中国科学院大学

University of Chinese Academy of Sciences

Deep Learning

Deep Generation Network

Xinfeng Zhang（张新峰）

School of Computer Science and Technology

University of Chinese Academy of Sciences

Email: xfzhang@ucas.ac.cn

计算机科学与技术学院

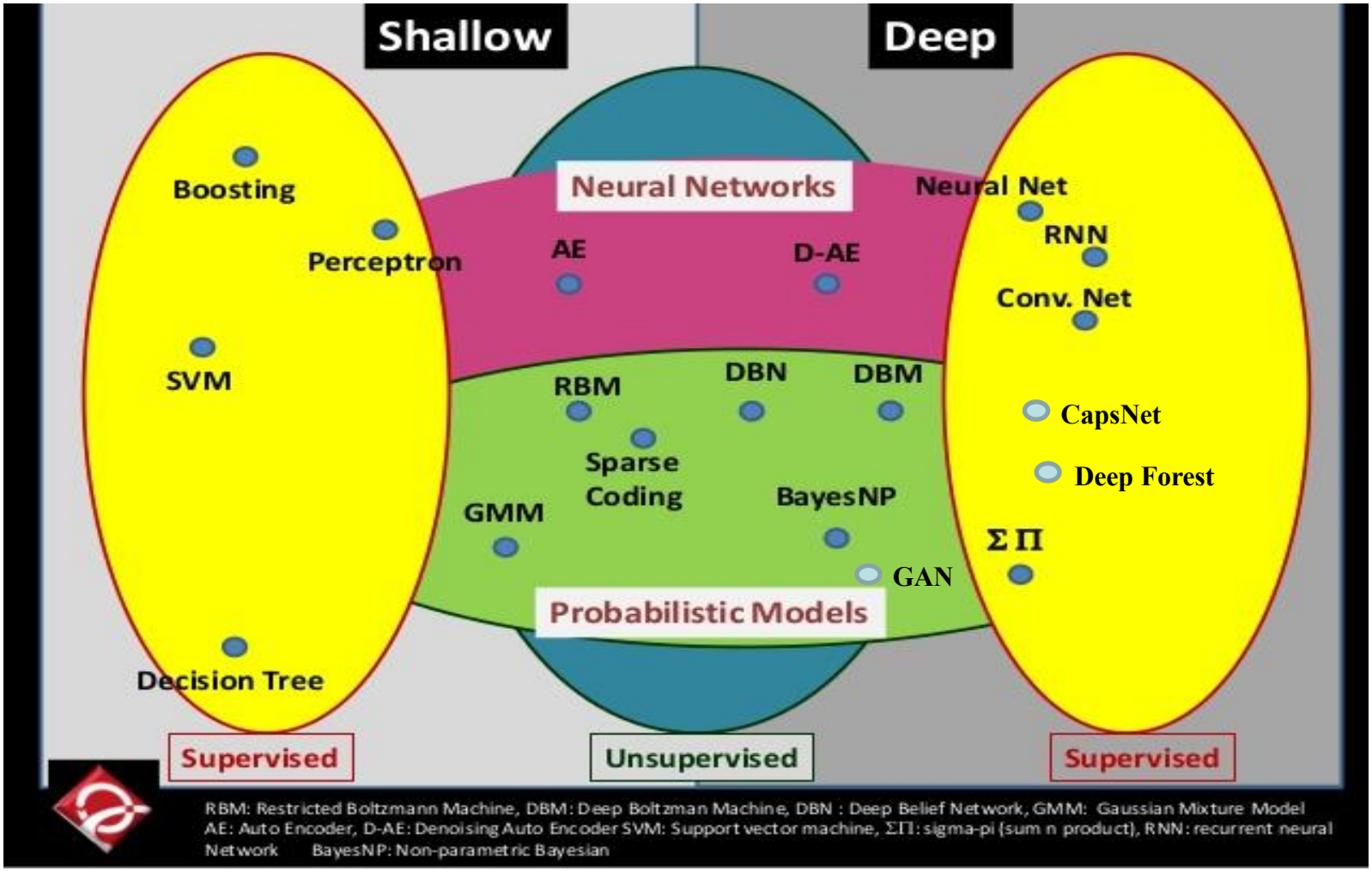
SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

提纲

- 深度生成模型概述
- Hopfield神经网络
- 玻尔兹曼机和受限玻尔兹曼机
- 深度玻尔兹曼机和深度信念网络
- 自编码器及其变种
- 中英文术语对照

1 深度生成模型概述

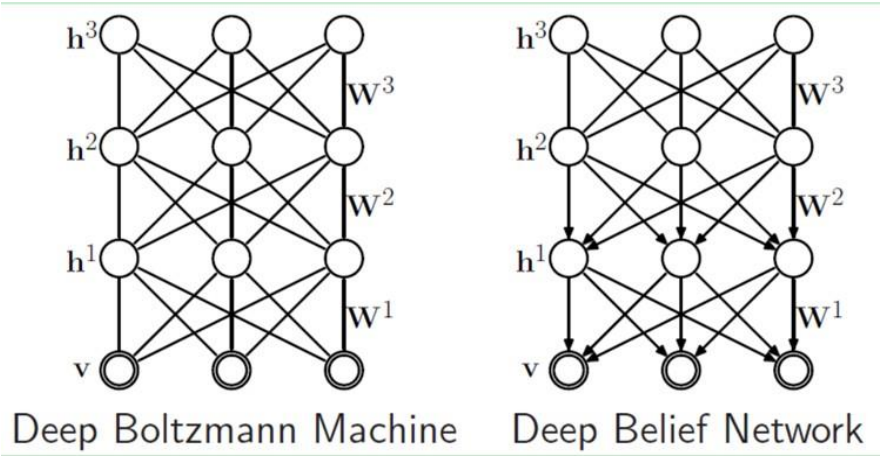
深度生成模型概述

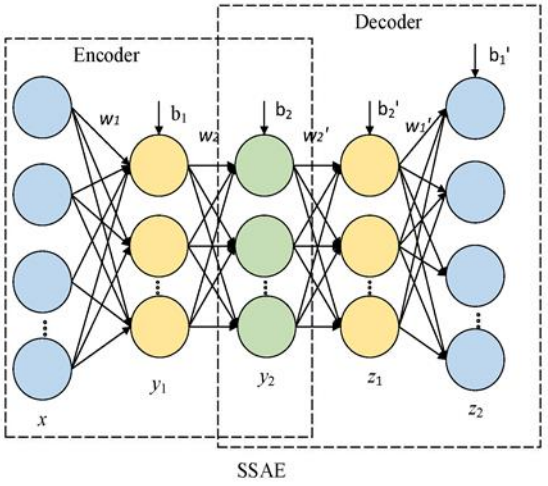

Shallow
Deep
Boosting
Perceptron
SVM
Decision Tree
Neural Networks
AE
D-AE
RBM
DBN
DBM
Sparse Coding
GMM
BayesNP
Probabilistic Models
Neural Net
RNN
Conv. Net
CapsNet
Deep Forest
ΣΠ
GAN
Supervised
Unsupervised
Supervised
RBM: Restricted Boltzmann Machine, DBM: Deep Boltzman Machine, DBN : Deep Belief Network, GMM: Gaussian Mixture Model
AE: Auto Encoder, D-AE: Denoising Auto Encoder SVM: Support vector machine, ΣΠ: sigma-pi (sum n product), RNN: recurrent neural Network BayesNP: Non-parametric Bayesian

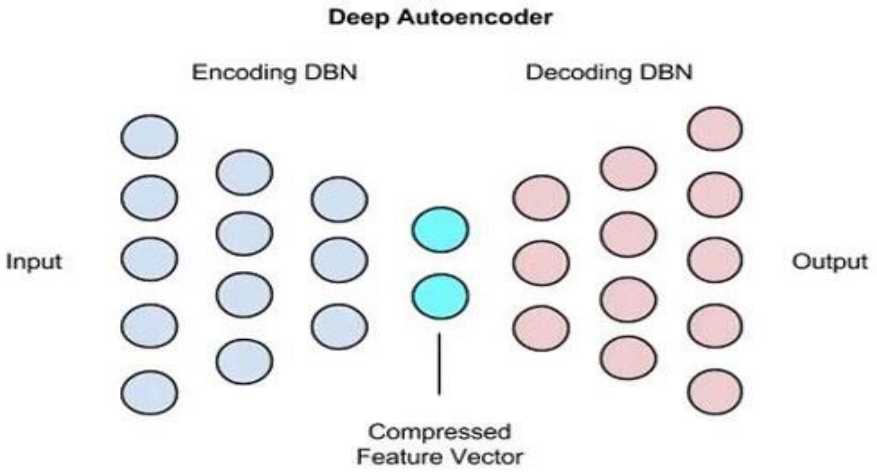
深度生成模型概述

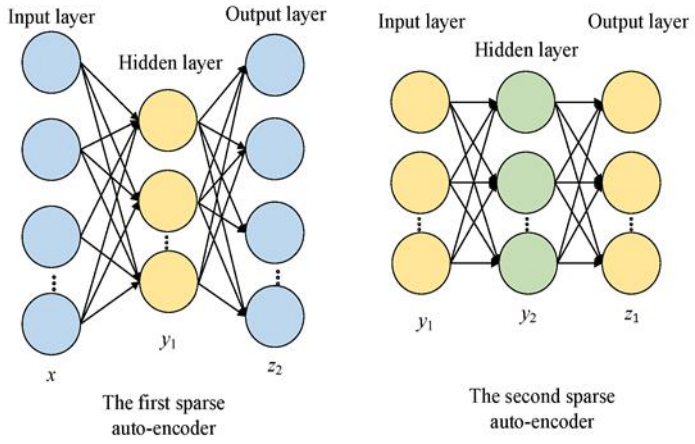
- 深度信念网络（Deep Belief Network, DBN）
- 深度玻尔兹曼机（Deep Boltzmann Machine, DBM）
- 深度自编码器（Deep Auto-Encoder, DAE）
- 降噪自编码器（Denoising Auto-Encoder, D-AE）
- 栈式自编码器（Stacked Auto-Encoder, SAE）
- 生成对抗网络（Generative Adversarial Networks，GAN）
- 非参数贝叶斯网络（Non-parametric Bayesian networks）

