\in\Theta}Q_i(y_i)\log\frac{p(x_i,y_i\mid\theta)}{Q_i(y)}$$

就是

$$E\left[f\left(\frac{p(x_i, y_i \mid \theta)}{Q_i(y_i)} \right) \right]$$

虽然（2.39）式右端变成了“对数的和”，这使得求导数变得简单了。但是，这里是不等式，当无法观测数据$\{y_i : y_i \in \Theta(i=1,2,\cdots,m)\}$给定式，对于不等式右端

$$\sum_{i=1}^{m} \sum_{y_i \in \Theta} Q_i(y_i) \log \frac{p(x_i, y_i \mid \theta)}{Q_i(y_i)} \qquad (2.40)$$

求最大似然估计，并不等价于得到$\ell(\theta)$的最大似然估计。但是，根据（2.39）式，

$$J(P,\theta) = \sum_{i=1}^{m} \sum_{y_i \in \Theta} Q_i(y_i) \log \frac{p(x_i, y_i \mid \theta)}{Q_i(y_i)}$$

是$\ell(\theta)$的一个下界。我们可以不断地最大化这个下界，使得$\ell(\theta)$的值不断提高，最终趋近于最大值，这就是 EM 算法的整个思路。

3. EM 算法中未知分布$Q_i(y_i)$的选取方法

假定θ已经给定，根据 Jensen 不等式，要让（2.39）式变成等式，需要让随机变量$\frac{p(x_i, y_i \mid \theta)}{Q_i(y_i)}$变成常数值（记为$c$），即

$$\frac{p(x_i, y_i \mid \theta)}{Q_i(y_i)} = c(i=1,2,\cdots,m) \qquad (2.41)$$

又因为$Q_i(y_i) \geq 0, \sum_{y_i \in \Theta} Q_i(y_i) = 1$，所以

$$\sum_{y_i \in \Theta} p(x_i, y_i \mid \theta) = c \qquad (2.42)$$

所以

$$Q_i(y_i)=\frac{p(x_i,y_i\mid\theta)}{c}=\frac{p(x_i,y_i\mid\theta)}{\sum_{y_i\in\Theta}p(x_i,y_i\mid\theta)}$$
$$=\frac{p(x_i,y_i\mid\theta)}{p(x_i\mid\theta)}=p(y_i\mid x_i,\theta)(i=1,2,\cdots,m) \quad (2.43)$$

即参数θ给定的情况下，$Q_i(y_i)$的最优选择就是后验概率$p(y_i\mid x_i,\theta)$，它直接给定了$\ell(\theta)$的下界$J(P,\theta)$。在给定$Q_i(y_i)$之后，我们再通过最大化下界（θ为变量，对参数θ求解最大似然估计$\hat{\theta}$），进一步提高似然函数的值。

综上所述，EM 算法在下述 E 步和 M 步之间循环重复，直至收敛：

E 步：对于每一个$i=1,2,\cdots,m$，计算：

$$Q_i(y_i)=p(y_i\mid x_i,\theta)(i=1,2,\cdots,m)$$

M 步：计算最大似然估计：

$$\hat{\theta}=\arg\max\sum_{i=1}^{m}\sum_{y_i\in\Theta}Q_i(y_i)\log\frac{p(x_i,y_i\mid\theta)}{Q_i(y_i)} \quad (2.44)$$

更新参数$\theta:=\hat{\theta}$。

4. EM 算法流程

现在我们总结下 EM 算法的流程。

（1） 初始化未知参数；

（2） 迭代下述过程直至收敛：

E 步（Expectation）：计算期望，即利用对未知参数的现有估计值，通过计算当前的最大似然期望值，得到潜在变量的后验概率；

M 步（Maximization）:最大化，即宠幸估计未知参数，使得在 E 步求得数据的似然性最大，给出未知参数的期望估计。

下面给出一般的 EM 算法的步骤如下：

算法：EM 算法

输入：观察数据$X=(x_1,x_2,\cdots,x_m)$，潜在变量$Y=(y_1,y_2,\cdots,y_m)$所在的参数空

间Θ，未知参数θ的随机初始化的初值θ^0，最大迭代次数K，终止条件T；

for k=1:K

step1（E 步）对于每一个$i=1,2,\cdots,m$，，$y_i \in \Theta$计算：

$$Q_i^k(y_i) = p(y_i \mid x_i, \theta^{k-1}) \qquad (2.45)$$

step2（M 步）计算最大似然估计：

$$\theta^k = \arg\max_{\theta} \sum_{i=1}^{m} \sum_{y_i \in \Theta} Q_i^k(y_i) \log\left(\frac{p(x_i, y_i; \theta)}{Q_i^k(y_i)} \right) \qquad (2.46)$$

if $\frac{\ell(\theta^k) - \ell(\theta^{k-1})}{\ell(\theta^{k-1})} \le T$，则算法终止

输出：最大似然估计值$\hat{\theta} = \theta^k$

4. EM 算法的收敛性

EM 算法的流程并不复杂，但是还有两个问题需要我们思考：

1） EM 算法能保证收敛吗？

2） EM 算法如果收敛，那么能保证收敛到全局最大值吗？

首先我们来看第一个问题, EM 算法的收敛性。要证明 EM 算法收敛，则我们需要证明我们的对数似然函数的值在迭代的过程中一直在增大。即：

假设 EM 算法在第k次迭代后得到的参数值为θ^k，接着进行第$k+1$次迭代，第 E 步得到：

$$Q_i^{k+1}(y_i) = p(y_i \mid x_i, \theta^k) \qquad (2.47)$$

在给定θ^k时，$Q_i^{k+1}(y_i)$使得 Jensen 不等式中的等式成立：

$$\ell(\theta^k) = \sum_{i=1}^{m} \sum_{y_i \in \Theta} Q_i^{k+1}(y_i) \log\left(\frac{p(x_i, y_i \mid \theta^k)}{Q_i^{k+1}(y_i)} \right) \qquad (2.48)$$

即在 E 步利用未知参数的现有估计值θ^k计算$Q_i^{k+1}(y_i)$，得到当前能够达到的最大似然估计值$\ell(\theta^k)$，即下界$J(P, \theta^k)$。然后执行第$k+1$次迭代的 M 步，固定$Q_i^{k+1}(y_i)$，将（2.48）中的θ^k作为变量，通过最大似然估计，对$\ell(\theta^k)$求导，得到更新参数θ^{k+1}。因此有：

$$\sum_{i=1}^{m}\sum_{y_i\in\Theta} Q_i^{k+1}(y_i)\log\left(\frac{p(x_i,y_i\mid\theta^{k+1})}{Q_i^{k+1}(y_i)}\right)\geq\sum_{i=1}^{m}\sum_{y_i\in\Theta} Q_i^{k+1}(y_i)\log\left(\frac{p(x_i,y_i\mid\theta^{k})}{Q_i^{k+1}(y_i)}\right)$$

(2.49)

再利用（2.39），有

$$\ell(\theta^{k+1})\geq\sum_{i=1}^{m}\sum_{y_i\in\Theta} Q_i^{k+1}(y_i)\log\left(\frac{p(x_i,y_i\mid\theta^{k+1})}{Q_i^{k+1}(y_i)}\right)\qquad(2.50)$$

由（2.48）、（2.49）和（2.50），有：

$$\ell(\theta^{k+1})\geq\ell(\theta^{k})\qquad(2.51)$$

因此，通过交替使用 E 步和 M 步，EM 算法逐步改善模型参数，使得参数和相关的随机变量 $X=(x_1,x_2,\cdots,x_m)$ 的似然概率 $\ell(\theta)$ 随着迭代步数增加而单调增加，最终 θ^k 趋近于他的最大似然估计 $\hat{\theta}$。

从上面的推导可以看出，EM 算法可以保证收敛到一个稳定点，但是却不能保证收敛到全局的极大值点，因此它是局部最优的算法，当然，如果我们的优化目标 $\ell(\theta)$ 是凸的，则 EM 算法可以保证收敛到全局最大值，这点和梯度下降法这样的迭代算法相同。至此我们也回答了上面提到的第二个问题。

5. EM 算法的一些思考

如果我们从算法思想的角度来思考 EM 算法，我们可以发现我们的算法里已知的是观察数据，未知的是隐含数据和模型参数，在 E 步，我们所做的事情是固定模型参数的值，优化隐含数据的分布，而在 M 步，我们所做的事情是固定隐含数据分布，优化模型参数的值。比较其它的机器学习算法，其实很多算法都有类似的思想。比如支持向量机的 SMO 算法、坐标轴下降法(Lasso 回归算法)，都使用了类似的思想来求解问题。

EM 算法的迭代过程可以通过图 2.18 说明。图中上方黑粗实线曲线表示待求解的最大似然函数 $\ell(\theta)$，其极大似然估计值为 $\hat{\theta}$，$\ell(\theta)$ 的下界 $J(P^k,\theta)$ 下界为图中最下方的虚线曲线，此时 $\ell(\theta^k)\geq J(P^k,\theta^k)$，即 Jensen 不等式中的等号成立。EM 算法的第 $k+1$ 步迭代的 E 步固定 θ^k，调整潜在变量的分布函数，使得 $Q_i^{k+1}(y_i)=p(y_i\mid x_i,\theta^k)$，将下方的虚线曲线上的 A 点 $J(P^k,\theta^k)$ 上移与上方黑粗实线曲线 $\ell(\theta)$ 在 B 点 $J(P^{k+1},\theta)$ 相切，使得下界 $J(P,\theta)$ 上升至与 $\ell(\theta^k)$ 相等。由于 B 点不是 $\ell(\theta)$ 极大值点，EM 算法的第 $k+1$ 步迭代的 M 步，固定分布函数

$Q_i^{k+1}(y_i) = p(y_i \mid x_i, \theta^k)$，调整参数$\theta$，使得下界$J(P^{k+1}, \theta)$在 C 点$J(P^{k+1}, \theta^{k+1})$达到最大值，得到参数的新的估计值$\theta^{k+1}$。再固定$\theta = \theta^{k+1}$，调整潜在变量的分布函数（上面虚线曲线）……，直到收敛似然函数$\ell(\theta)$的极大值点$\theta = \hat{\theta}$。

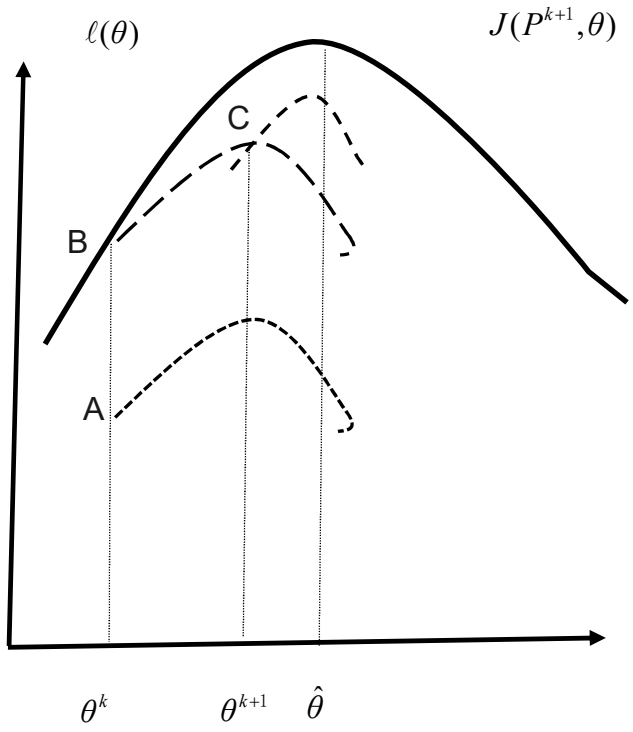


图 2.18 EM 算法说明

6. EM 算法的在高斯混合模型中的应用

高斯混合模型（Gaussian Mixture Model, GMM）解决的问题：一组样本数据$\{x_i : x_i \in R^n (i = 1, 2, \cdots, m)\}$服从高斯分布，即数据由$\rho$个满足正态分布$N(\mu_j, \sigma_j), j = 1, 2, \cdots, \rho$的不同概率模型混合而成，却不知道数据$x_i (i = 1, 2, \cdots, m)$具体来自哪个高斯分布模型。它更可能是以一定的概率$q_i (q_i \geq 0, \sum_{i=1}^{\rho} q_i = 1)$来自于第$j (j = 1, 2, \cdots, \rho)$个模型，也不知道每个高斯模

型的具体参数$(\mu_j,\sigma_j), j=1,2,\cdots,\rho$，希望能够根据这些观测到的样本数据$\{x_i : x_i \in R^n (i=1,2,\cdots,m)\}$估计出这些参数：

$$\theta=(q,\mu,\sigma)=\left\{(q_j,\mu_j,\sigma_j), j=1,2,\cdots,\rho\right\}$$

样本数据$\{x_i : x_i \in R^n (i=1,2,\cdots,m)\}$是可观测的，而它们来自哪个高斯分布模型是不知道的，用潜在变量y_i表示$x_i (i=1,2,\cdots,m)$所在的模型的参数，则有$p(y_i=j|\theta)=q_j$。为了估计高斯混合模型的参数，写出其对数似然函数：

$$\ell(\theta)=\sum_{i=1}^{m}\log p(x_i|\theta)=\sum_{i=1}^{m}\log\sum_{j=1}^{\rho}\log p(x_i,y_j|\theta) \qquad (2.52$$