深度生成模型概述



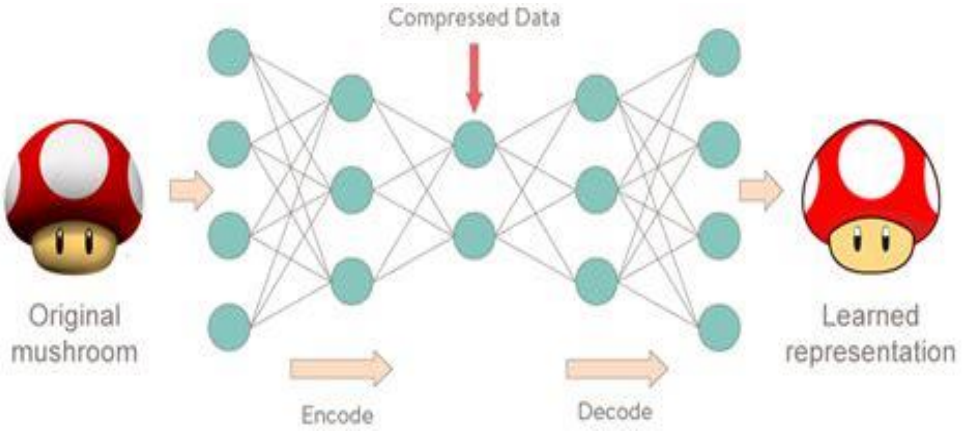




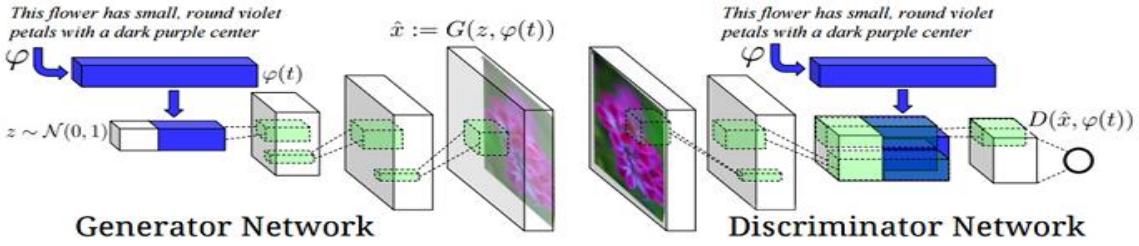


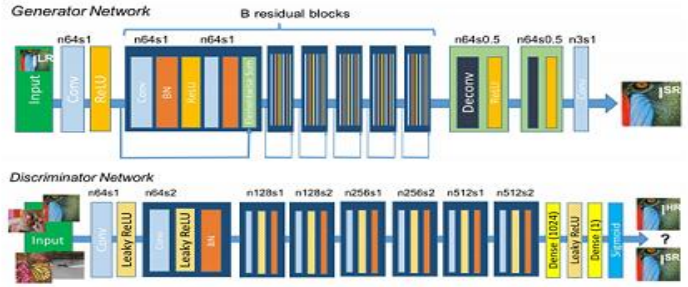
Stacked Auto-encoder

深度生成模型概述



Denoising Auto-encoder





深度生成模型概述

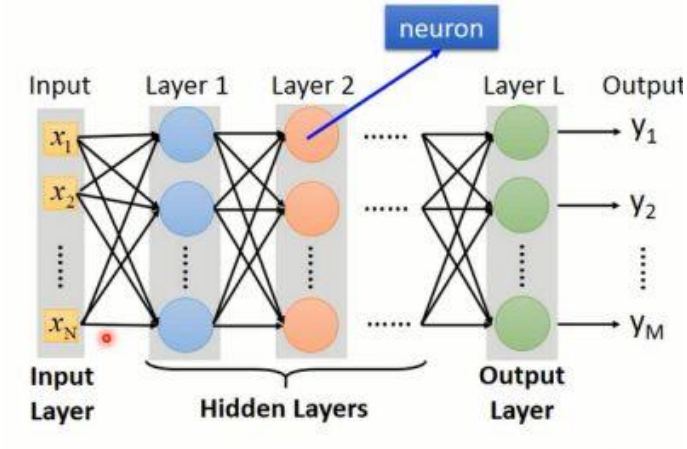
年份	事件	相关论文/Reference
1982	Hopfield网络提出	J. J . Hopfield . Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, Vol. 79 , No. 8, pp. 2554-2558, 1982
1983	玻尔兹曼机生成模型被提出	Fahlman, S. E., Hinton, G. E., & Sejnowski, T. J. (1983). Massively parallel architectures for Al: NETL, Thistle, and Boltzmann machines. Proceedings of AAAI-83109, 113
1992	Neal提出了sigmoid信念网络	Neal, R. M. (1992). Connectionist learning of belief networks. Artificial intelligence, 56(1), 71-113
2006	Hinton提出深度信念网络	Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural computation, 18(7), 1527-1554
2008	采用DBNs来完成回归任务	Hinton, G. E., & Salakhutdinov, R. R. (2008). Using deep belief nets to learn covariance kernels for Gaussian processes. In Advances in neural information processing systems (pp. 1249-1256)
2009	将卷积深度信念网络用于可视物体识别	Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009, June). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In Proceedings of the 26th annual international conference on machine learning (pp. 609-616). ACM
2012	DBNs应用在语音识别领域	Mohamed, A. R., Dahl, G. E., & Hinton, G. (2012). Acoustic modeling using deep belief networks. IEEE Transactions on Audio, Speech, and Language Processing, 20(1), 14-22
2014	Goodfellow提出生成对抗网络	Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680)
2016	Deepmind创建了一个深度生成模型Wavenet，可用来生成具有自然人声的语音	Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... & Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499

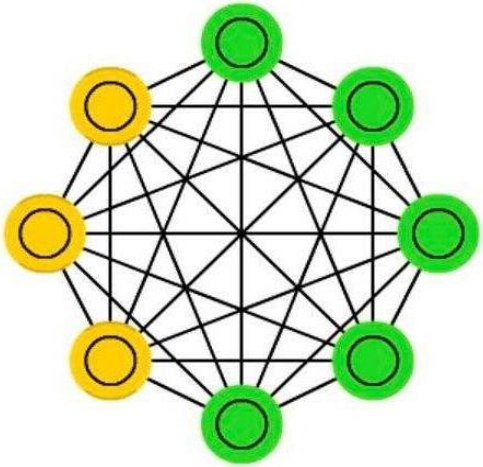
2 Hopfield神经网络

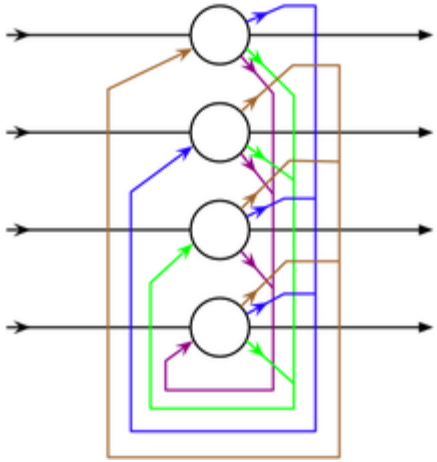
Hopfield神经网络

❑ 神经网络分类

- 多层神经网络：模式识别
- 相互连接型网络：通过联想记忆去除数据中的噪声





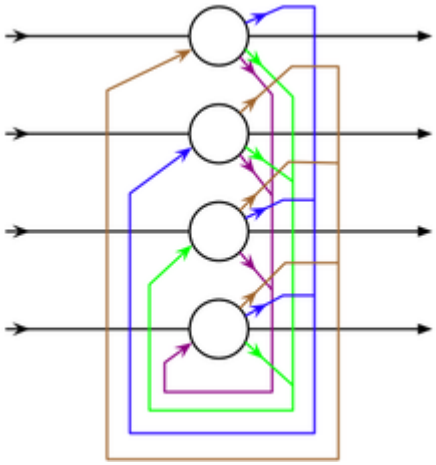


Hopfield神经网络

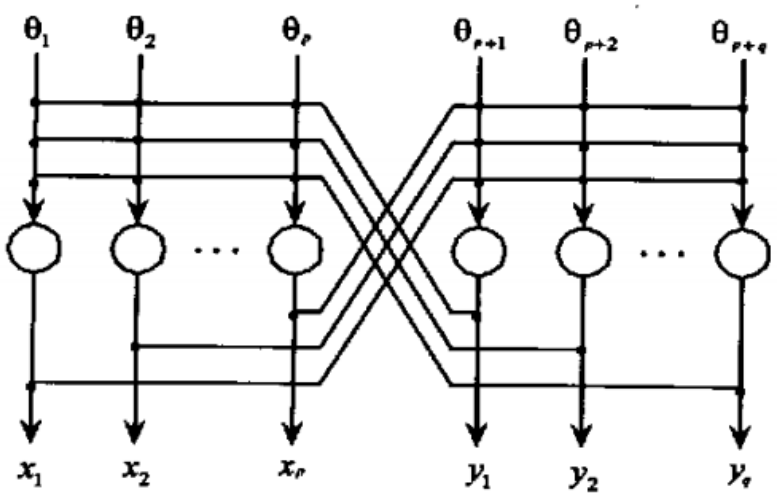
- 1982年提出的Hopfield神经网络是最典型的相互连结型网络
- Hopfield网络的优点
  - 单元之间的连接权重对称（$w_{ij}=w_{ji}$）
  - 每个单元没有到自身的连接（$w_{ii}=0$）
  - 单元的状态采用随机异步更新方式，每次只有一个单元改变状态
  - n个二值单元做成的二值神经网络，每个单元的输出只能是0或1的两个值