可以看出对高斯混合模型的未知参数θ的求解可以用 EM 算法进行。

7. PCA 中的 EM 算法

我们直接写出生成模型可观测变量$y=Cx+v$，其中隐藏变量$x\sim N(0,I)$，噪音$v\sim N(0,R)$，R是一个对角矩阵，并且x和v是独立的。这时候我们通过分布函数的定义以及贝叶斯公式等可以很容易的推出下列结论

$$y\sim N(0,CC^T+R), \qquad (2.53)$$

$$P(x|y)=\frac{P(y|x)P(x)}{P(y)}=\frac{N(Cx,R)|_y\ N(0,I)|_x}{N(0,CC^T+R)|_y}$$

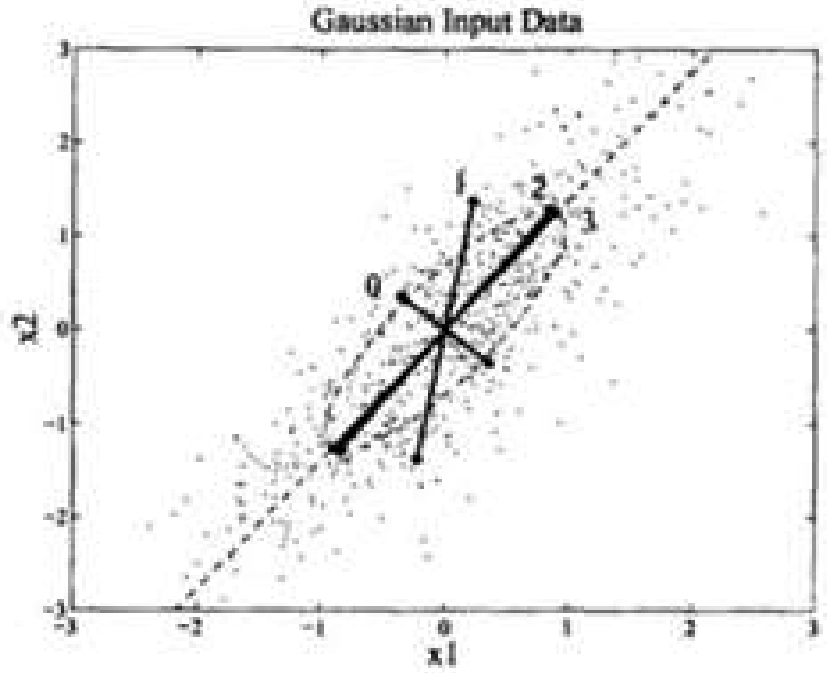
$$P(x|y)=N(\beta y,I-\beta C)|_x,\quad \beta=C^T(CC^T+R)^{-1} \qquad (2.54)$$

这时我们已经完成了使用 EM 算法的准备工作，然后我们只需将之带入即可得到 PCA 的 EM 算法如下：

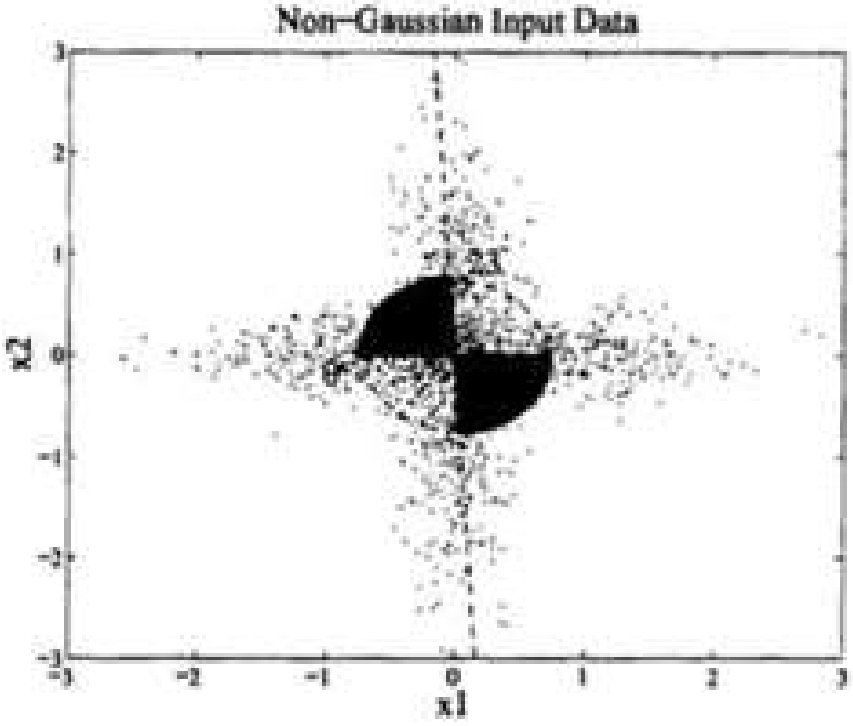
E步骤：　　$X=(C^TC)^{-1}C^TY$

M步骤：　　$C^{new}=YX^T(XX^T)^{-1}$

其中Y是可观测样本数据所构成的矩阵，X是隐藏变量所构成的矩阵，C的列向量可以张成前k个主成分空间（为了计算相应的特征值和特征向量，可观测变量能够投影到k维子空间中，并建立子空间中协方差的有序正交基，在 PCA 方法中已经介绍）。算法的工作原理在下图中以图形的方式给出。



(a)



(b)

图 2.18 算法迭代的示例

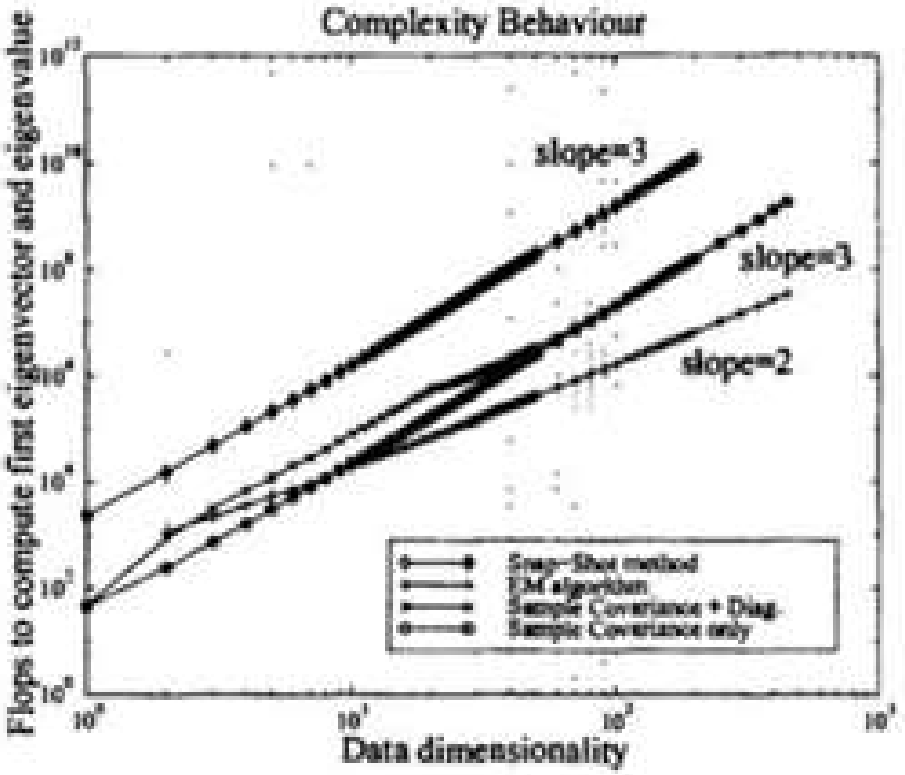
图 2.18 (a) 图显示了从高斯分布中绘制的数据的第一主成分的学习，而 (b) 图显示了对来自非高斯分布的数据的学习。虚线表示样本协方差的前导特征向量的方向。虚线椭圆是样本协方差的一个标准偏差边界。算法的进度由实线表示，这是对特征向量的猜测，其长度表示每次迭代时特征值的猜测。数字是表示迭代是第几次；0 是初始条件。请注意，(b)图的学习

过程不会陷入局部最小值，虽然它需要超过 20 次迭代才能收敛，这对于高斯数据来说是不常见的（见图 2.18）。

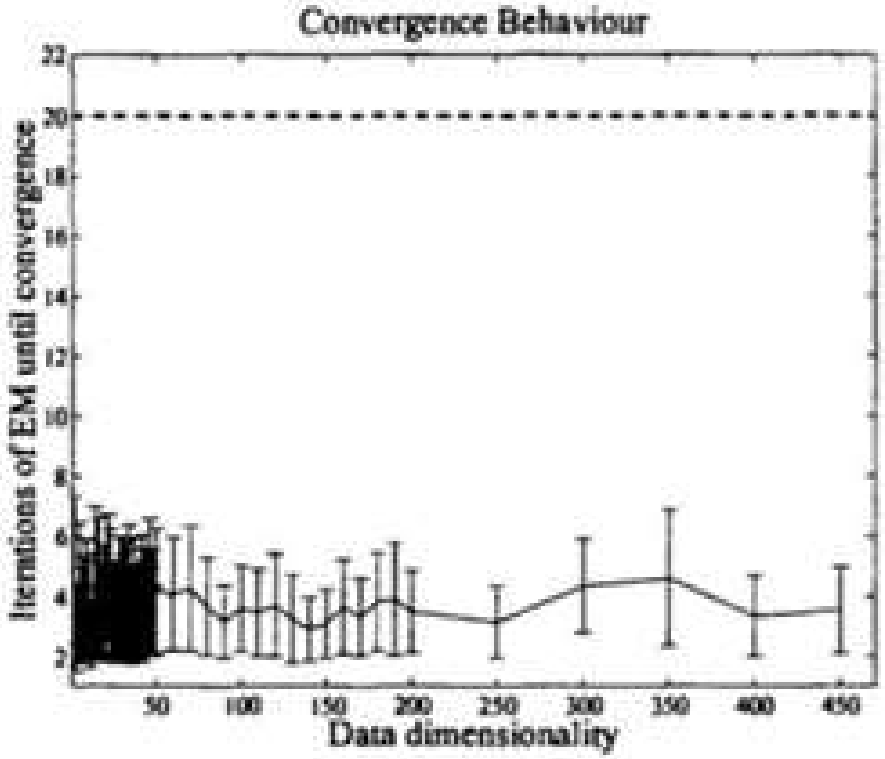
我们可给出算法背后的直观解释。猜测主子空间的方向。修复猜测的子空间并将数据 y 投影到其中以给出隐藏状态 x 的值。现在修正隐藏状态的值并选择子空间方向，以最小化数据点的平方重建误差。对于上面简单的二维示例，我可以给出一个物理类比。想象一下，我们有一个固定在原点的杆，可以自由旋转。选择杆的方向。保持杆保持不动，将每个数据点投射到杆上，并用弹簧将每个投影点连接到其原始点。现在释放杆。重复。棒的方向代表我们对数据集主要成分的猜测。存储在弹簧中的能量是我们试图最小化的重建误差。

PCA 的 EM 学习算法相当于用于找到由前 k 个特征向量张成的子空间的迭代过程，无需显式计算样本协方差。它对于比较小的 k 是好用的，因为它每次操作的复杂度是 $O(npk)$，因此仅线性地取决于数据的维数和点的数量。计算样本协方差矩阵的方法复杂度小于 $O(np^2)$，而形成数据线性组合的快照方法等方法必须计算和对角化点之间所有可能内积的矩阵，其计算复杂度为 $O(n^2 p)$。与这些方法相比，算法的复杂度缩放如下面的图 2.19 所示。对于每个维度，生成随机协方差矩阵 $\sum$，然后从 $N(0,\sum)$ 绘制 $10p$ 个点。使用 MATLAB 的 flop 函数记录查找第一主成分所需的离线点操作次数。正如预期的那样，在 $k < p$ 和 n 都很大的情况下，EM 算法更有效。如果 $k \approx p \approx n$（我们想要所有的特征向量）那么所有方法都是 $O(p^3)$ 的。

EM 算法的标准收敛性证明也适用于该算法，因此我们可以确保它总是达到局部最大似然。 进一步的，Tipping 和 Bishop[29]已经证明唯一稳定的局部极值是找到真正主子空间的全局最大值；所以它收敛到正确的结果。另一个可能的问题是收敛所需的迭代次数可以用 p 或 n 来缩放。为了研究这个问题，人们已经明确地计算了合成数据集的领先特征向量（如上所述，$n=10p$）的变化维度。最多 450 个维度（4500 个数据点），迭代次数大致保持不变，平均值为 3.6。第一个 k 特征值的比率似乎是控制直到收敛的迭代次数的关键参数（例如，在图 2.19b 中，该比率为 1.0001。）



(a)



(b)

图 2.19 算法的时间复杂度和收敛性

在所有情况下，数据点的数量 n 是维数 p 的 10 倍。对于图 2.19(a) 图，

记录了用于找到前几个特征向量和特征值的浮点运算的数量。 EM 算法总是运行 20 次迭代。 使用 MATLAB 函数 cov 和 eigs 来展示示例中协方差矩阵对角化的成本。 snap-shot 方法显示仅指示缩放； 当 $n > p$ 时，通常不会使用它。 在 (b) 图中，通过显式计算前几个特征向量然后运行 EM 算法来研究收敛，直到其猜测和真实本征方向的点积为 0.999 或更高。 误差条显示多次运行的±一个标准偏差。 虚线表示用于在左侧面板中生成 EM 算法曲线（'+'）的迭代次数。

现在简单介绍一下 SPCA，当噪音的协方差矩阵 $R = \varepsilon I$ 时（也就是说椭球变成了球），我们将在 R 的对角线上将标量值称为全局噪声级别。注意到 SPCA 用了 $1 + pk - k(k-1)/2$ 个自由参数来模拟协方差。需要注意的一点是，即使 SPCA 发现的主要成分与 PCA 相同，后验的平均值通常与 PCA 投影给出的点不同。学习 SPCA 也使用 EM 算法（如下所示）。

由于它具有无限的噪声水平 ε，SPCA 在数据空间中定义了适当的生成模型和概率分布：

$$y \sim N(0, CC^T + \varepsilon I) \qquad (2.55)$$

用于学习 SPCA 模型的 EM 算法如下：

E步骤： $\beta = C^T(CC^T + \varepsilon I)^{-1} \quad \mu_x = \beta Y \quad \Sigma_x = nI - n\beta C + \mu_x \mu_x^T$

M步骤： $C^{new} = Y\mu_x^T \Sigma^{-1} \qquad \varepsilon^{new} = trace[XX^T - C\mu_x Y^T]/n^2$

注意到利用两点可以说明 SPCA 也具有受到 $O(knp)$ 限制的复杂性并且不会更糟。一方面

$$(CC^T + \varepsilon I)^{-1} = (I/\varepsilon - C(I + C^T C/\varepsilon)C^T/\varepsilon^2), \qquad (2.56)$$

另一方面在 M 步骤中，我们只需要计算矩阵的迹，不用计算整个协方差矩阵。

三. 单层与多层变分自编码网络

变分自编码网络[27]可谓是近些年来深度学习领域中研究的热点之一。和其它的自编码网络一样，也是一种无监督的前端学习算法。

在机器学习中，若想估计一个数据分布中的参数，常用的方法是最大似然估计，也就是要找到一个参数 θ 使得关于训练数据的似然函数值最大。现假设数据集是 $x_1, x_2, ..., x_m$，则似然函数为

$$\log p_\theta(x_1, ..., x_m) = \sum_{i=1}^{m} \log p_\theta(x_i) \qquad (2.57)$$

而

$$
\begin{aligned}
&\log p_\theta(x_i) = \sum_z q_\phi(z|x_i)\log\left(p_\theta(x_i)\right) \\
&= \sum_z q_\phi(z|x_i)\log\frac{p_\theta(z,x_i)}{p_\theta(z|x_i)} \\
&= \sum_z q_\phi(z|x_i)\log\frac{p_\theta(z,x_i)}{q_\phi(z|x_i)} + \sum_z q_\phi(z|x_i)\log\frac{q_\phi(z|x_i)}{p_\theta(z|x_i)} \\
&= D_{KL}\left(q_\phi(z|x_i)\middle\|p_\theta(z|x_i)\right) + L(\theta,\phi,x_i) \\
&\geq L(\theta,\phi,x_i)
\end{aligned}
\tag{2.58}
$$

$$
\begin{aligned}
&L(\theta,\phi,x_i) \\
&= \sum_z q_\phi(z|x_i)\log\frac{p_\theta(z)\,p_\theta(x_i|z)}{q_\phi(z|x_i)} \\
&= -D_{KL}\left(q(z|x_i)\middle\|p(z)\right) + E_{q_\phi(z|x_i)}\log p(x_i|z) \\
&= E_{q_\phi(z|x_i)}\left[-\log q_\phi(z|x_i) + \log p_\theta(z,x_i)\right]
\end{aligned}
\tag{2.59}
$$

其中，$q_\phi(z|x)$是真实后验分布$p_\theta(z|x)$的近似，z是一个隐变量（导致数据生成的观测不到的变量）。

但是，随之而来的一个问题是下界$L(\theta,\phi,x_i)$关于ϕ的梯度的计算。用传统的蒙特卡洛梯度估计算子在实际中的效果往往不好。因此就有另外一种梯度估计的方法，即 SGVB 方法以及优化下界$L(\theta,\phi,x_i)$的算法 AEVB。

SGVB 算子中主要用到了重参数化技巧。对于近似的后验概率$q_\phi(z|x)$，随机变量$\tilde{z} \sim q_\phi(z|x)$，它可以用一个可导的变换$g_\phi(\varepsilon,x)$使得：

$$
\tilde{z} = g_\phi(\varepsilon,x),\ \ \varepsilon \sim p(\varepsilon) \tag{2.60}
$$

这时某个函数$f(z)$关于$q_\phi(z|x)$的期望的蒙特卡洛估计为：

$$
E_{q_\phi(z|x_i)}\left[f(z)\right] = E_{p(\varepsilon)}\left[f\left(g_\phi(\varepsilon,x_i)\right)\right] \simeq \frac{1}{L}\sum_{l=1}^{L} f\left(g_\phi(\varepsilon_l,x_i)\right),
$$

$$
\varepsilon_l \sim p(\varepsilon) \tag{2.61}
$$

因此，下界 $L\left(\theta,\phi,x_i\right)$ 的 SGVB 估计 $\tilde{L}_A\left(\theta,\phi,x_i\right)$ 为：

$$\tilde{L}_A\left(\theta,\phi,x_i\right)=\frac{1}{L}\sum_{l=1}^{L}\log p_\theta\left(x_i,z_{i,l}\right)-\log q_\phi\left(z_{i,l}\left|x_i\right.\right) \quad (2.62)$$

或者

$$\tilde{L}_B\left(\theta,\phi,x_i\right)=-D_{KL}\left(q_\phi\left(z\left|x_i\right.\right)\left\|p_\theta\left(z\right)\right.\right)+\frac{1}{L}\sum_{l=1}^{L}\log p_\theta\left(x_i\left|z_{i,l}\right.\right) \quad (2.63)$$

其中，

$$z_{i,l}=g_\phi\left(\varepsilon_{i,l},x_i\right),\ \ \varepsilon_l\sim p\left(\varepsilon\right) \quad (2.64)$$

算法 2.3（AEVB 算法）：

θ，$\varphi\leftarrow$初始化参数

repeat

$X_M\leftarrow$ Random mini-batch of M data points (drawn from full dataset)

$\varepsilon\leftarrow$ Random samples from noise distribution $p(\varepsilon)$

$g\leftarrow\nabla_{\theta,\varphi}\tilde{L}_M(\theta,\varphi;X_M,\varepsilon)$ (Gradients of mini-batch estimator)

$\theta,\varphi\leftarrow$ Update parameters using gradients g (e.g. SGD or Adagrad)

until convergence of parameters (θ,φ)

return (θ,φ)

其中，

$$L\left(\theta,\phi;X\right)\approx\tilde{L}_M\left(\theta,\phi;X_M\right)=\frac{N}{M}\sum_{i=1}^{M}\tilde{L}\left(\theta,\phi;X_i\right),\ X^M=\left\{x_1,x_2,\cdots,x_M\right\} \quad (2.65)$$

由下式

$$\tilde{L}\left(\theta,\phi,x_i\right)=-D_{KL}\left(q_\phi\left(z\left|x_i\right.\right)\left\|p_\theta\left(z\right)\right.\right)+\frac{1}{L}\sum_{l=1}^{L}\log p_\theta\left(x_i\left|z_{i,l}\right.\right) \quad (2.66)$$

可以很清晰地看出和变分自编码网络的联系：
第一项（先验概率和近似后验概率之间的 KL 散度）这一项看作是一个正则项；第二项是一个负的期望重构误差项。$g_\phi(\cdot)$ 将数据 $x^{(i)}$ 和随机噪声向量 $\varepsilon^{(l)}$ 映射到从近似后验分布中采样的

$$z_{i,l}=g_\phi\left(\varepsilon_l,x_i\right),\ z_{i,l}\sim q_\phi\left(z\middle|x_i\right),$$

然后样本 $z_{i,l}$ 作为函数 $\log p_\theta\left(x_i\middle|z_{i,l}\right)$ 的输入，这里 $\log p_\theta\left(x_i\middle|z_{i,l}\right)$ 等于给定 $z_{i,l}$ 的生成模型中 x_i 的对数概率密度。

下面正式给出变分自编码网络的描述：

令隐变量的先验概率是中心化的各项同性的多元高斯函数：

$$p_\theta\left(z\right)=N\left(z;0,I\right),\ p_\theta\left(x\middle|z\right)$$

它是一个多元的高斯分布或者是伯努利分布，其中分布中的参数可以通过全连接的单层（只有一个隐藏层）神经网络来计算。

（1）伯努利分布作为解码网络

$$\log p\left(x\middle|z\right)=\sum_{i=1}^{D}x^{(i)}\log y^{(i)}+\left(1-x^{(i)}\right)\cdot\log\left(1-y^{(i)}\right)\qquad(2.67)$$

其中

$$y=f_\sigma\left(W_2\tanh\left(W_1z+b_1\right)+b_2\right),\qquad(2.68)$$

f_σ 是一个逐元素的 sigmoid 函数，$\theta=\{W_1,W_2,b_1,b_2\}$ 是神经网络的权重和偏置。

（2）高斯分布作为编码网络和解码网络

$$\log\left(x\middle|z\right)=\log N\left(x;\mu,\sigma^2I\right)\qquad(2.69)$$

其中，

$$\mu=W_4h+b_4,\ \log\sigma^2=W_5h+b_5,\qquad(2.70)$$

$$h=\tanh\left(W_3z+b_3\right),\ \theta=\left\{W_3,W_4,W_5,b_3,b_4,b_5\right\}\qquad(2.71)$$

是神经网络的权重和偏置，如果当这个网络作为编码网络 $q_\phi\left(z\middle|x\right)$ 时，z 和 x 交换，参数 ϕ 则是神经网络的权重和偏置。

假设 $p_\theta\left(z\middle|x\right)$ 的近似 $q_\phi\left(z\middle|x\right)$ 的分布也是一个多元高斯函数（方差矩阵是对角矩阵），即：

$$\log q_\phi\left(z\middle|x_i\right)=\log N\left(z;\mu_i,\sigma_i^2I\right)\qquad(2.72)$$

其中 μ_i 和 σ_i 是编码网络的输出，这里编码网络是数据 x_i 和参数 ϕ 的函数。利用重参数化技巧

$$z_{i,l}=g_{\phi}\left(x_i,\varepsilon_l\right)=\mu_i+\sigma_i\odot\varepsilon_l,\ \ \varepsilon_l\sim N\left(0,I\right),\quad (2.73)$$

其中$\odot$是一个逐元素的积。由于$p_{\theta}(z)$和$q_{\phi}\left(z|x\right)$都是高斯函数，很容易计算出它们之间的 KL 散度为：

$$-\frac{1}{2}\sum_{j=1}^{J}\left(1+\log\sigma_j^2-\mu_j^2-\sigma_j^2\right),\quad (2.74)$$

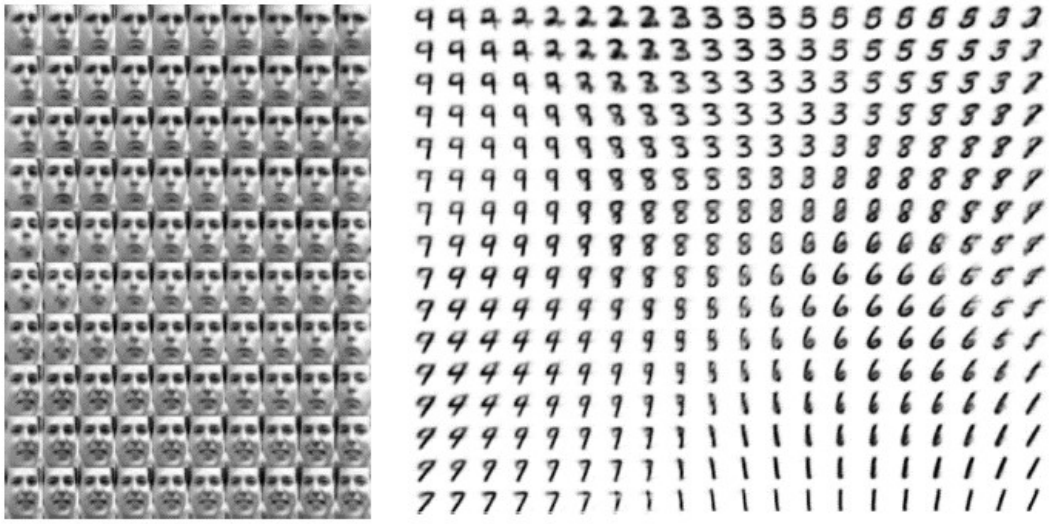
因而对于数据$x^{(i)}$的损失函数为：

$$L\left(\theta,\phi;x_i\right)\simeq\frac{1}{2}\sum_{j=1}^{J}\left(1+\log\left(\sigma_i^{(j)}\right)^2-\left(\mu_i^{(j)}\right)^2-\left(\sigma_i^{(j)}\right)^2\right)+\frac{1}{L}\sum_{l=1}^{L}\log p_{\theta}\left(x_i\left|z_{i,l}\right.\right)\quad (2.75)$$

其中$z_{i,l}=\mu_i+\sigma_i\odot\varepsilon_l,\ \ \varepsilon_l\sim N\left(0,I\right)$，至于解码网络中$\log p_{\theta}\left(x_i\left|z_{i,l}\right.\right)$是高斯分布还是伯努利分布，取决于模型中数据的类型。

下面给出实验效果（生成模型学到的数据流形的可视化）：

（1）二维的隐藏层空间，单位正方形上线性排列的坐标通过高斯分布函数的反函数计算出隐变量z的值，然后通过解码网络生成数据。图 2.20 中分别是学到的人脸流形和 MNIST 流形。