Hopfield神经网络

- 联想记忆就是当输入模式为某种状态时，输出端要给出与之相应的输出模式
  - 如果输入模式与输出模式一致，称为自联想记忆，否则称为异联想记忆



自联想记忆



异联想记忆

输入：$x_1,x_2,\ldots,x_p$

输出：$y_1,y_2,\ldots,y_q$

Hopfield神经网络

❑ Hopfield神经网络计算

– 假设有n个单元组成的Hopfield神经网络，第 i 个单元在 t 时刻的输入记作$u_i(t)$，输出记作$x_i(t)$，连接权重为w_{ij}，阈值为$b_i(t)$，则 $t+1$ 时刻i单元的输出$x_i(t+1)$可表示为:

$$x_i(t+1)=\begin{cases}1, & u_i(t)>0\\ x_i(t), & u_i(t)=0\\ 0, & u_i(t)<0\end{cases}$$

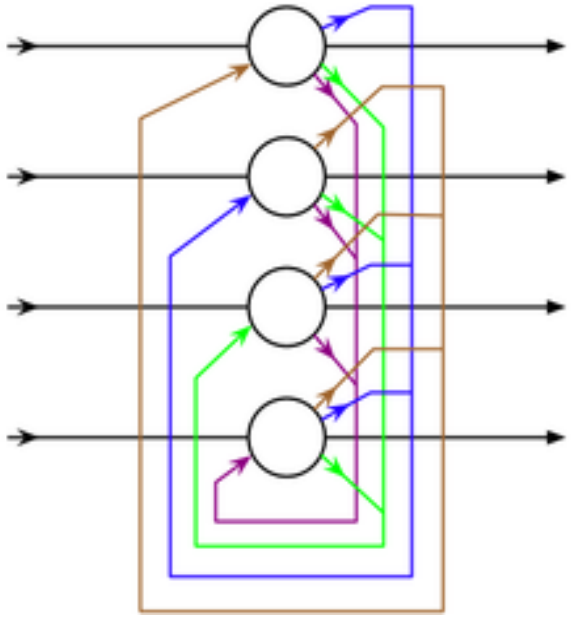
权重总和大于阈值，取1
权重总和小于阈值，取0

$$u_i(t)=\sum_{j=1}^{n} w_{ij}x_j(t)-b_i(t)$$

来自于其他单元的输入$x_j(t)$的权重总和

Hopfield神经网络

- 在Hopfield神经网络中，每个时刻都只有一个随机选择的单元会发生状态变化
- 对于一个由n个单元组成的网络，如果要完成全部单元的状态变化，至少需要n个时刻
  - 单元的状态变化会一直进行下去，直到网络达到稳定状态。各单元的最终状态就是输出模式



Hopfield神经网络

- 根据输入模式联想输出模式时，需要事先确定连接权重w_{ij}，而连接权重w_{ij}要对输入模式的训练样本进行训练后才能确定
- 和多层神经网络一样，一次训练并不能确定连接权重，而是要不断重复这个过程，直到满足终止判断条件，而这个指标就是Hopfield神经网络的**能量函数 E**

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_i x_j + \sum_{i=1}^{n} b_i x_i$$

  - 当输入模式与输出模式一致时，能量函数 E 的结果是0

Hopfield神经网络

- 根据前面定义的状态变化规则改变网络状态时，上式中定义的能量函数 E 总是非递增的，即随时间的不断增加而逐渐减小，直到网络达到稳定状态为止
  - 能量函数分解成单元k的能量函数和 k 以外的单元的能量函数

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_i x_j + \sum_{i=1}^{n} b_i x_i$$

$$E = -\frac{1}{2}\left(x_k \sum_{j=1}^{n} w_{kj}x_j + \sum_{i\neq k}^{n}\sum_{j=1}^{n} w_{ij}x_i x_j \right) + b_k x_k + \sum_{i\neq k}^{n} b_k x_k$$

$$= -\frac{1}{2}\left(x_k \left(w_{kk}x_k + \sum_{j\neq k}^{n} w_{kj}x_j \right) + \sum_{i\neq k}^{n}\left(w_{ik}x_i x_k + \sum_{j\neq k}^{n} w_{ij}x_i x_j \right) \right) + b_k x_k + \sum_{i\neq k}^{n} b_i x_i$$

$$= -\frac{1}{2}\left(w_{kk}x_k^2 + x_k \sum_{j\neq k}^{n} w_{kj}x_j + \sum_{i\neq k}^{n}\left(w_{ik}x_i x_k + \sum_{j\neq k}^{n} w_{ij}x_i x_j \right) \right) + b_k x_k + \sum_{i\neq k}^{n} b_i x_i$$

$$= -\frac{1}{2}\sum_{i\neq k}^{n}\sum_{j\neq k}^{n} w_{ij}x_i x_j + \sum_{i\neq k}^{n} b_i x_i - \frac{1}{2}\left(\sum_{j\neq k}^{n} w_{kj}x_j + \sum_{i\neq k}^{n} w_{ik}x_i \right) x_k + b_k x_k$$

Hopfield神经网络

- 根据前面定义的状态变化规则改变网络状态时，上式中定义的能量函数 E 总是非递增的，即随时间的不断增加而逐渐减小，直到网络达到稳定状态为止
  - 能量函数分解成单元k的能量函数和 k 以外的单元的能量函数

$$E = -\frac{1}{2}\sum_{i\neq k}^{n}\sum_{j\neq k}^{n} w_{ij}x_i x_j + \sum_{i\neq k}^{n} b_i x_i - \frac{1}{2}\left(\sum_{j\neq k}^{n} w_{kj}x_j + \sum_{i\neq k}^{n} w_{ik}x_i\right)x_k + b_k x_k$$

$$\Delta E_k = -\frac{1}{2}\left(\sum_{j\neq k}^{n} w_{kj}x_j + \sum_{i\neq k}^{n} w_{ik}x_i\right)\Delta x_k + b_k \Delta x_k \qquad \Delta x_k = x_k(t+1) - x_k(t)$$

$$\Delta E_k = -\left(\sum_{j\neq k}^{n} w_{kj}x_j - b_k\right)\Delta x_k = -u_k \Delta x_k$$

当$\Delta x_k > 0$时，x_k由0变成1，即$u_k(t) > 0$

$\Delta E_k(t) < 0$

当$\Delta x_k < 0$时，x_k由1变成0，即$u_k(t) < 0$

$\Delta E_k(t) < 0$

$$x_k(t+1) = \begin{cases} 1, & u_k(t) > 0 \\ x_k(t), & u_k(t) = 0 \\ 0, & u_k(t) < 0 \end{cases}$$

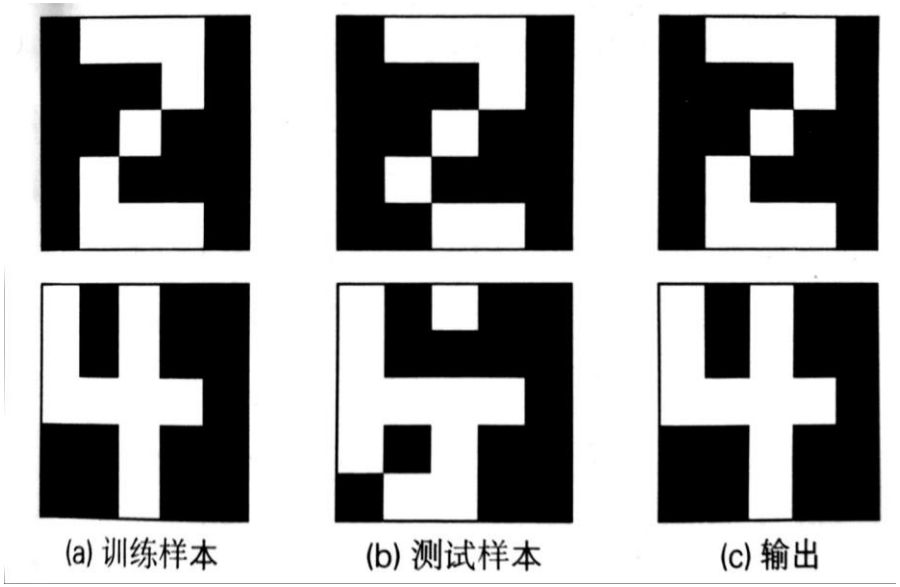
Hopfield神经网络

- 根据前面定义的状态变化规则改变网络状态时，上式中定义的能量函数 E 总是非递增的，即随时间的不断增加而逐渐减小，直到网络达到稳定状态为止
  - 能量函数分解成单元k的能量函数和 k 以外的单元的能量函数

$$u_i(t+1)=\sum_{j=1}^{n} w_{ij}x_j(t)-b_i(t) \qquad E=-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_ix_j+\sum_{i=1}^{n} b_ix_i$$

$$u_i(t+1)=\sum_{j=1}^{n} w_{ij}x_j(t)+b_i(t) \qquad E=-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_ix_j-\sum_{i=1}^{n} b_ix_i$$