(a) Learned Frey manifold　　(b) Learned MNIST manifold

图 2.20 带有二维隐藏层变量的生成模型，用 AEVB 算法学习的数据流形的可视化效果。

由于隐藏层空间的先验分布是高斯的，因此线性空间中坐标在单位正方形上的点，通过高斯的逆 CDF 变换以产生隐藏层的变量值z. 对每一个这样的值z，绘制相应的带参数θ的生成值$p_{\theta}(x|z)$。

（2）图 2.21 是在不同维度的隐藏层空间中采样得到的结果

(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

图 2.21: MNIST 学习生成模型中对隐藏层空间不同维度的随机抽样结果

四. Spike-and-slab 稀疏编码[32]

Spike-and-Slab Sparse Coding（简称 S3C）[32]与 ssRBM 类似，在隐含层有二元 spike **变量** $h \in \{0,1\}^m$ 和实值 slab 变量 $s \in \mathbb{R}^m$，其中 spike 变量来控制稀疏性，输入为实值向量 $v \in \mathbb{R}^n$，则它们服从分布：

$$p(h_i = 1) = \sigma(b_i)$$

$$p(s_i \mid h_i) = N(s_i \mid h_i \mu_i, \alpha_{ii}^{-1})$$

$$p(v_d \mid s, h) = N(v_d \mid W_d(h \circ s), \beta_{dd}^{-1})$$

其中 σ 表示 sigmoid 函数 $\sigma(x) = \frac{1}{1+e^{-x}}$。$b$ 是 spike 变量的偏差(biases)，μ 和 W 分别控制 s 对 h 以及 v 对 s 的线性依赖。α 和 β 是各自条件变量的对角精度矩阵(precision matrix)，$h \circ s$ 表示 h 和 s 各自元素间（element-wise）的乘积。它与一般稀疏编码的区别是稀疏编码是给稀疏特征变量一个 Cauchy 或者 Laplace 的先验分布，而 S3C 模型则是给出一个 spike-and-slab 先验分布。S3C 能够有效的用于特征提取，[32]应用 S3C 赢得了 NIPS2011 的 Transfer Learning Challenge 比赛。

2.3.2 机器学习的概率图模型

概率图模型（Probabilistic Graphical Model，PGM），简称图模型（Graphical Model，GM），是指一种用图结构来描述多元随机变量之间条件独立关系的概率模型，从而给研究高维空间中的概率模型带来了很大的便捷性。

对于一个 n 维随机向量 $\xi = (\xi_1, \xi_2, \cdots, \xi_n)^T$，其联合概率为高维空间中的分布，一般难以直接建模。假设每个变量为离散变量并有 m 个取值，在不作任何独立假设条件下，则需要 $m^n - 1$ 个参数才能表示其概率分布。当 $m = 2, n = 1000$ 时，参数量就十分巨大，远远超出了目前计算机的存储能力。一种有效减少参数量的方法是独立性假设。 **n 维随机向量的联合概率分解为 n 个条件概率的乘积。**

当概率模型中的变量数量比较多时，其条件依赖关系也比较复杂。我们可以使用图结构的方式将概率模型可视化，以一种直观、简单的方式描述随机变量之

间的条件独立性的性质，并可以将一个复杂的联合概率模型分解为一些简单条件概率模型的组合。

概率图模型在形式上是由图结构组成的。图的每个节点都关联了一个随机变量，而图的边则被用于编码这些随机变量之间的关系。

图 2.16 给出了 4 随机个变量之间的条件独立性的图形化描述。图中每个节点表示一个随机变量，每条连边变量之间的依赖关系。对于一个非全连接的图，都存在一个或多个条件独立性假设，可以根据条件独立性将联合概率分布进行分解，表示为一组局部条件概率分布的乘积。

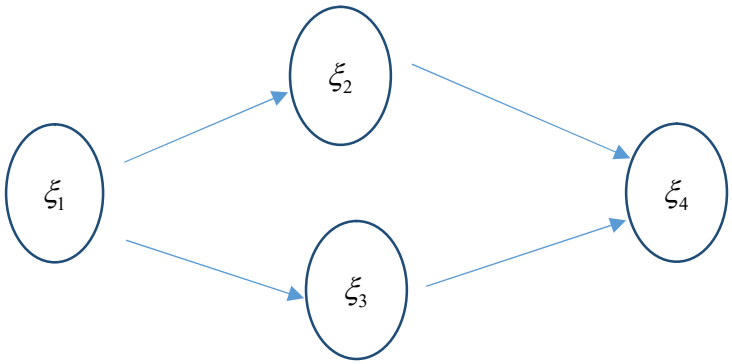


图 2.16 随机变量 ξ_1,ξ_2,ξ_3,ξ_4 之间条件独立性的图形化表示

图模型有三个基本问题：

- 表示问题：对于一个概率模型，如何通过图结构来描述变量之间的依赖关系。
- 推断问题：在已知部分变量时，计算其它变量的后验概率分布。
- 学习问题：图模型的学习包括图结构的学习和参数的学习。

很多机器学习模型都可以归结为概率模型，即建模输入和输出之间的条件概率分布。因此，图模型提供了一种新的角度来解释机器学习模型，并且这种角度有很多优点，比如了解不同机器学习模型之间的联系，方便设计新模型等。在机器学习中，图模型越来越多地用来设计和分析各种学习算法。机器学习的概率图模型主要包括以下内容，如图 2.17 所示：

图 2.17 机器学习的概率图模型的主要内容

1. 模型表示

图由一组节点和节点之间的边组成。在概率图模型中，每个节点都表示一个随机变量（或一组随机变量），边表示这些随机变量之间的概率依赖关系。

常见的概率图模型可以分为两类：有向图模型和无向图模型。有向图模型的图结构为有向非循环图，如果两个节点之间有连边，表示对应的两个变量为因果关系。无向图模型使用无向图来描述变量之间的关系。每条边代表两个变量之间有概率依赖关系，但是并不一定是因果关系。

有向概率图模型包括隐马尔科夫模型、贝叶斯网络和动态贝叶斯网络。无向概率图模型包括马尔科夫随机场 MRF，以及条件随机场 CRF。另外，还有一类混合概率图模型，如链图。

图 2.18 给出了两个代表性图模型（有向图和无向图）的示例，分别表示了四个随机变量 ξ_1,ξ_2,ξ_3,ξ_4 之间的依赖关系：

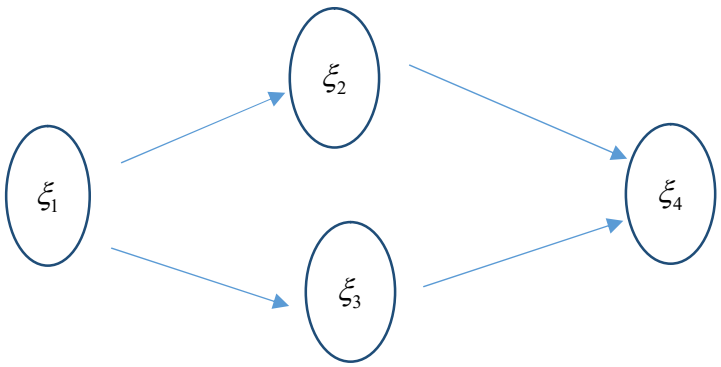


图 2.16 有向图：贝叶斯网络

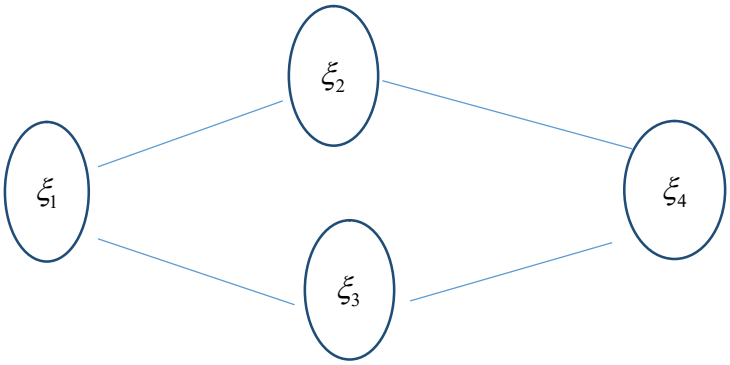


图 2.17 无向图：马尔可夫随机场

2. 为什么需要概率图模型?

对于复杂系统理解和拆分，图是一个重要的工具。概率图模型就是一类用图形模式表达基于概率相关关系的模型的总称。概率图模型结合概率论与图论的知识，利用图来表示与模型有关的变量的联合概率分布。也就是说，概率图模型是用图来表示实体之间的关联和约束，具体到机器学习领域就是**特征和类别、特征和特征之间以及类别和类别之间的关联和约束。**

图的表达能力非常强，仅仅用点和线就可以表达实体之间复杂的关系。如果给关联实体的边再加附加上概率，就进一步表达了实体之间关系的强弱和推理逻辑。概率图模型具体可以给我们带来什么呢？这里可以简单概括一下：

- 分类任务中，借助概率图建立实体之间紧凑的依赖关系，可以减小类后验概率计算所需的参数估计工作量。
- 概率图模型可以很容易与专家和领域知识结合，比如做一些实体之间的独立性假设，简化系统实体之间的依赖关系。

3. 网络结构

概率图模型如图主要分为两种，即贝叶斯网络和马尔可夫网络。

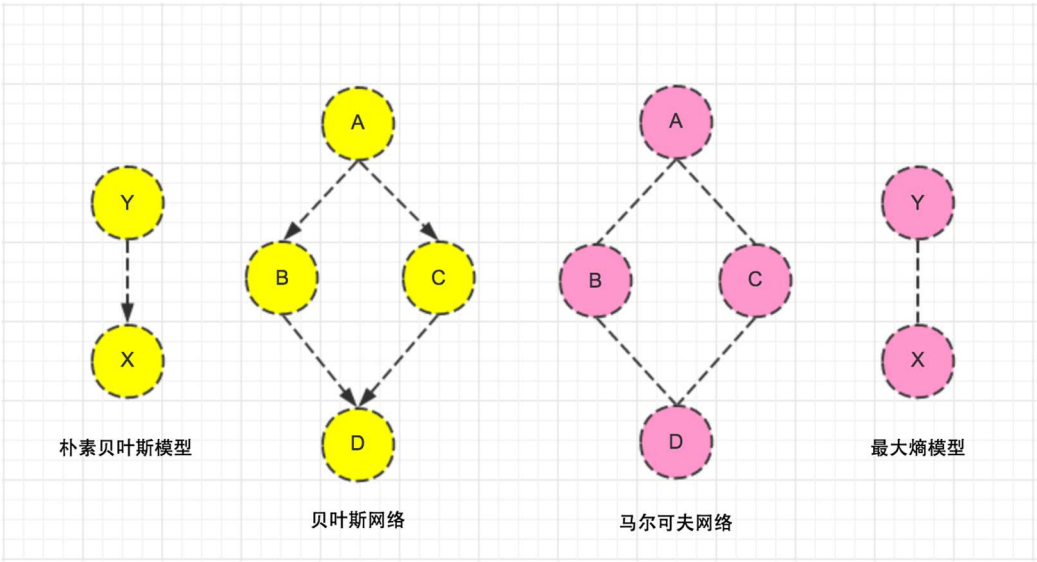


图 2.18 概率图模型网络结构

贝叶斯概率图模型是有向图，因此可以解决有明确单向依赖的建模问题，而马尔可夫概率图模型是无向图，可以适用于实体之间相互依赖的建模问题。这两种模型以及两者的混合模型应用都非常广泛。

3. 表示、推理及学习

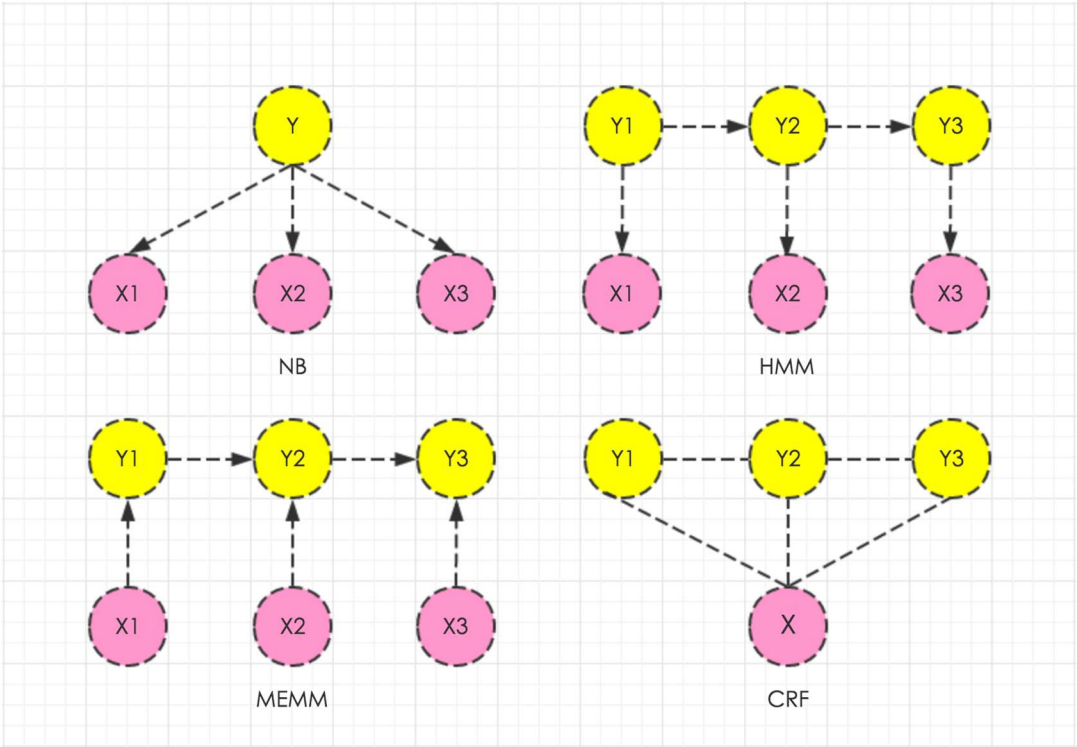
现在从表示、推理和学习的角度，来进一步说明概率图模型的优点和作用。

- 从表示的角度看，概率图模型可以很好表示实体之间关系，而且可以很容易导出相应的概率公式。同时，这种表示方法可以很容易被领域内外的人所理解。
- 从推理的角度看，当我们得到相关的信息和观测数据，我们可以很容易利用概率图导出的计算公式进行推理，给出判别结果。
- 从学习的角度看，可以利用专家经验和语料数据，对概率图模型的相关参数进行估计，效率更高而不需要估计冗余的参数。