Hopfield神经网络



记住训练样本的输入

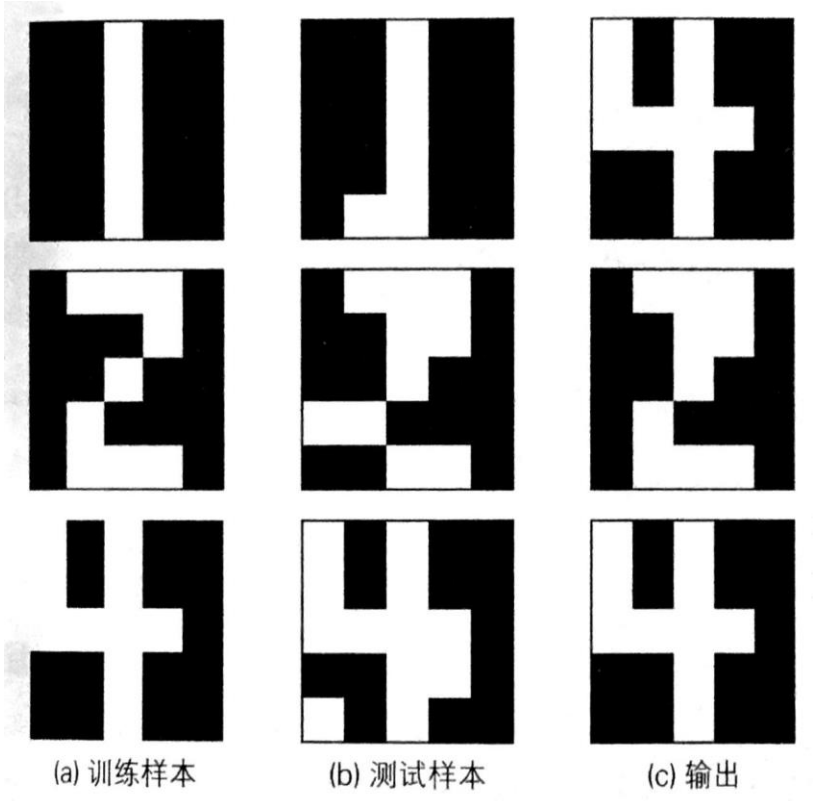
自联想记忆

对测试样本去噪

Hopfield网络的应用

Hopfield神经网络

- 当需要记忆的模式之间的较为相似，或者需要记忆的模式太多时，Hopfield神经网络就不能正确地辨别模式。这种相互干扰、不能准确记忆的情况称为串扰(crosstalk)



Hopfield网络的串扰

由于“4”中包含了“1”中的全部竖线，Hopfield网络错把带有“横线”噪音的测试样本“1”识别为“4”，这种情况即串扰

Hopfield神经网络

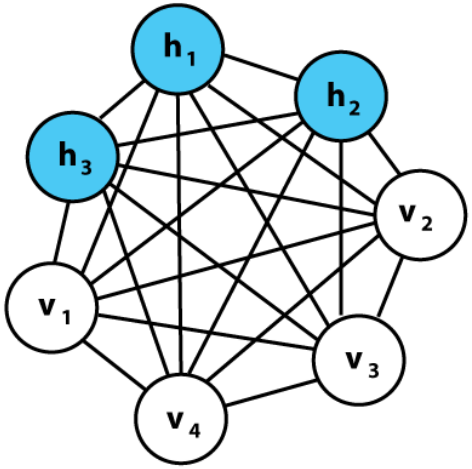
- Hopfield神经网络能够记忆的模式数量有限，大约是网络单元数的15%左右，为了防止串扰，可以采用先把模式的正交化再进行记忆等方法
- 但是正交化方法并不能完全解决问题，玻尔兹曼机可以解决这一问题

3

玻尔兹曼机及受限玻尔兹曼机

玻尔兹曼机

- **玻尔兹曼机也是相互连接型网络**
  - 如果发生串扰或陷入局部最优解，Hopfield神经网络就不能正确的辨别模式。而玻尔兹曼机(Boltzmann Machine)则可以通过让每个单元按照一定的概率分布发生状态变化，来避免陷入局部最优解
- **玻尔兹曼机保持了Hopfield神经网络的假设：**
  - 权重对称
  - 自身无连接
  - 二值输出



玻尔兹曼机

玻尔兹曼机

❑ 玻尔兹曼机的输出是按照某种概率分布决定的

$$\begin{cases} p(x_i = 1|u_i) = \dfrac{exp\left(\dfrac{x}{kT}\right)}{1 + exp\left(\dfrac{x}{kT}\right)} \\ p(x_i = 0|u_i) = \dfrac{1}{1 + exp\left(\dfrac{x}{kT}\right)} \end{cases}$$

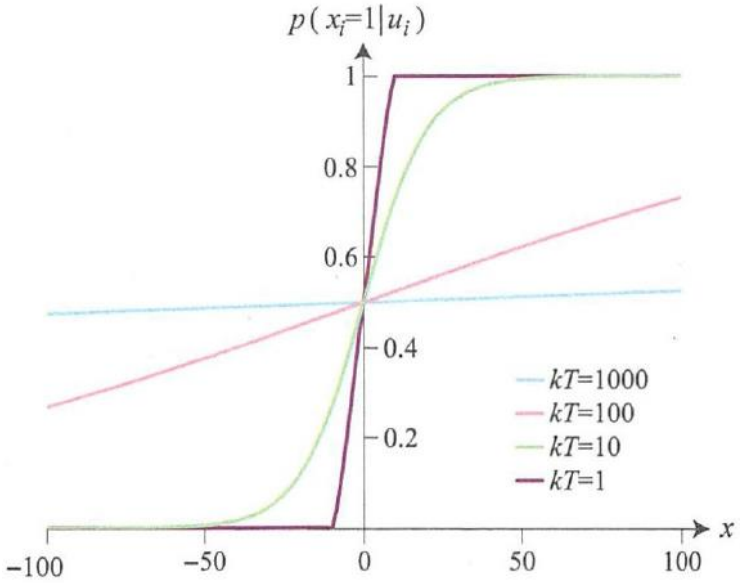
Hopfield网络的输出

$$x_i(t+1) = \begin{cases} 1, & u_i(t) > 0 \\ x_i(t), & u_i(t) = 0 \\ 0, & u_i(t) < 0 \end{cases}$$

- $T(>0)$表示温度系数，当 T 趋近于无穷时，无论u_i取值如何，x_i等于 1 或 0 的概率都是1/2，这种状态称为稳定状态
- k是玻尔兹曼常数

玻尔兹曼机

- 玻尔兹曼机的输出是按照某种概率分布决定的
  - $T(>0)$表示温度系数，当 T 趋近于无穷时，无论u_i取值如何，x_i等于 1 或 0 的概率都是 1/2，这种状态称为稳定状态
  - k是玻尔兹曼常数



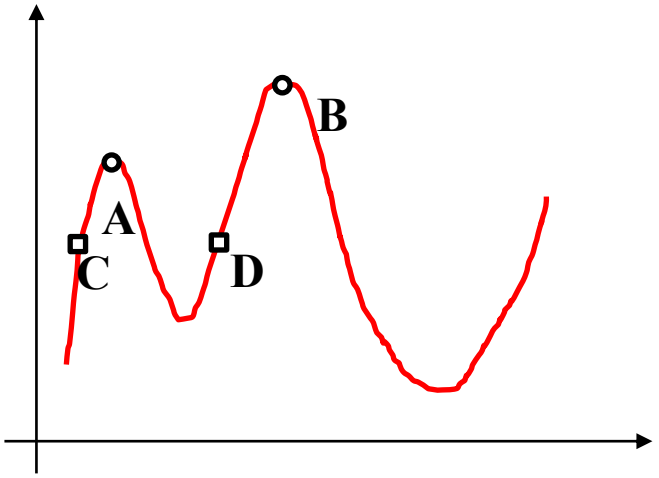
温度系数引起的概率变化

玻尔兹曼机

- 玻尔兹曼机选择模拟退火算法，可以先采用较大的温度系数及进行粗调，然后逐渐减小温度系数进行微调
  - 温度系数越大，跳出局部最优解的概率越高。但是温度系数增大时，获得能量函数极小值的概率就会降低
  - 反之，温度系数减小时，虽然获得能量函数极小值的概率增加了，但是玻尔兹曼机需要经历较长时间才能达到稳定状态

玻尔兹曼机

❑ 模拟退火算法



模拟退火是一种贪心算法，但是它的搜索过程引入了随机因素，以一定的概率来接受一个比当前解要差的解，因此有可能会跳出这个局部的最优解，达到全局的最优解，而且这个概率随着时间推移逐渐降低(逐渐降低才能趋向稳定)

玻尔兹曼机

❑ 模拟退火算法

– 示例: 求函数$F(x)=6x^2+8x^5-xy$的最小值, $x\in[0,100]$

```
while( T > Tmin )
{
    for (int i = 0; i < k; i++)
    {
        y_current =F(x[i]); // x[i]中存在初始化的值
        x_new = x[i] + random();
        if (x_new >= 0 && x_new <= 100)
        {
            y_new =F(x_new);
            if(y_new < y_current)
                x[i] = x_new;
            else
            {
                p = 1/(1+e^(-(y_new-y_current)/T));
                if (random() > p)
                    x[i] = x_new;
            }
        }
    }
    T = T * delta; // delta<1, 温度下降速率
}
for (i = 0; i < k; i++) result = min(result, F(x[i]));
```

玻尔兹曼机

❑ 玻尔兹曼机的训练过程

- 1. 训练准备：初始化连接权重w_{ij}和偏置b_i
- 2. 调整参数
- 2.1 选取一个单元 i ,求u_i
- 2.2 根据u_i的值，计算输出x_i
- 2.3 根据输出x_i和x_j的值，调整连接权重w_{ij}和偏置b_i
- 重复步骤2，直到满足终止判断条件

$$u_i(t+1)=\sum_{j=1}^{n} w_{ij}x_j(t)+b_i(t)$$

$$\begin{cases} p(x_i=1|u_i)=\dfrac{exp\left(\dfrac{x}{kT}\right)}{1+exp\left(\dfrac{x}{kT}\right)} \\ p(x_i=0|u_i)=\dfrac{1}{1+exp\left(\dfrac{x}{kT}\right)} \end{cases}$$

玻尔兹曼机

- **具体而言，当初始化连接权重后，选取一个单元 i**
  - 1.计算单元激活值u_i

$$u_i(t) = \sum_{j=1}^{n} w_{ij} x_j(t) + b_i(t)$$

$$x_i(t+1) = \begin{cases} 1, & u_i(t) > 0 \\ x_i(t), & u_i(t) = 0 \\ 0, & u_i(t) < 0 \end{cases}$$