4. 概率图模型都有哪些应用模型？

贝叶斯和马尔可夫网络是两种基本的概率图模型，结合两者又可以产生一些混合模型。那么实际应用中，有哪些模型属于概率图模型呢？

- 朴素贝叶斯模型 NB
- 最大熵模型 MEM
- 隐马尔可夫模型 HMM
- 最大熵马尔可夫模型 MEMM
- 条件随机场 CRF



对于朴素贝叶斯模型来说，特定的类别样本在不同的特征属性上具备不同的数据表征，而且特征之间有着独立性假设，即特征之间是无关联的。

对于隐马尔可夫模型来说，隐状态之间满足马尔可夫性假设，即当前状态只和前一状态有关，而与历史状态和后续状态无关；另外，还假设特征之间也是相互独立的，且特征只由当前隐状态产生。

对于最大熵马尔可夫模型来说，与隐马尔可夫模型相比，每个隐状态只依赖前一状态和当前观测，而且每组这样三者的组合都是独立的，且采用最大熵模型建模。

对于条件随机场模型来说，当前状态依赖于上下文状态和上下文观测，所以没有过多的独立性假设，可以自由搭配特征以及标注。

2.3.2 数据再表达学习的无向概率图模型

无向图学习模型，也称为马尔可夫随机场（MRF）学习模型。

1. 马尔可夫随机场及在单层学习中的应用

（1）随机场（Random field）：随机场可以看成是一组随机变量的集合（这组随机变量对应同一个样本空间）。当然，这些随机变量之间可能有依赖关系，一般来说，也只有当这些变量之间有依赖关系的时候，我们将其单独拿出来看成一个随机场才有实际意义。

随机场是随机过程的概念在空间域上的推广。**随机过程 $X(t)$ 的基本参数是时间变量 t，而随机场 $X(u)$ 的基本参数是位置向量 $u=(x,y,z)$。**因此随机场可以视为定义在一个场域参数集上的随机变量系。对于场域参数集内的任一点 u_i 都有随机变量 $X(u_i)$ 与其对应。随机变量 $X(u_i)$ 称为随机场 $X(u)$ 在位置 u_i 的状态。

随机场处于离散状态或者连续状态取决于随机变量 $X(u_i)$ 是离散的或是连续的。显然，若不考虑参数的物理意义，随机过程即可视为一维随机场。一些已有的随机场包括：

- 马尔可夫随机场(MRF)
- 吉布斯随机场 (GRF)
- 条件随机场 (CRF)
- 高斯随机场

马尔可夫随机场就是在随机场的基础上添加马尔科夫性质，从而得到马尔科夫随机场，它包含两层意思：一是什么是马尔可夫，二是什么是随机场。马尔可夫一般是马尔可夫性质的简称。它指的是一个随机变量序列按时间先后关系依次排开的时候，第 N+1 时刻的分布特性，与 N 时刻以前的随机变量的取值无关。如传染病和谣言的传播规律，就是马尔可夫的。

随机场包含两个要素：位置（site），相空间（phase space）。当给每一个位置中按照某种分布随机赋予相空间的一个值之后，其全体就叫做随机场。**马尔科**

夫随机场其实表达出随机变量之间有些关系因素是必须要考虑的，而另外则有些是可以不用考虑的。马尔科夫随机场的某个随机变量，仅仅只与其相邻的随机变量有关，与那些不相邻的随机变量无关。

（2）马尔科夫随机场的数学描述

设 δ 为 S 上的邻域系统，若随机场 $X=\{X_s, s\in S\}$ 满足如下条件：

(a) $P\{X=x\}>0, \forall x\in A$；

(b) $P\{X_s=x_s \mid X_r=x_r, r\neq s, \forall r\in\delta(s)\}=P\{X_s=x_s, X_r=x_r, \forall r\in\delta(s)\}$

则称 X 为以 δ 为邻域系统的马尔科夫随机场，上式称为马尔科夫随机场的局部特性。

概率图模型是一类用图的形式表示随机变量之间条件依赖关系的概率模型，是概率论与图论的结合。把马尔科夫随机场映射到无向图中，此无向图中的节点都与某个随机变量相关，连接着节点的边代表与这两个节点有关的随机变量之间的关系。无向图模型也叫马尔科夫随机场或马尔科夫网络，**无向图模型有一个简单的独立定义：两个节点集 A、B 都与给定的第三个节点集 C 相互条件独立，A、B 节点之间的路径都被 C 中的节点分开。**

相比之下，有向图模型也叫贝叶斯网络或信念网络，有向图模型有一个更复杂的独立性观念。

形式上，一个马尔可夫网络包括：

（a）一个无向图 $G=(V,E)$，每个顶点 $v\in V$ 表示一个在集合中的随机变量，每条边 $(u,v)\in E$ 表示随机变量 u 和 v 之间的一种依赖关系。

（b）一个函数集合 $\{f_k\}$（也称为因子或者团因子，有时也称为特征），每一个 f_k 的定义域是图 $G=(V,E)$ 的团或子团 k。每一个 f_k 是其定义域（团或子团元素 k）到非负实数的映射。

马尔可夫性质可以看作是马尔可夫随机场的微观属性，那么其宏观属性就是其联合概率分布。

将概率无向图模型的联合概率分布表示为其最大团上的随机变量的函数的乘积形式的操作，称为概率无向图模型的因子分解。

联合分布函数：联合分布（吉布斯测度）用马尔可夫网络可以表示为：

$$P(X=x)=\frac{1}{Z}\prod_k f_k(x_{\{k\}}) \qquad (2.74)$$

其中 $x=\left(x_{\{1\}},x_{\{2\}},\cdots,x_{\{k\}},\cdots\right)$ ，

$x_{\{k\}}=(x_{\{k,1\}},x_{\{k,2\}},\cdots,\ x_{\{k,|ck|\}})$ 是随机变量 ， $x_{\{k\}}$ 在第 k 个团的状态（$|ck|$ 是第 k 个团中包含的节点数），乘积包括了图中的所有团。

注意，由马尔可夫性质，**在团内的节点存在依赖关系，而在团之间是不存在依赖关系**。这里，Z 是划分函数，有

$$Z=\sum_{x\in X}\prod_{k}f_k(x_{\{k\}})$$

实际上，马尔可夫网联络经常表示为对数线性模型。通过引入特征函数 Φ_k，得到:

$$f_k=\exp\left(w_k^T\Phi_k\left(x_{\{k\}}\right)\right) \qquad (2.75)$$

和

$$P(X=x)=\frac{1}{Z}\exp\left(\sum_k w_k^T\Phi_k\left(x_{\{k\}}\right)\right) \qquad (276)$$

以及划分函数

$$Z=\exp\left(\sum_k w_k^T\Phi_k\left(x_{\{k\}}\right)\right) \qquad (2.77)$$

其中，Z 是归一化因子，w_k 是权重，Φ_k 是势函数，是从团 k 到实数域的一个映射。这些函数有时亦称为吉布斯势。**术语“势”源于物理，通常从字面上理解为在临近位置产生的势能。**

对数线性模型是对势能的一种便捷的解释方式。一个这样的模型可以简约的表示很多分布，特别是在邻域很大的时候。另一方面，负的似然函数是凸函数也带来便利。但是即便对数线性的马尔可夫网络似然函数是凸函数，计算似然函数的梯度仍旧需要模型推理，而这样的推理通常是难以计算的。

无向图模型的马尔可夫性质。马尔可夫网络有这样的马尔可夫性质：**图的顶点 u 在状态的概率只依赖顶点 u 的最近临节点，并且顶点 u 对图中的其它任何节点是条件独立的**。该性质表示为：

$$P(X_u=x_u\mid X_v,v\neq u)=P(X_u=x_u\mid X_v,v\in N_u) \qquad (2.78)$$

顶点 u 的最近临节点集合 N_u 也称为顶点 u 的马尔可夫毯。

(3) 条件随机场(Conditional random fields， CRF)，CRF 是马尔科夫随机场的特例，它假设马尔科夫随机场中只有 X 和 Y 两种变量，X 一般是给定的，而 Y 一般是在给定 X 的条件下的输出。这样马尔科夫随机场就特殊化成了条件随机场。设 X 与 Y 是随机变量，$P(Y|X)$ 是给定 X 时 Y 的条件概率分布，若随机变量 Y 构成的是一个马尔科夫随机场，则称条件概率分 $P(Y|X)$)是条件随机场（CRF）。CRF 是一种判别式图模型，因为其强大的表达能力和出色的性能，得到了广泛的应用。从最通用角度来看，**CRF 本质上是给定了观察值集合的马尔可夫随机场**。在这里，我们直接从最通用的角度来认识和理解 CRF，最后可以看到，线性 CRF 和所谓的高阶 CRF，都是某种特定结构的 CRF。

（4）线性链条件随机场

无向图的结构理论上可以是任意的，但在 NLP (Natural Language Processing) 中对于标记处理问题，对其建模主要用最简单最普通的链式结构，即线性链条件随机场。如下图，可以看到节点为线性链结构，节点对应了序列 Y 的元素，而观察序列 X 不做任何独立性假设，但 X 序列的结构也可以是线性链结构。

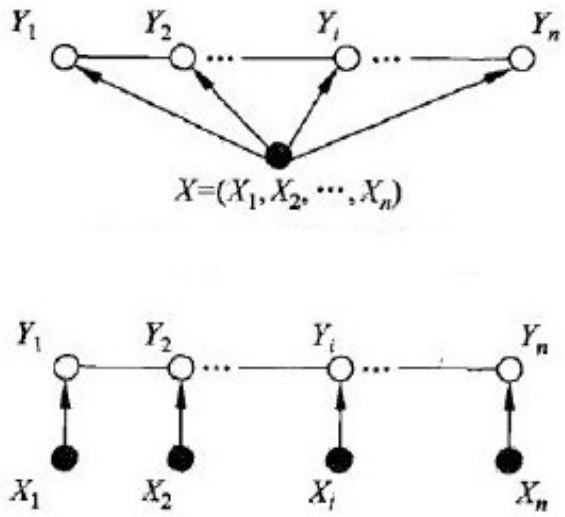


图 2.22 线性链条件随机场

综上所述，设有线性链结构的随机变量序列：

$$X=(X_1,X_2,\cdots,X_n), Y=(Y_1,Y_2,\cdots,Y_n) \qquad (2.79)$$

在给定观察序列 X 的条件下，随机变量序列 Y 的条件概率分布为 $P(Y|X)$，若其满足马尔科夫特性，即

$$P(Y_i \mid X, Y_1, Y_2, \cdots, Y_n) = P(Y_i \mid X, Y_{i-1}, Y_{i+1}) \qquad (2.80)$$

这时 $P(Y \mid X)$ 则为线性链条件随机场。

在线性链随机场中，在给定的观察序列 X 情况下，某个特定序列 Y 的概率为 $P(Y \mid X)$，根据定义有：

$$P(Y \mid X) = \exp\left(\sum_{i,k} \lambda_k t_k(Y_{i-1}, Y_i, X, i) + \sum_{i,l} \mu_l s_l(Y_i, X, i) \right) \qquad (2.81)$$

其中 $t_k(Y_{i-1}, Y_i, X, i)$ 表示转移函数，表示在序列 X 下，序列 Y 在位置 $i-1$ 及对应的值的转移概率，而 $s_l(Y_i, X, i)$ 表示状态函数，表示在序列 X 下，序列 Y 位置 i 对应的值的概率。另外，λ_k 和 μ_l 分别为两个函数的权重。

转移函数和状态函数都称为特征函数，特征函数一般取值为 0 或者 1，满足特征函数的则取 1，否则去 0。比如下面的转移函数，只有当 Y_{i-1} 和 Y_i 满足一定条件时才为 1，否则为 0：

$$t_k(Y_{i-1}, Y_i, X, i) = \begin{cases} 1, & Y_{i-1}, Y_i \text{应满足的条件} \\ 0, & \text{其他情况} \end{cases} \qquad (2.82)$$

如果令 $s_l(Y_i, X, i)$=$s_l(Y_{i-1}, X, i)$，则转移函数和状态函数可以统一由特征函数表示。对特征在各个位置 i 求和，有

$$F_k(Y, X) = \sum_{i=1}^{n} f_k(Y_{i-1}, Y_i, X, i) \qquad (2.83)$$

最后再加上归一化，最终条件随机场的条件概率为：

$$P(Y \mid X) = \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} \lambda_k F_k(Y, X) \right) \qquad (2.84)$$

其中

$$Z(X) = \sum_{y} \exp\left(\sum_{k=1}^{K} \lambda_k F_k(Y, X) \right) \qquad (2.85)$$