2.计算x_i等于 1 或 0 的概率

$$\begin{cases} p(x_i = 1|u_i) = \dfrac{exp\left(\frac{x}{kT}\right)}{1 + exp\left(\frac{x}{kT}\right)} \\ p(x_i = 0|u_i) = \dfrac{1}{1 + exp\left(\frac{x}{kT}\right)} \end{cases}$$

根据概率x_i等于1或0的概率，调整x_i的取值。一般是随机产生一个(0,1)之间的随机数λ，如果p>λ，确认状态改变，否则不改变

不能将计算所得概率直接作为x_i的值，而是作为概率来决定x_i的值

玻尔兹曼机

- 调整连接权重w_{ij}和偏置b_i，这里用似然函数$L(\theta)$导出调整值，θ表示所有的连接权重和偏置

$$L(\theta) = \prod_{n=1}^{N} p(x_n|\theta)$$

  - 其中，概率分布的定义如下，E表示能量函数，$Z(\theta)$是归一化常数

$$p(x_n|\theta) = \frac{1}{Z(\theta)} exp\{-E(x,\theta)\}$$

$$Z(\theta) = \sum_{x} exp\{-E(x,\theta)\}$$

玻尔兹曼机

❑ 通常，使用对数似然函数求解

$$\ln L(\theta) = \sum_{n=1}^{N} \ln p(x_n|\theta)$$

➢ 当对数似然函数的梯度为0时，就可以得到最大似然估计量，即通过求连接权重w_{ij}和偏置b_i的相关梯度，可以求出调整值

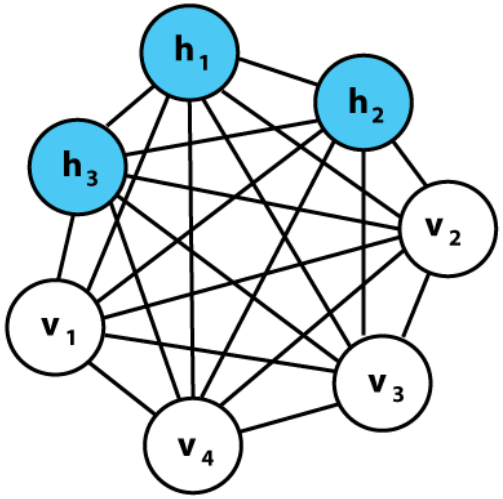
玻尔兹曼机

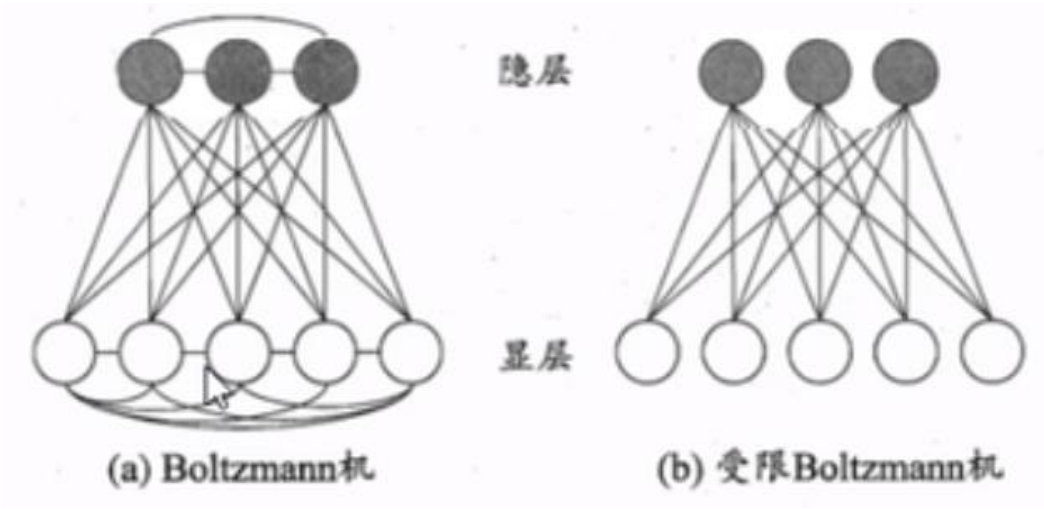
❑ 求解困难

- 似然函数是基于所有单元组合来计算的，所以单元数过多将导致组合数异常庞大，无法进行实时计算。为了解决这个问题，人们提出了一种近似算法，对比散度算法（稍后介绍）

受限玻尔兹曼机

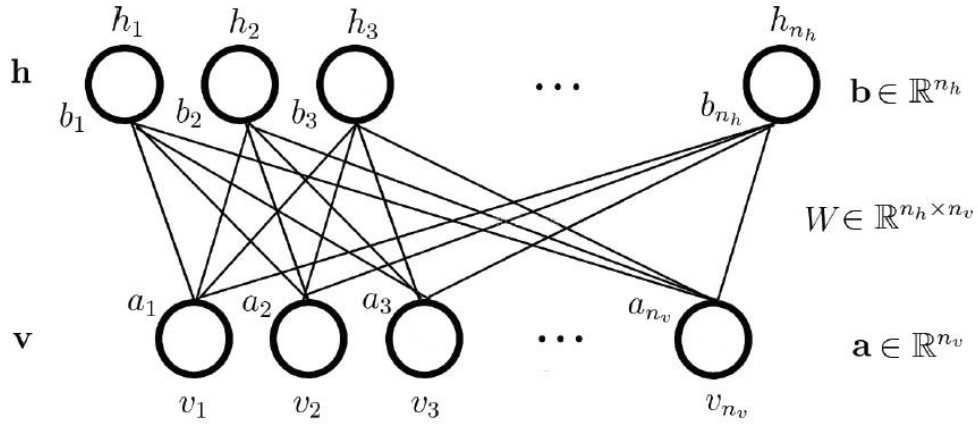
- 前面介绍的玻尔兹曼机默认了所有单元都为可见单元，在实际应用上，玻尔兹曼机还可以由可见单元和隐藏单元共同构成
  - 隐藏单元与输入数据没有直接联系，但会影响可见单元的概率。假设可见单元为可见变量 v，隐藏单元为隐藏变量 h。玻尔兹曼机含有隐藏变量时，概率分布仍然与前面计算的结果相同





受限玻尔兹曼机

- **含有隐藏变量的波尔兹曼机训练非常困难，所以Hinton等人提出了受限玻尔兹曼机（Restricted Boltzmann Machine）**
  - 由可见层和隐藏层构成
  - 层内单元之间无连接
  - 信息可双向流动



- n_v,n_h分别表示可见层和隐藏层中包含的神经元数目，下标v和h代表visible和hidden
- $\mathbf{v}=(v_1,v_2,\ldots,v_{n_v})^T$: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
- $\mathbf{h}=(h_1,h_2,\ldots,h_{n_h})^T$: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
- $\mathbf{a}=(a_1,a_2,\ldots,a_{n_v})^T$: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
- $\mathbf{b}=(b_1,b_2,\ldots,b_{n_h})^T$: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
- $W=(w_{ij})^T$:隐藏层和可见层之间的权值矩阵，w_{ij}表示隐藏层中第i个神经元和可见层中第j个神经元之间的权重

受限玻尔兹曼机

❑ 受限玻尔兹曼机的能量函数为：

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

$$E(\mathbf{v},\mathbf{h},\theta)=-\mathbf{a}^T\mathbf{v}-\mathbf{b}^T\mathbf{h}-\mathbf{h}^T W\mathbf{v}$$

其中，a_i是可见变量的偏置，b_j是隐藏变量的偏置，w_{ij}是连接权重，$\theta=(W,\mathbf{a},\mathbf{b})$是表示所有连接权重和偏置的参数集合。

受限玻尔兹曼机

- 利用能量函数，可以给出状态(v,h)的联合概率分布

$$P(\mathbf{v},\mathbf{h}|\theta)=\frac{1}{Z_\theta}exp\{-E(\mathbf{v},\mathbf{h},\theta)\}$$

$$Z_\theta=\sum_{\mathbf{v},\mathbf{h}}exp\{-E(\mathbf{v},\mathbf{h},\theta)\}$$

受限玻尔兹曼机

- 观测数据S={$\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^{n_s}$}的概率分布，也就是可见层状态变量的概率分布

$$P(\mathbf{v}|\theta) = \sum_{\mathbf{h}} P(\mathbf{v}, \mathbf{h}|\theta) = \frac{1}{Z_\theta} \sum_{\mathbf{h}} exp\{-E(\mathbf{v}, \mathbf{h}, \theta)\}$$

- 类似地，有关于隐藏层状态变量的概率分布

$$P(\mathbf{h}|\theta) = \sum_{\mathbf{v}} P(\mathbf{v}, \mathbf{h}|\theta) = \frac{1}{Z_\theta} \sum_{\mathbf{v}} exp\{-E(\mathbf{v}, \mathbf{h}, \theta\}$$

受限玻尔兹曼机

- 给定训练样本S={$\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^{n_s}$}，调整参数$\theta$训练RBM
  - $\mathbf{v}^i=(v_1^i, v_2^i, \cdots, v_{n_v}^i)^T$, i=1,2,…,n_s, 它们是独立同分布的，训练RBM是最大化似然函数

$$L_{\theta,S} = \prod_{i=1}^{n_s} P(\mathbf{v}^i|\theta)$$

- 和玻尔兹曼机一样，计算时通常使用对数似然函数

$$\ln L_{\theta,S} = \ln \prod_{i=1}^{n_s} P(\mathbf{v}^i|\theta) = \sum_{i=1}^{n_s} \ln P(\mathbf{v}^i)$$