训练过程就是通过一组样本，希望能够得到 CRF 对应的分布形式，并且用这种分布形式对测试样本进行分类。也就是测试样本中每个随机变量的取值。

在实际应用中，团的势函数主要由用户自己定义的特征函数组成，即用户自己定义一组函数，这些函数被认为是可以用来帮助描述随机变量分布的。而这些特征函数的强弱以及正向、负向是通过训练得到的一组权重来表达的，这样，实际应用中我们需要给出特征函数以及权重的共享关系(不同的特征函数可能共享同一个权重)，而团的是函数本质上成了对应特征函数的线性组合。这些权重就成了 CRF 的参数。因此，本质上，图的结构是用户通过给出特征函数的定义确定的（例如，只有一维特征函数，对应的图上是没有边的)还有，CRF 的分布成了对数线性形式。

看到这个分布形式，我们自然会想到用最大似然准则来进行训练。对其取对数之后，表达式是凸的，也就是具有全局最优解，而且，其梯度具有解析解，这样可以用 LBFGS 来求解极值。

此外，也可以使用最大熵准则进行训练，这样可以用比较成熟的 GIS 和 IIS 算法进行训练。由于对数线性的分布形式下，最大熵准则和最大似然准则本质上是一样的，所以两者区别不是很大。

此外，由于前面两种训练方法在每一轮迭代时，都需要 inference， 这样会极大地降低训练速度。因此普遍采用另一种近似的目标函数，称为伪似然。它用每个随机变量的条件分布(就是给定其他所有随件变量的分布)之积来替代原来的似然函数，根据马尔科夫性质，这个条件分布只和其邻居有关(马尔科夫毯)，这样在迭代过程中不需要进行全局的 inference， 速度会得到极大的提升。经验表明，当特征函数很多取实数值时，伪似然的效果跟最大似然的差不多，甚至略好于后者。但对于大量二元特征，伪似然的效果就很差了。

训练 CRF 主要就是要训练特征函数的权重，对于训练集 $(x_1,y_1),(x_2,y_2),\cdots(x_n,y_n)$，采用极大似然估计法计算权重参数，条件概率的对数似然函数为：

$$L(\lambda)=\sum_{x,y}\tilde{p}(x,y)\sum_{i=1}^{n}\left(\sum_{k=1}^{K}\lambda_k f_k(y_{i-1},y_i,x,i)\right)-\sum_{x}\tilde{p}(x)\log Z(x) \qquad (2.86)$$

其中 $\tilde{p}(x,y)$ 为训练样本集中 (x,y) 的经验概率，它等于 (x,y) 同时出现的次数除以样本空间容量；$\tilde{p}(x)$ 为训练样本集中 x 的经验概率，它等于 x 出现的次数除以样本空间容量。然后令 $\frac{\partial L(\lambda)}{\partial\lambda}=0$，得到解。

在无监督的数据再表达学习中，用到一种特殊形式的马尔可夫随机场结构，被称为玻尔兹曼分布，其中团的非负势函数定义为如下述形式：

$$p(x,h)=\frac{1}{Z_\theta}\exp(-\varepsilon_\theta(x,h)) \tag{2.87}$$

其中$\varepsilon_\theta(x,h)$是能量函数，包含着被 MRF 团的势描述的相互作用，θ是描述这些相互作用的模型参数。

这个玻尔兹曼机最早是被作为一个对称—二进制随机变量（或者单元）网络来定义的。这些单元被分成两组：（1）可见单元$x\in\{0,1\}^{d_x}$，表达数据，（2）隐藏单元$h\in\{0,1\}^{d_h}$，通过相互作用调节可见单位之间的依赖关系。通过下述能量函数确定相互作用的模式：

$$\varepsilon_\theta^{\mathrm{BM}}(x,h)=-\frac{1}{2}x^TUx-\frac{1}{2}h^TVh-x^TWh-b^Tx-d^Th \tag{2.88}$$

其中$\theta=\{U,V,W,b,d\}$是模型参数，分别编码可见到可见单元的相互作用、隐藏单元到隐藏单元的相互作用、可见单元到隐藏单元的相互作用、可见单元的自连接、隐藏单元的自连接（被称之为偏置）。为了避免过参数化，U,V的对角元素置为 0。

玻尔兹曼机的能量函数用玻尔兹曼分布来定义，划分函数Z_θ定义为：

$$Z_\theta=\sum_{x_1=0}^{x_1=1}\cdots\sum_{x_{d_x}=0}^{x_{d_x}=1}\sum_{h_1=0}^{h_1=1}\cdots\sum_{h_{d_h}=0}^{h_{d_h}=1}\exp\left(-\varepsilon_\theta^{\mathrm{BM}}(x,h;\theta)\right) \tag{2.89}$$

这个联合概率分布由下述条件分布给出：

$$P(h_i\mid x,h_{\setminus i})=\mathrm{sigmoid}\left(\sum_j W_{ji}x_j+\sum_{i'\neq i}V_{ii'}h_{i'}+d_i\right) \tag{2.90}$$

$$P(x_j\mid h,x_{\setminus j})=\mathrm{sigmoid}\left(\sum_i W_{ji}x_j+\sum_{j'\neq j}U_{jj'}x_{j'}+b_j\right) \tag{2.91}$$

通常，玻尔兹曼机的推断比较困难。例如，在给定可见单元的概率$P(h_i\mid x)$的情况下，计算h_i的条件概率，需要求出其余的隐藏单元的边际分布，这意味着要用2^{d_h-1}项来求下述和：

$$P(h_i \mid x) = \sum_{h_1=0}^{h_1=1} \cdots \sum_{h_{i-1}=0}^{h_{i-1}=1} \sum_{h_{i+1}=0}^{h_{i+1}=1} \cdots \sum_{h_{d_h}=0}^{h_{d_h}=1} P(h \mid x) \quad (2.92)$$

然而，在可见和隐藏单元之间的相互作用模式中有一些更好的选择，这就是受限玻尔兹曼机。

2. 受限玻尔兹曼机学习模型(RBM)

与自编码不一样，受限波尔兹曼机 (Restricted Boltzmann Machines，简称 RBM)是一种概率模型，通过计算特征的后验概率，再经过采样来获得特征值。受限波尔兹曼机由波尔兹曼机 (Boltzmann Machines) 发展而来，它限制了波尔兹曼机中可见层 (visible layer) 和隐含层 (hidden layer)，使各层内节点没有直接的连接。这样可见层 v 和隐含层 h 的节点形成了一个二部图 (bipartite graph)。

3. 单层与多层置信网络

BP 神经网络是 1968 年由 Rumelhart 和 Mcclelland 为首的科学家提出的概念，是一种按照误差反向传播算法进行训练的多层前馈神经网络，是目前应用比较广泛的一种神经网络结构。BP 网络神经网络由输入层、隐藏层和输出层三部分构成，无论隐藏层是一层还是多层，只要是按照误差反向传播算法构建起来的网络（不需要进行预训练，随机初始化后直接进行反向传播），都称为 BP 神经网络。BP 神经网络在单隐层的时候，效率较高，当堆积到多层隐藏层的时候，反向传播的效率就会大大降低，因此 BP 神经网络在浅层神经网路中应用较广，但由于其隐层数较少，所以映射能力也十分有限，所以浅层结构的 BP 神经网络多用于解决一些比较简单的映射建模问题。

在深层神经网络中，如果仍采用 BP 的思想，就得到了 BP 深层网络结构，即 BP-DNN 结构。由于隐藏层数较多（通常在两层以上），Δw、Δb 自顶而下逐层衰减，等传播到最底层的隐藏层时，Δw、Δb 就几乎为零了。如此训练，效率太低，需要进行很长时间的训练才行，并且容易产生局部最优问题。因此，便有了一些对 BP-DNN 进行改进的方法，例如，采用 ReLU 的激活函数来代替传统的 sigmoid 函数，可以有效地提高训练的速度。此外，除了随机梯度下降的反向传播算法，还可以采用一些其他的高效的优化算法，例如小批量梯度下降算法（Mini-batch Gradient Descent）、动量梯度下降算法等，也有利于改善训练的效率问题。直到 2006 年，Hinton 提出了逐层贪婪预训练受限玻尔兹曼机的方法，大大提高了训练的效率，并且很好地改善了局部最优的问题，开启了深度神经网络发展的新时代。Hinton 将这种基于玻尔兹曼机预训练的结构称为深度置信网络结构（DBN），用深度置信网络构建而成的 DNN 结构，即 DBN-DNN。

BP 神经网络和 DBN-DNN 根据学习任务，后传播误差，都是前后端融合的学习方法，我们将在第四部分中再详细介绍。但是，这种单纯的基于玻尔兹曼机预训练的结构实际上是一种前端学习的逐层数据再表达方法。

1）产生和背景

Boltzmann 机(也称为隐单元随机 Hopfield 网络)是一种随机递归神经网络(需要马尔可夫随机场辨识)。也被看作是 Hopfield 网络的随机、生成对应物。它们是最早能够学习内部表示的神经网络之一，并且能够表示和（给定足够的时间）解决困难。由于它们的训练算法的局部性和赫布性质（Hebbian principle）以及它们的并行性和动力学与简单物理过程的相似性，所以理论上非常吸引人。

Hebbian principle 的精确表达：如果两个神经元常常同时产生动作电位，或者说同时激动（fire），这两个神经元之间的连接就会变强，反之则变弱（neurons that fire together, wire together）。

具有无约束连通性的 Boltzmann 机器对于机器学习或推理中的实际问题还没有被证明是有用的，但是如果连通性被适当地约束，则该学习可以变得足够有效从而对实际问题有用[33]。

2） 玻尔兹曼机

（1）基础结构

玻尔兹曼机是一个由很多单元构成的网络模型，所有单元可以分为可视单元 V 和隐藏单元 H 两种，其中可视单元是肯定存在的，它是从外界“环境”中获取相关信息的，并且是通过外界环境决定它的状态；而隐藏单元可能没有，如果有的话，它的状态不是由外界环境决定的。其中每个单元都有未激活（取值 0）和激活（取值 1）两种情况，分别意味着系统当前状态下对该单元的接受或者拒绝。每两个点之间有权重值 w_{ij}，它表示两点之间的成对约束的权重。和 Hopfield 模型相同的是它也有一个模型的“能量”的概念，定义的值为

$$E_i = \sum_{i<j} w_{ij} s_i s_j + \sum_i \theta_i s_i \qquad (2.93)$$

其中 s_i 表示它的状态，$s_i \in \{0,1\}$，θ_i 表示单元 i 偏移量。整个模型的权重值构成一个权重矩阵 W，并且它是一个对角线元素为 0 的对角矩阵。

下图为一个含有三个隐藏单元（蓝色）和四个可视单元（白色）的 BM 模型，其中给出了几个单元之间的权重。

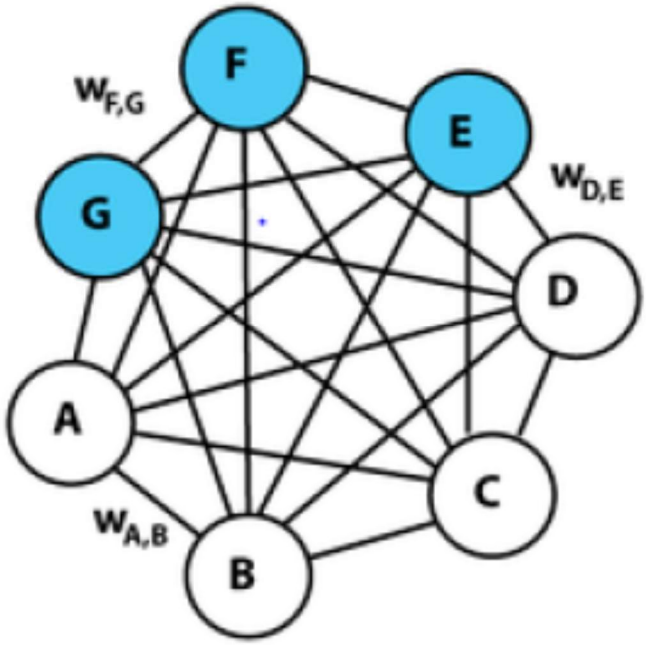


图 2.23 BM 模型

（2）概率模型

对于模型里的每一个单元都由激活（取值 1）和未激活（取值 0）两种状态，则都有自身的状态概率。那么其中一个单元 i 处于激活和未激活状态下，整个网

络的能量是不同的，记它的差为

$$\Delta E_i = E_{i=\text{off}} - E_{i=\text{on}} \qquad (2.94)$$

那么就有:

$$\Delta E_i = \sum_j w_{ij} s_i + \theta_i \qquad (2.95)$$

这里根据 Boltzmann 因子用相对概率来代替每个状态下的能量，可以得到:

$$\Delta E_i = -\kappa_B T \ln(p_{i=\text{off}}) - \left(-\kappa_B T \ln(p_{i=\text{on}})\right) \qquad (2.96)$$

其中 κ_B 玻尔兹曼常数，并且会被吸收到温度 T 的全局概念中。对于一个单元的激活和未激活的概率和应该为 1，所以可以对上面公式进行转化:

$$\frac{\Delta E_i}{T} = \ln p_{i=\text{on}} - \ln p_{i=\text{off}} \qquad (297)$$