最大化时就沿着梯度的正方向更新 $\theta := \theta + \lambda \frac{\partial \ln L_{\theta,S}}{\partial \theta}$

受限玻尔兹曼机

❑ 梯度求解

$$\frac{\partial \ln L_{\theta,S}}{\partial \theta} = -\sum_{\mathbf{h}} \left(P(\mathbf{h}|\mathbf{v}^i) \frac{\partial E(\mathbf{v}^i, \mathbf{h})}{\partial \theta} \right) + \sum_{\mathbf{v},\mathbf{h}} \left(P(\mathbf{v}, \mathbf{h}) \frac{\partial E(\mathbf{v}, \mathbf{h})}{\partial \theta} \right)$$

$$\frac{\partial \ln L_{\theta,S}}{\partial w_{ij}} = P(h_j = 1|\mathbf{v}^i) v_i^i - \sum_{\mathbf{v}} P(\mathbf{v}) P(h_j = 1|\mathbf{v})\, v_i \qquad w_{ij} \leftarrow w_{ij} + \lambda \frac{\partial \ln L_{\theta,S}}{\partial w_{ij}}$$

$$\frac{\partial \ln L_{\theta,S}}{\partial a_i} = v_i - \sum_{\mathbf{v}} P(\mathbf{v})\, v_i \qquad a_i \leftarrow a_i + \lambda \frac{\partial \ln L_{\theta,S}}{\partial a_i}$$

$$\frac{\partial \ln L_{\theta,S}}{\partial b_i} = P(h_i = 1|\mathbf{v}^i) + \sum_{\mathbf{v}} P(\mathbf{v}) P(h_i = 1|\mathbf{v}) \qquad b_i \leftarrow b_i + \lambda \frac{\partial \ln L_{\theta,S}}{\partial b_i}$$

多个样本再求和即可

受限玻尔兹曼机

- 改良后的受限玻尔兹曼机依然在计算上存在着问题：
  $\sum_v P(v)$是所有输入模式的总和，不可避免会产生庞大的计算量

- 要想解决这个问题，可以使用**Gibbs采样**（**Gibbs Sampling**）算法进行迭代计算求近似解。但即使这样处理，迭代次数也仍然非常多。人们提出了**对比散度算法**

对比散度算法

❑ 对比散度法

- 2002年Hinton提出, MCMC的状态以训练样本为起点，这样只需很少的状态转移就可以得到RBM的分布

❑ 对比散度算法的训练过程

- 1. 训练准备：初始化连接权重和偏置
- 2. 调整参数
  - 2.1 在可见层$v^{(0)}$设置输入模式
  - 2.2 调整隐藏层中单元$h^{(0)}$的值
  - 2.3根据输出x_i和x_j的值，调整连接权重w_{ij}、偏置a_i、偏置b_j
- 重复步骤2，直到满足终止判断条件

Hinton, Geoffrey E. "Training products of experts by minimizing contrastive divergence." Neural computation 14, no. 8 (2002): 1771-1800.

对比散度算法

❑ 初始化连接权重和偏置

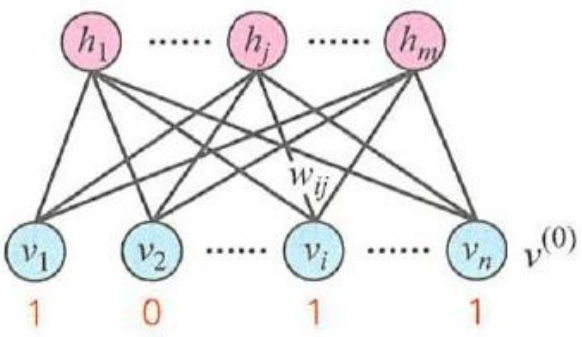
– 随机数初始化

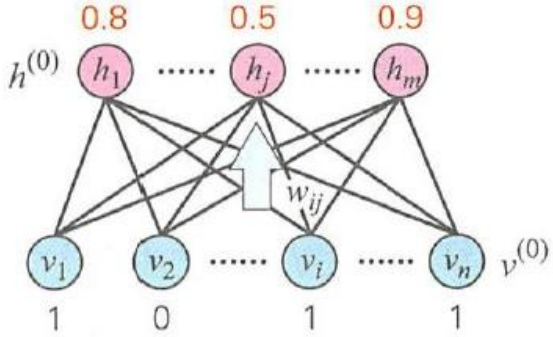
❑ 调整参数

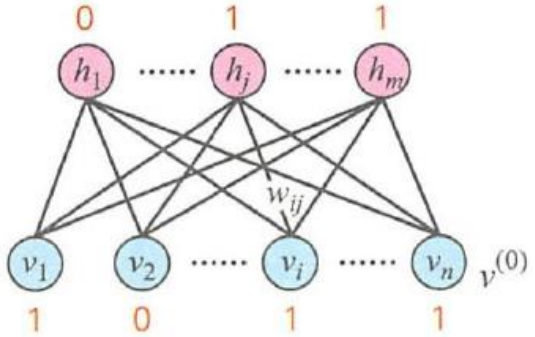
– 在可见层设初值$\mathbf{v}^{(0)}$，根据参数初始值计算隐藏层为状态1的概率

$$P\left(h_j^{(0)}=1|\mathbf{v}^{(0)}\right)=\sigma\left(b_j+\sum_{i=1}^{n_v}w_{ij}v_i^{(0)}\right)$$

根据这个概率计算符合二项分布的隐藏层中单元$h_j^{(0)}$的状态，其中σ表示*sigmoid*函数







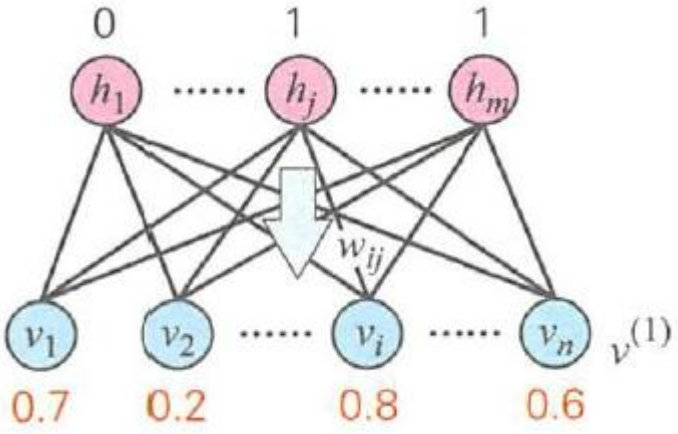
对比散度算法

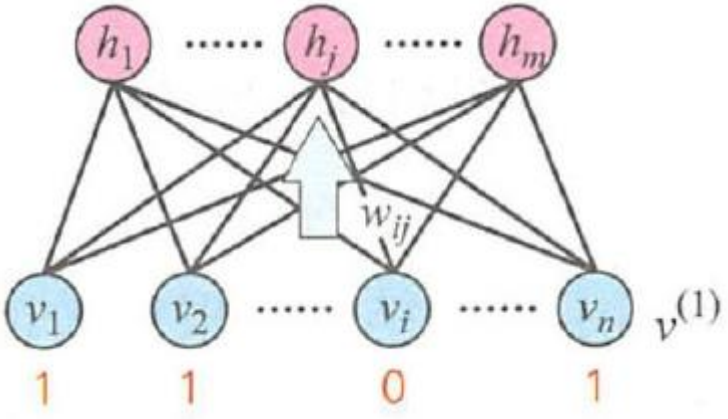
❑ 调整参数

– 根据隐藏层的状态，计算在可见层各单元状态为1的概率

$$P\left(v_i^{(1)}=1|\mathbf{h}^{(0)}\right)=\sigma\left(a_i+\sum_{j=1}^{n_h}w_{ij}h_j^{(0)}\right)$$

根据这个概率计算各可见单元$v_i^{(1)}$的状态





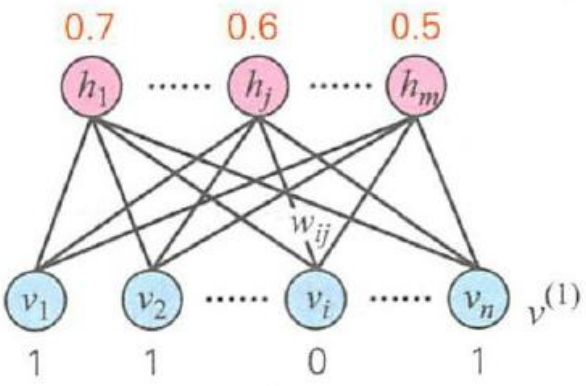
对比散度算法

❑ 调整参数

– 再次根据可见层的值$\mathbf{v}^{(1)}$计算隐层单元$h_j^{(1)}$状态为1的概率

$$P\left(h_j^{(1)}=1|\mathbf{v}^{(1)}\right)=\sigma\left(b_j+\sum_{i=1}^{n_v}w_{ij}v_i^{(1)}\right)$$

– 此时可以利用$P\left(h_j^{(1)}=1|\mathbf{v}^{(1)}\right)$和$v_i^{(1)}$进行参数更新，当然上述过程还可以继续进行$k$次($k$步的Gibbs采样)，称为$k$步对比散度法，记做CD-$k$