从而

$$e^{-\frac{\Delta E_i}{T}} = \frac{1}{p_{i=\text{on}}} - 1 \qquad (2.98)$$

即

$$p_{i=\text{on}} = \frac{1}{1 + e^{-\frac{\Delta E_i}{T}}} \qquad (2.99)$$

其中 T 表示整个下系统的温度标量。

（3）训练方法

对于整个系统来说，网络通过反复选择不同单元状态，在一定温度下网络运行足够长的时间之后，整个网络状态的 概率仅仅取决于该全局状态的能量，而不取决于初始状态，这意味着全局状态的对数概率在其能量中变为线性。此时就是网络所需要达到的平衡状态，当机器处于“热平衡”时，这意味着全局状态的概率分布已经收敛。从高温开始运行网络，其温度逐渐降低，直到在较低温度下达到热平衡。然后它可以收敛到能量水平围绕全局最小值波动的分布。该过程称为模拟退火过程。所以我们需要训练该网络获得最优的权重矩阵，使它具有最高的概率可以收敛到最低能量的状态[34]。

将训练集上的分布记为 $p(v)$。当整个系统达到热平衡的时候，分布会趋于全局的一个收敛，计算它在隐藏层的边缘分布，记作 $p'(v)$。那么目标就是用模型最后得到的概率分布 $p'(v)$ 去逼近真实的分布 $p(v)$。为了描述两者差异，使用的是 KL 散度 G（相对熵）, 即有：

$$G=\sum_{v} p'(v)\ln\frac{p(v)}{p'(v)} \qquad (2.100)$$

其中v表示的是所有的可能状态。由于各个状态的能量是由权重值来决定的，所以G是一个关于权重值w_{ij}的函数，那么就可以使用梯度下降法来通过改变权重使G达到足够小。通过计算可以得：

$$\frac{\partial G}{\partial w_{ij}}=-\frac{1}{T}\left(p(i,j)-p'(i,j)\right) \qquad (2.101)$$

其中$p(i,j)$表示在外界条件下，i单元和j单元同时处于激活状态的概率，而$p'(i,j)$是未受外界环境影响下，网络自身运行达到平衡的时候，i单元和j单元同时处于激活状态的概率。那么要去减小G，则要让权重值做以下更新：

$$\Delta w_{ij}=\varepsilon\left(p(i,j)-p'(i,j)\right) \qquad (2.102)$$

其中ε表示各个权重的变化量。从上式只需要使用部分的可用信息（单独两单元之间的权重值），就可以使全体结构达到最优的情况。

3）受限玻尔兹曼机

玻尔兹曼机是一个很好的学习模型，但是存在一个严重的实际问题，就是当机器被运用到更大的量级上时可能会停止正确学习。所以就此导致了玻尔兹曼机的实际应用不是很有效。随后 Paul Smolensky 在玻尔兹曼机的基础上发明了受限的玻尔兹曼机器，在 Geoffrey Everest Hinton 在 2006 年发明了快速学习算法后，玻尔兹曼机应用开始变得广泛起来。受限玻尔兹曼机在降维、分类、特征学习以及主题建模等等方面得到了很多应用。根据任务的不同，受限玻尔兹曼机可以使用监督学习和无监督学习的方法来进行训练[35]。

受限玻尔兹曼机是一种可以通过输入数据集学习概率分布的随机生成网络，它是玻尔兹曼机的变体，但是它限定模型必须为二部图，即图中的每条边的两个端点必须一个为可见单元，另一个为隐单元（而玻尔兹曼机中的边可以连接同一类的单元），此限定使用的它比一般的玻尔兹曼机更加高效可用。

（1） 概率模型

假定网络中的可见单元集合为V，隐单元集合为H，那么整个网络就是V和H构成的一个二部图，整个网络的能量为：

$$E(v,h)=-\sum_{i\in V}a_i v_i-\sum_{j\in H}a_j v_j-\sum_{i,j}v_i h_i w_{ij} \qquad (2103)$$

其中v_i和h_j分别是可视单元i和隐单元j的值，a_i和b_j是其对应的偏移量。在网络中每对单元（可见单元和隐单元）的概率为：

$$p(v,h)=\frac{1}{Z}e^{-E(v,h)} \qquad (2.104)$$

其中Z为归一化函数，定义为

$$Z=\sum_{v,h}e^{-E(v,h)} \qquad (2.105)$$

而对于可见单元v的概率，可以用对所有可能的隐单元概率和来表示：

$$p(v)=\frac{1}{Z}\sum_{h}e^{-E(v,h)} \qquad (2.106)$$

而 RBM 是一个二部图，层内没有连接，所以可见单元的激活状态在给定的隐层取值的情况下是条件独立的。那么可见层情况v对于隐层情况h的条件概率为：

$$p(v\mid h)=\prod_{i=1}^{m}p(v_i\mid h) \qquad (2.107)$$

类似可以得到h对于隐层情况v的条件概率为：

$$p(h\mid v)=\prod_{j=1}^{n}p(h_j\mid v) \qquad (2.108)$$

（2） 训练

受限玻尔兹曼机的训练目标就是通过调整权重值和偏移量去使训练集状态的概率达到最大，等价于求对数的最大似然，即要去求得：

$$\arg\max\{\log p(v):w\} \qquad (2.109)$$

对于受限玻尔兹曼机的训练，即优化权重矩阵w,常用到的方法是杰弗里·辛顿提出的对比散度（contrastive divergence，CD）算法。这一算法最早被用于训练辛顿提出的“专家积”模型。这一算法在梯度下降的过程中使用吉布斯采样完成对权重的更新，与训练前馈神经网络中利用反向传播算法类似。

CD 算法的基本思想是：首先根据可见层数据v来得到隐h的状态，然后通过h来重构可见向量v_1，然后再根据v_1来生成新的隐藏向量h_1。因为 RBM 的特殊结构(层内无连接，层间有连接)， 所以在给定v时，各个隐藏单元h_j的激活状态之间是相互独立的，反之，在给定h时，各个可见单元的激活状态v_i也是相互独立的。

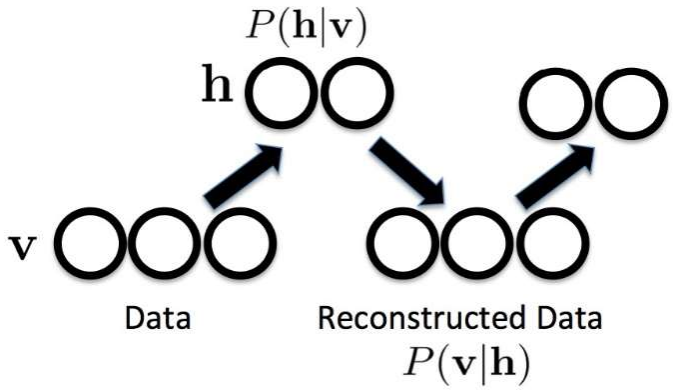


图 2.24 受限玻尔兹曼机的 CD 算法

首先对于重构的具体过程如下图所示：

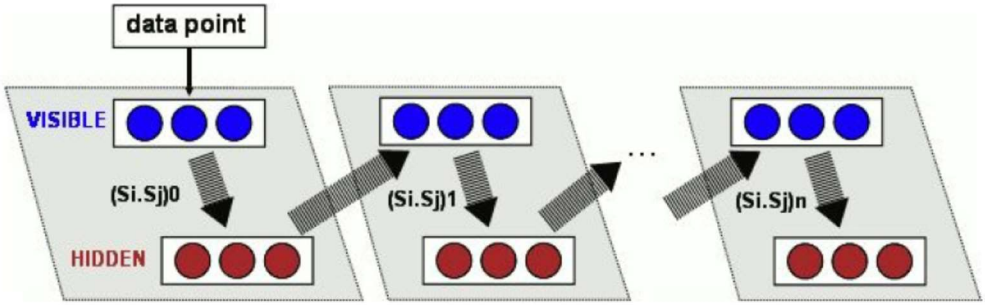


图 2.25 受限玻尔兹曼机训练与重构图

Step1 首先得到数据集中的一个数据点；

Step2 用该数据点设置为 v_i；

Step3 用隐藏层的条件概率公式来计算得到隐藏层中各单元的状态 h_j；

Step4 计算得到乘积 $(v_i, h_j)_0$；

Step5 然后利用计算得到的隐藏层 h_j 来计算新的可见层 s_i，即重构；

Step6 用上一步的 s_i 再次计算得到一个隐藏层 h_j；

Step7 用上两个步骤得到的新状态计算 $(v_i, h_j)_1$；

Step8 重复 Step5、6、7，得到 $(v_i, h_j)_n$；

那么就对于模型的训练就为：

Step1 记 $CDpos = 0, CDneg = 0$；

Step2 进行上面重构过程；

Step3 计算 $CDpos = (v_i, h_j)_0$ 和 $CDneg = (v_i, h_j)_n$ 其中 $(v_i, h_j)_n$ 是矩阵形 式）

Step4 再除以样本总数得到平均 $\bar{C}Dpos$ s 和 $\bar{C}Dneg$;

Step5 计算得到 $CD=(v_i,h_j)_0-(v_i,h_j)_0=\bar{C}Dpos-\bar{C}Dneg$;

Step6 更新权重 $W'=W+\sigma CD$;

Step7 计算重构误差，重复训练直到达到误差要求。

其中学习率 σ 是通过 $P(v)$ 对 w_{ij} 求偏导得到。用同样的方法可以对偏移量进行迭代。这样每加入一个训练样本就可以得到一次参量的更新昏都会使训练集状态的概率增大。

4）多层受限玻尔兹曼机

在多层受限玻尔兹曼机里，第一层还是为可见层 V ，即与外界环境相关联的单元层，另外还有 m 个隐藏层 $H_i,i=1,2,\cdots,m$ 。V 层的数据由训练集给出，它与第一个隐藏层相连接，充分训练第一个 RBM 模型，可以得到一个 H_1 层的状态表示，再将 H_1 层作为输入层，与下个隐藏层 H_2 连接，再训练这个新的 RBM 模型，这样就可以得到一个 H_2 层的状态表示，一直下去，到最后个隐藏层，就可以得到最后一层 H_m 的一个表示可以作为输出数据。

（1）模型及训练方法

首先考虑一个有两个隐藏层的模型，各层内部没有连接，那么这整个模型处于 v,h^1,h^2 状态时候的能量为：

$$E(v,h^1,h^2)=-v^TW^1h^1-h^1W^2h^2 \quad (2.110)$$

其中 $\theta=(W^1,W^2)$ 为模型的参数，即相连接的两层之间的联系。

并且对于可见单元点v的模型概率为：

$$p(v;\theta)=\frac{1}{Z(\theta)}\sum_{h^1,h^2}\exp(-E(v,h^1,h^2;\theta))$$

可见单元和两组隐藏单元的条件分布概率由下面函数给出：

$$p(h_j^1=1|v,h^2)=\sigma(\sum_i W_{ij}^1 v_i+\sum_m W_{jm}^2 h_j^2)$$

$$p(h_m^2=1|h^1)=\sigma(\sum_j W_{im}^2 h_i^1)$$

$$p(h_i=1|h^1)=\sigma(\sum_j W_{ij}^1 h_j)$$

对 BM 和 RBM 中用的是最大似然估计的方法来进行模型训练，在这里对 DBM 的训练使用的是贪婪算法。贪婪算法是在对第一个 RBM 模型进行训练之后可以得到第一个隐藏层的表示，然后将第一个模型当做一个整体，第一个模型的输出当做第二个 RBM 模型的输入，再对第二个模型进行训练，但这是个训练的模型在隐藏层之间的连接是有方向的，为有向图，而 DBM 的模型是无向图，所以需要一些修改才能运用。当训练完第一个 RBM 模型后，生成模型可以写做下面概率模型：

$$p(v;\theta)=\sum_{h^1}p(h^1;W^1)p(v\,|\,h^1;W^1)$$

其中

$$\sum_{h^1}p(h^1;W^1)=\sum_v p(h^1,v;W^1)$$

是由参数定义的h^1的先验概率。在第二个 RBM 模型中，使用

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

来代替$\sum_{h^1}p(h^1;W^1)$，如果第二个模型运行顺利，那么$p(h^1;W^2)$就可以更好的描述h^1的先验概率，且$p(h^1;W^2)$就会变为一个更好的h^1的后验分布模型。因为第二个 RBM 模型更好取代了$p(h^1;W^1)$，那么就可以对这两个模型的关于h^1的概率求平均数来得到$p(v;W^1,W^2)$，即由底层和上一层得到的概率的一半相加得到。但是，由于h^2对v是独立的，这里的计算相当于重复了。

为了消除了上述的重复计算问题，对贪婪算法做一点小的改变：对于最下层的 RBM，使输入的单元加倍且约束可见层和隐藏层之间的权重。那么修改相关参数后的第一层 RBM 的相关的条件概率分布为：

$$p(h_j^1=1\,|\,v)=\sigma\left(\sum_i W_{ij}^1 v_i+\sum_i W_{ij}^1 v_i\right)$$

$$p(h_j^1=1\,|\,v)=\sigma\left(\sum_i W_{ij}^1 v_i\right)$$

相反的，可以把最顶端的 RBM 模型的隐藏单元加倍，就得到最顶层的 RBM 分布如下：

$$p(v_i=1\,|\,h^1)=\sigma\left(\sum_j W_{ij}^1 h_j^1\right)$$

$$p(h_j^1=1\,|\,h^2)=\sigma\left(\sum_m W_{jm}^2 h_m^2+\sum_m W_{jm}^2 h_m^2\right)$$

$$p(v_m^2=1\,|\,h^1)=\sigma\left(\sum_j W_{jm}^2 h_j^1\right)$$

把两个模型组合在一起后，进入第一个隐藏层的总输入，由以下的条件分布：

$$p(h_j^1=1\,|\,h^2)=\sigma\left(\sum_m W_{jm}^2 h_m^2+\sum_i W_{ij}^1 v_i\right)$$