对比散度算法

❑ 调整参数

– 连接权重w_{ij}、偏置a_i、偏置b_j 的调整值分别为

$$w_{ij} \leftarrow w_{ij} + \lambda\left(P\left(h_j^{(0)}=1|\mathbf{v}^{(0)}\right)v_i^{(0)} - P\left(h_j^{(1)}=1|\mathbf{v}^{(1)}\right)v_i^{(1)}\right)$$

$$a_i \leftarrow a_i + \lambda\left(v_i^{(0)} - v_i^{(1)}\right)$$

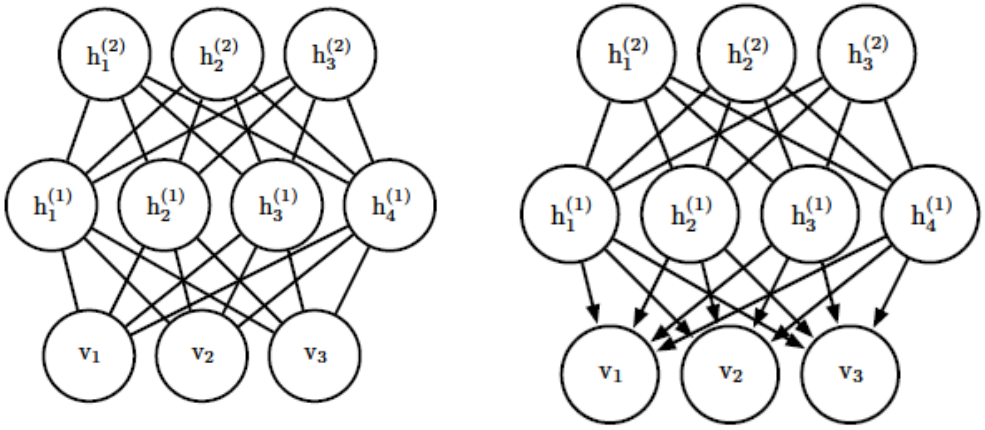
$$b_j \leftarrow b_j + \lambda\left(P\left(h_j^{(0)}=1|\mathbf{v}^{(0)}\right) - P\left(h_j^{(1)}=1|\mathbf{v}^{(1)}\right)\right)$$

4

深度玻尔兹曼机和深度信念网络

玻尔兹曼机的变体

- **深度玻尔兹曼机(Deep Boltzmann Machine, DBM)**
  - 由受限玻尔兹曼机堆叠组成
- **深度信念网络(Deep Belief Network, DBN)**
  - The top two layers have undirected, symmetric connections between them and form an associative memory. The lower layers receive top-down, directed connections from the layer above. The states of the units in the lowest layer represent a data vector.

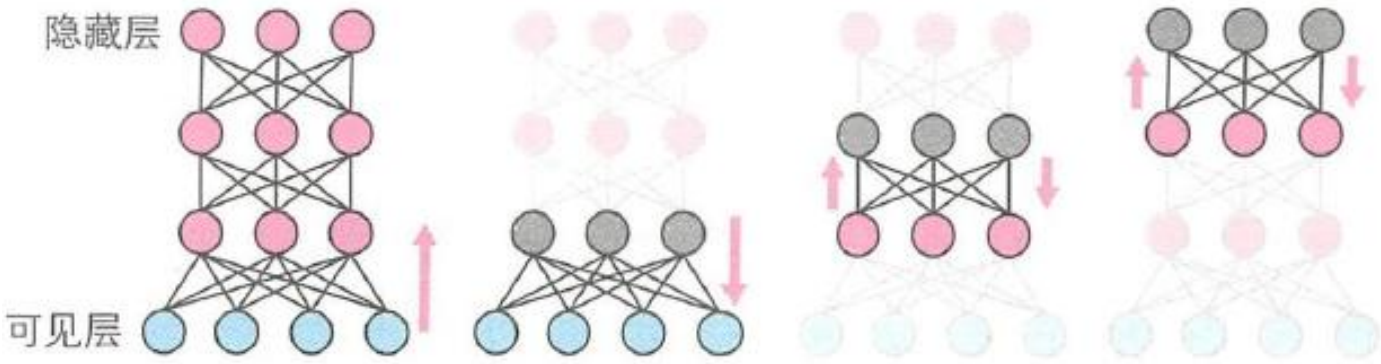


[1] Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." Neural computation 18, no. 7 (2006): 1527-1554.

[2] Hinton GE. Deep belief networks. Scholarpedia. 2009 May 31;4(5):5947.

深度玻尔兹曼机

- 深度玻尔兹曼机采用与多层神经网络不同的训练方法，在训练时采用对比散度算法，逐层来调整连接权重和偏置
- 具体做法：
  - 首先训练输入层和隐藏层之间的参数，把训练后得到的参数作为下一层的输入
  - 再调整该层与下一个隐藏层之间的参数
  - 然后逐次迭代，完成多层网络的训练

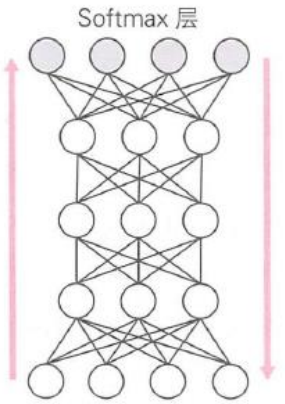


深度玻尔兹曼机

- 深度玻尔兹曼机既可以当作生成模型，也可以当作判别模型
  - 作为生成模型使用时，网络会按照某种概率分布生成训练数据。概率分布可根据训练样本导出，但是覆盖全部数据模式的概率分布很难导出，所以通常选择最大似然估计法训练参数，得到最能覆盖训练样本的概率分布
  - 这种生成模型能够：去除输入数据中含有的噪声，得到新的数据，对输入数据压缩和特征表达

深度玻尔兹曼机

- 作为判别模型使用时，需要在模型顶层添加一层Softmax实现分类
  - 进行分类时，需要同时提供训练样本和期望输出，在最顶层级联一个$Softmax$层
- 训练方法
  - 除最顶层外，其他各层都可以使用无监督学习进行训练。
  - 把训练得到的参数作为初始值，使用误差反向传播算法对包含最顶层的神经网络进行训练
  - 最顶层的参数使用随机数进行初始化



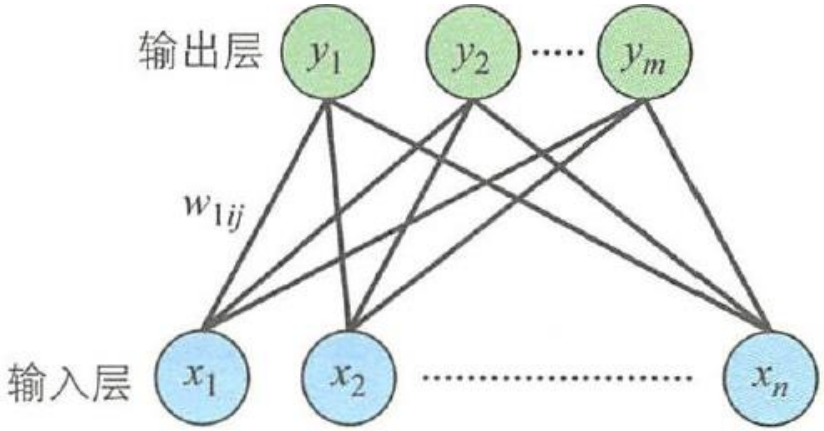
5 自编码器及其变种

自编码器

- **自编码器(Autoencoder)：是一种有效的数据维度压缩算法，主要应用在以下两个方面：**
  - 构建一种能够重构输入样本并进行特征表达的神经网络
    - 特征表达：是指对于分类会发生变动的不稳定模式，例如手写字体识别中由于不同人的书写习惯和风格的不同造成字符模式不稳定，或者输入样本中包含噪声等情况，神经网络也能将其转换成可以准确识别的特征
  - 训练多层神经网络时，通过自编码器训练样本得到参数初始值

自编码器

- 自编码器的基本形式如下图所示



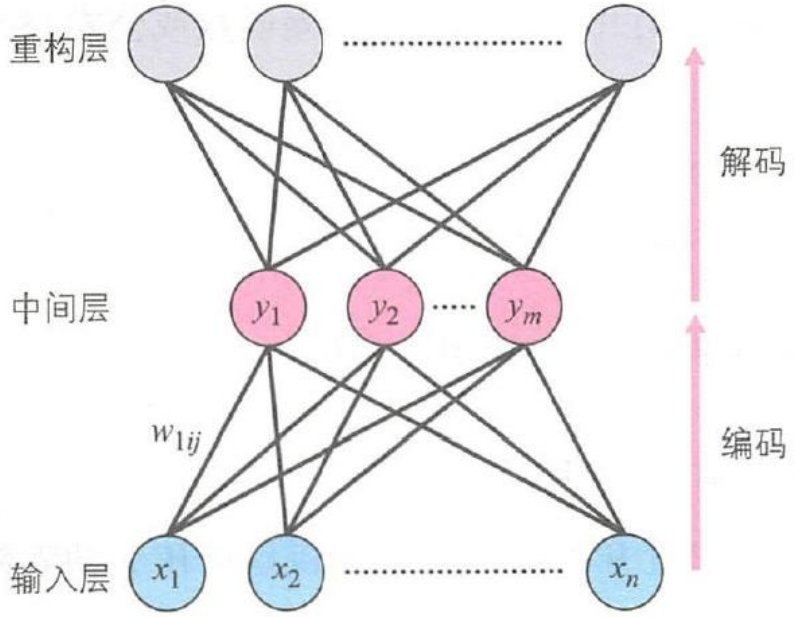
自编码器(两层结构)

$$\boldsymbol{y = f(Wx + b)}$$

和RBM类似，由输入层和输出层组成。输入数据x与对应的连接权重W相乘，再加上偏置b，并经过激活函数$f(\cdot)$变换后，就可以得到输出y

自编码器

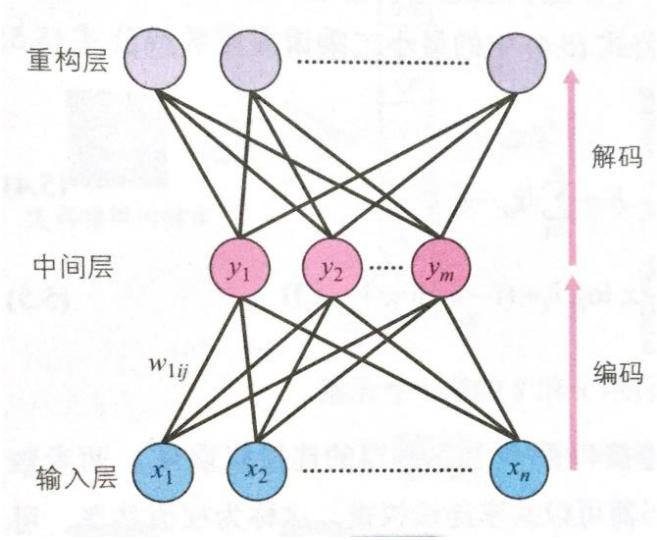
- 自编码器是一种基于无监督学习的神经网络，目的在于通过不断调整参数，重构经过维度压缩的输入样本
- 一种能够重构输入样本的三层神经网络如右图