当使用贪婪算法训练两个以上的 RBM 算法时，只需要修改最低和最顶层的两个 RBM 模型。对中间的 RBM，只需要在组合成 DBM 的时候，简单的把两个方向上的权值均分就可以。

用上述贪婪算法方法进行训练时，以通过 RBM 堆单一向上传播的方式进行，在可见层给一个训练数据，通过传递，每一个隐藏单元层都可以被激活，而且通过翻倍的形式弥补自上而下的反馈时的不足。

2.3.3 数据再表达学习的有向图概率模型

有向图概率模型，也称为贝叶斯网络（Bayesian Network），或置信网络（Belief Network，BN），是指用有向图来表示概率分布的图模型。假设一个有向图 $G(V,E)$，节点集合 $V=\{\xi_1,\xi_2,\cdots\xi_K\}$ 表示 K 个随 机变量，节点 k 对应随机变量 ξ_k。 E 为边的集合，每条边表示两个变量之间的因果关系。

定义–贝叶斯网络：对于一个随机向量 $\xi=(\xi_1,\xi_2,\cdots\xi_K)^T$ 和一个有 K 个节点的有向非循环图 $G(V,E)$，$G(V,E)$ 中的每个节点都对应一个随机变量，可以是可观

测的变量，隐变量或是未知参数。$G(V,E)$ 中的每个连接 e_{ij} 表示两个随机变量 ξ_i 和 ξ_j 之间具有非独立的因果关系。ξ_{π_k} 表示随机变量 ξ_k 的所有父节点变量集合，每个随机变量的局部条件概率分布为 $p(\xi_k \mid \xi_{\pi_k})$。如果 $\xi=(\xi_1,\xi_2,\cdots\xi_K)^T$ 的联合概率分布可以分解为每个随机变量 ξ_k 的局部条件概率的连乘形式，即

$$p(\xi)=\prod_{k=1}^{K} p(\xi_k \mid \xi_{\pi_k}) \qquad (2111)$$

那么 (G,ξ) 构成了一个贝叶斯网络。

贝叶斯网络的局部马尔可夫性质：对一个贝叶斯网络，其局部马尔可夫性质是指每个随机变量在给定父节点的情况下，条件独立于它的非后代节点。

- **深度置信网络（sigmoid 置信网络，Neal[31]）比较**

深度信念网络（DBN）也是由一个可见层和多个隐藏层构成的，即可以视为多个 RBM 模型叠加而成的一个神经网络，既可以把它当做一个生成模型，也可以把它当做一个判别模型。它和 DBM 一样，第一层为可见层，由外界环境输入数据集，然后依次连接这个多个隐藏层，层内之间没有连接，第一层 RBM 由可见层连接第一层隐藏层；随后都是由隐藏层连接下一个隐藏层，与 DBM 不同的是隐藏层之间的连接是有方向的，即为一个有向图，从底层向顶层传输。

它的训练是通过非监督的贪婪算法进行的：

- Step 1 首先将训练集输入到可见层，充分训练得到第一个 RBM 网络；
- Step 2 固定第一层的 RBM 的权重和偏移量，然后使用其隐藏层的单元状态，作为第二个 RBM 的输入数据；
- Step 3 充分训练第二个 RBM 后，将第二个 RBM 堆叠在第一个 RBM 模型上方；
- Step 4 重复以上步骤直到最后一个 RBM 模型。

那么这样就可以训练得到一个 DBN 模型，之后就是对模型进行有监督的调优过程：需要利用向前传播算法，从输入得到一定的输出值:

（1）利用 CD 算法训练得到的每一层的权重值和偏移量来确定每一个隐单元 的状态，得到隐单元的激励值为：

$$h_j=W_j v+b_j$$

（2）逐层向上传播，将激励值通过激活函数计算出输出值：

$$out(h_j)=\frac{1}{1+e^{h_j}}$$

（3）最后计算出输出层的输出值为 y'

然后利用向后传播算法来更新网络的权重值和偏移量：

（1）采用最小均方误差准则的算法来更新整个网络参数，代价函数为

$$E=\frac{1}{N}\sum_{i=1}^{N}(y'-y_i)^2$$

（2）采用梯度下降法来对 W 和偏移量进行更新：对于第 N 层的 RBM，求

$$\Delta=\frac{\partial E}{\partial(W^N,B^N)}$$

那么就对参数作更新：

$$(W^N,B^N)\leftarrow(W^N,B^N)+\Delta$$

这样就训练完成了一个 DBN 模型。而 DBN 的应用十分广泛，如果让模型从下往上运行，就可以输出得到一个类，即起到一个分类器或者特征提取的作用；如果给从上往下运行，那么就可以作为一个生成模型，得到想要的原数据。下图所示为一个只有三个隐藏层的模型情况：

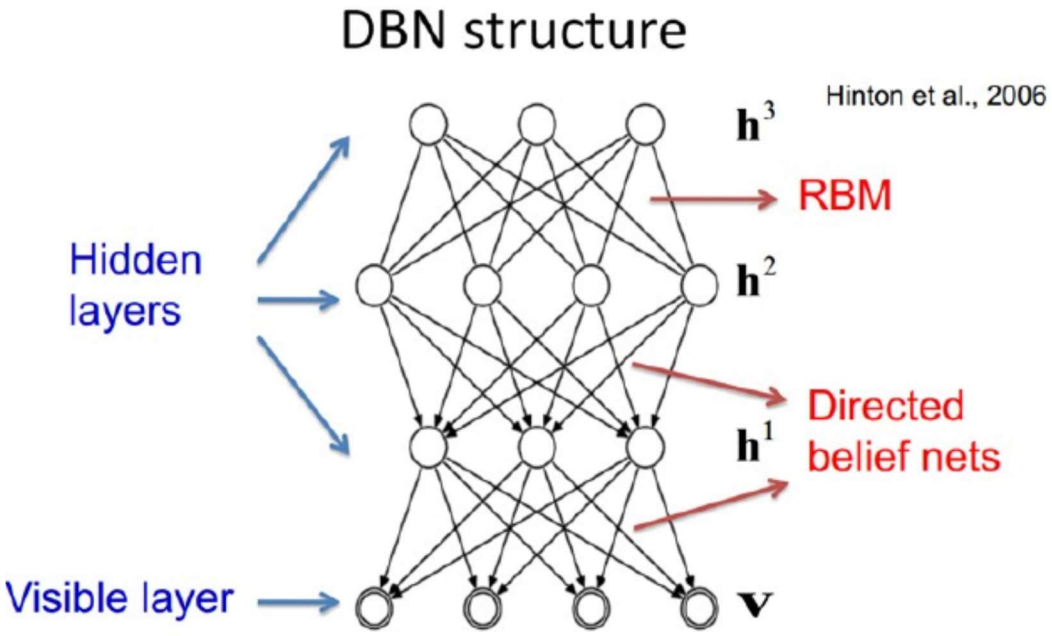


图 2.26 有三个隐藏层 DBN 模型

对于 DBN 和 DBM，都是由多个 RBM 叠加在一起而形成的，但是在隐藏层之间一个是无向图，一个是有向图。导致在训练时候有一定的差别，DBN 训练较为简单，学习速度会比较快；而 DBM 训练复杂，算法较慢，但是它的鲁棒性以及最后收敛效果会比 DBN 好很多。

5）算法应用

由于 BM 算法的实际应用要求过多，所以得到的运用实例不多。而 RBM 算法的应用面很广泛，在降维、分类、特征学习等等方面都有很多的应用。

（1）降维：在高维数据的降维应用方面，PCA 算法十分受欢迎。而 RBM 的算法特性也让其可以应用于数据降维上[36]。利用 RBM 的算法结构，可以创建一个数据的降维网络结构，如下图所示。

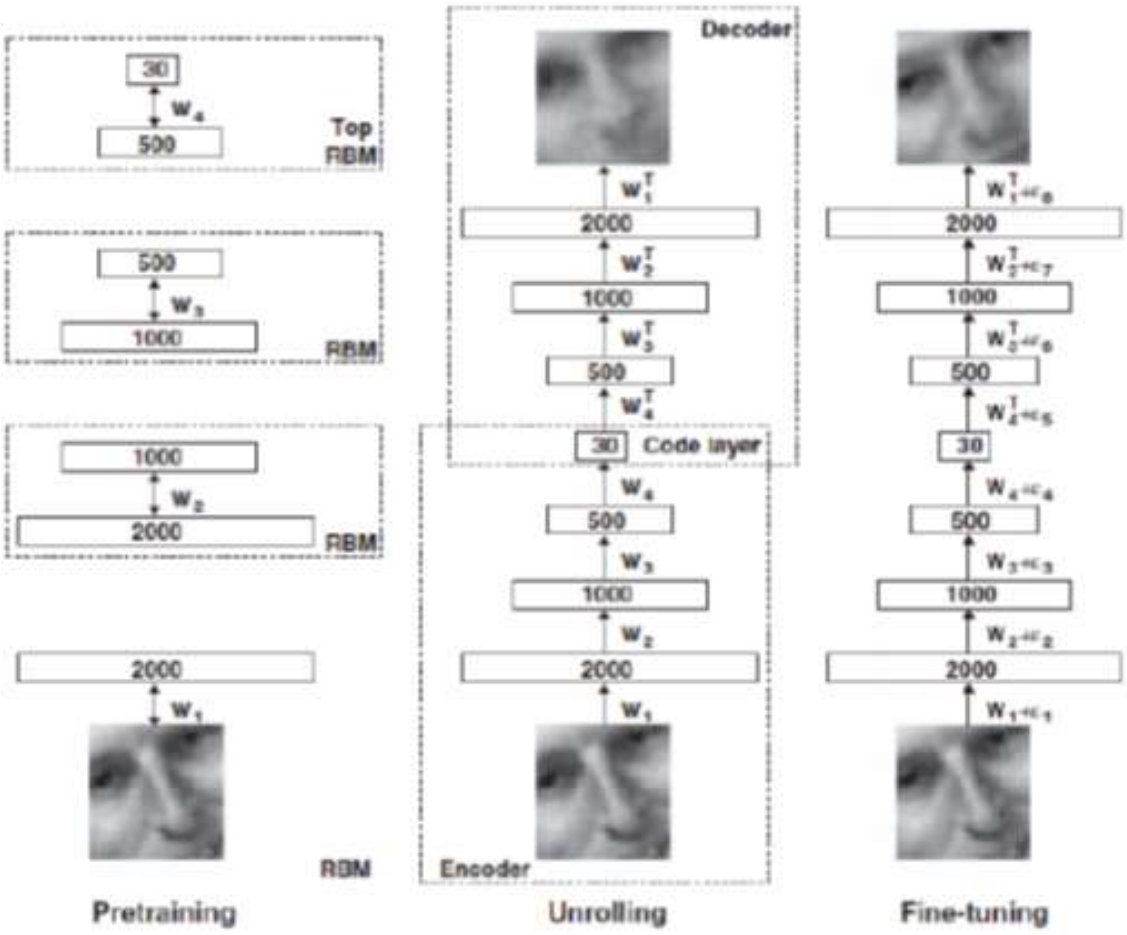


图 2.27 RBM 数据降维网络结构

其中包括三个步骤，最左侧为预训练阶段，即在每一层的 RBM 上进行预训练，得到每一层的权重矩阵和偏移量，这样就初步形成了网络；中间为显示阶段，即利用得到的网络结构和最后表示，反编码得到重构图像，和原图像进行比较；第三步即为微调阶段，由重构图像和原图像的重构差来对网络结构进行调整，即对每一层的参数进行调整，最后达到要求范围。

在实际应用中与 PCA 对比如下：

在下图(A)中，RBM 降维结构为$28*28-400-200-100-50-25-6$，与 PCA 算法一起应用于有 20000 个数据的训练集和 10000 个数据的测试集上；从上往下，第一行为随机抽取的原始数据，第二行为 RBM 降维结构得到的重构图像，下面三行不同 PCA 算法得到的重构图像，明显可以看出 PCA 算法的重构误差大出很多。随后在下图(B)中，RBM 结构为 784-1000-500-250-30，数据为 MNIST 手写笔迹的训练集；从上往下，第一行为随机抽取的原始数据，第二行为 RBM 降维结构得到的重构图像，下面两行是 PCA 算法得到的重构图像。

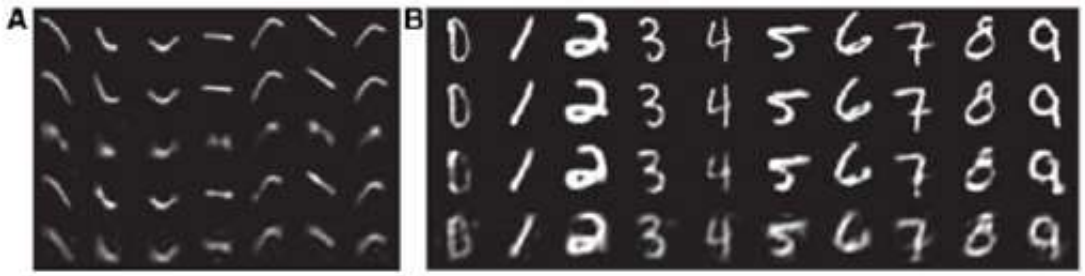


图 2.28 RBM 算法与 PCA 算法的降维比较

综合上述两个实验，可以清楚看到由 RBM 算法得到的降维网络可以更好的实现数据降维，并且大大降低重构时的误差。