自编码器(三层结构)

自编码器

- 这里，$f(\cdot)$表示编码器的激活函数，$\tilde{f}(\cdot)$表示解码器的激活函数。中间层和重构层之间的连接权重及偏置分别记为$\tilde{W}$和$\tilde{b}$，重构值记作$\tilde{x}$



$$\tilde{x} = \tilde{f}\left(\tilde{W}y + \tilde{b}\right)$$

$$y = f(Wx + b)$$

$$\tilde{x} = \tilde{f}\left(\tilde{W}f(Wx + b) + \tilde{b}\right)$$

自编码器的训练

- 自编码器的训练就是确定编码器和解码器的参数$W, \widetilde{W}, b, \tilde{b}$的过程。其中$W$和$\widetilde{W}$可以相同，这称为权值共享。训练的目的是尽可能重构其原始输入
- 参数的训练使用误差反向传播算法，误差函数可以使用最小二乘误差函数或交叉熵代价函数

$$E = \sum_{n=1}^{N} \| x_n - \tilde{x}_n \|^2$$

$$E = -\sum_{n=1}^{N} (x_n \log \tilde{x}_n + (1 - x_n) \log(1 - \tilde{x}_n))$$

自编码器

- 当样本中包含噪声时，如果神经网络能够消除噪声，则被称为降噪自编码器（Denoising Autoencoder）
- 还有一种称为稀疏自编码器(Sparse Autoencoder)的网络，它在自编码器中引入了正则化项，以去除冗余信息

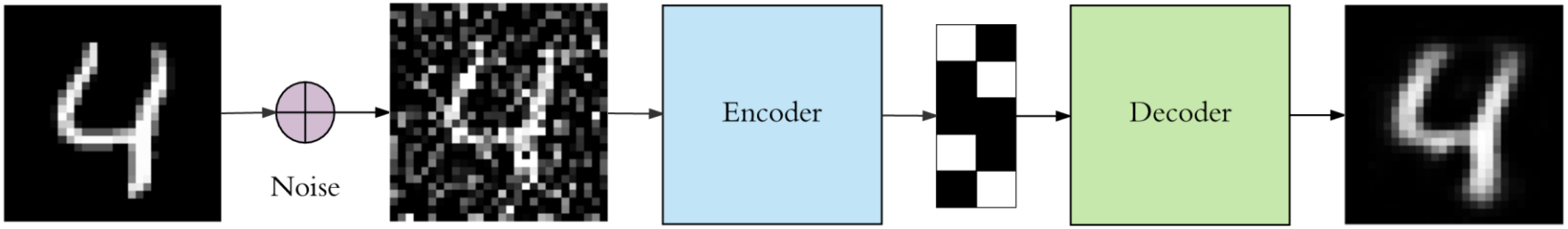
降噪自编码器

- **降噪自编码器（Denoising Autoencoder）的网络结构和自编码器一样，只是对训练方法进行了改进**
  - 自编码器是把训练样本直接输入给输入层，而降噪自编码器则是把通过向训练样本中加入随机噪声得到的样本$\tilde{x}$输入给输入层

$$\tilde{x} = x + \varepsilon$$

降噪自编码器

- 假设随机噪声ε服从**均值为0，方差为σ^2的正态分布**，我们需要训练神经网络，使得**重构结果和不含噪声的样本之间的误差收敛于极小值**
- 误差函数会对不含噪声的输入样本进行计算，故降噪自编码器可以完成以下两项训练
  - 保持输入样本不变的条件下，能够更好地反映样本属性的特征
  - 消除输入样本中包含的噪声



降噪自编码器

- 在MNIST数据集上应用降噪自编码器

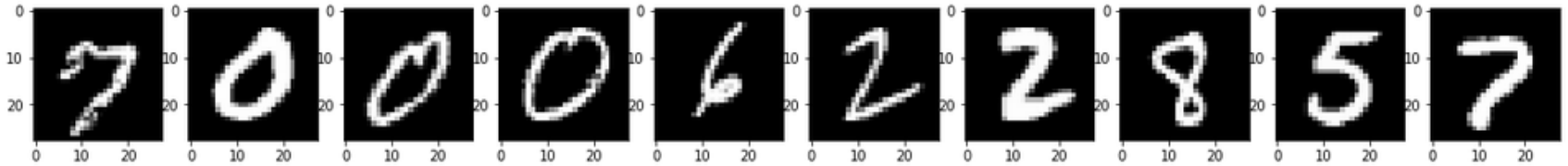


图1：原始图像

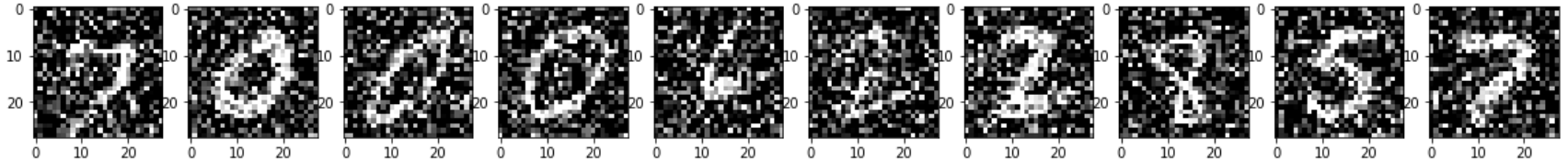


图2：加入噪声后的图像

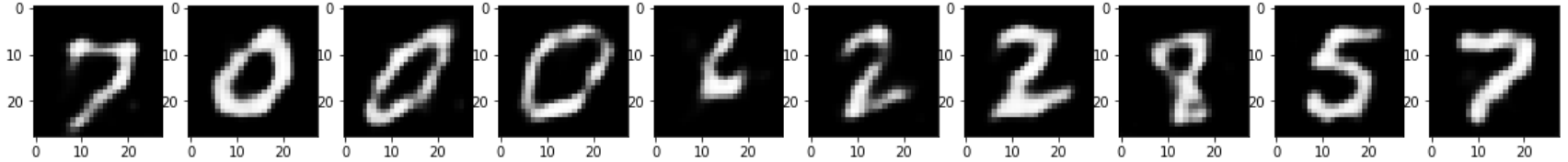


图3：去噪后的图像

稀疏自编码器

- 在多层自编码器中，中间层的单元数太少会导致神经网络很难重构输入样本，而单元数太多又会产生单元冗余，降低压缩效率
- 为了解决这个问题，人们将稀疏正则化引入到自编码器中，提出了**稀疏自编码器**（Sparse Autoencoder）
  - 通过增加正则化项，大部分单元的输出都变成了0，就能利用少数单元完成压缩或重构

稀疏自编码器

❑ 加入正则化后的误差函数E如下所示

$$E = \sum_{n=1}^{N} \|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2 + \beta \sum_{j=1}^{M} KL\left(\rho || \hat{\rho}_j\right)$$

– ρ表示平均激活度的目标值，β用于控制稀疏性的权重，$\hat{\rho}_j$表示中间层第j个单元的平均激活度

$$\hat{\rho}_j = \frac{1}{N} \sum_{n=1}^{N} f\left(W_j \boldsymbol{x}_n + b_j\right)$$

– $KL(\rho||\hat{\rho}_j)$表示KL距离（Kullback-Leibler divergence）

$$KL\left(\rho || \hat{\rho}_j\right) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}$$

稀疏自编码器的训练

- KL距离反映了平均激活度和目标值的差异。ρ值越接近于0，中间层的平均激活度$\hat{\rho}_j$就越小
- 稀疏自编码器的训练也需要用到误差反向传播算法，对误差函数求导时须考虑$KL(\rho||\hat{\rho}_j)$的导数

$$KL(\rho||\hat{\rho}_j) = \rho \log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j} \qquad \hat{\rho}_j = \frac{1}{N}\sum_{n=1}^{N} f(W_j \boldsymbol{x}_n + b_j)$$

$$u_j = f(W_j \boldsymbol{x}_n + b_j)$$

$$\frac{\partial KL(\rho||\hat{\rho}_j)}{\partial u_j} = \left(-\frac{\rho}{\hat{\rho}_j} + \frac{1-\rho}{1-\hat{\rho}_j}\right)\frac{\partial \hat{\rho}_j}{\partial u_j}$$

稀疏自编码器的训练

- 平均激活度是根据所有样本计算出来的，所以在计算任何单元的反向传播之前，需要对所有样本计算一遍正向传播，从而获取平均激活度，所以使用小批量梯度下降法进行训练时的效率很低
- 为了解决此问题，可以只计算Mini-Batch中包含的样本的平均激活度，然后在Mini-Batch之间计算加权平均并求近似值

稀疏自编码器的训练

- 假设时刻$(t-1)$的Mini-Batch的平均激活度为$\hat{\rho}_j^{(t-1)}$，当前t时刻Mini-Batch的平均激活度为

$$\hat{\rho}_j^{(t)} = \lambda\hat{\rho}_j^{(t-1)} + (1-\lambda)\hat{\rho}_j^{(t)}$$

这里的λ是权重，λ越大，则时刻$(t-1)$的Mini-Batch所占的比重也越高

栈式自编码器

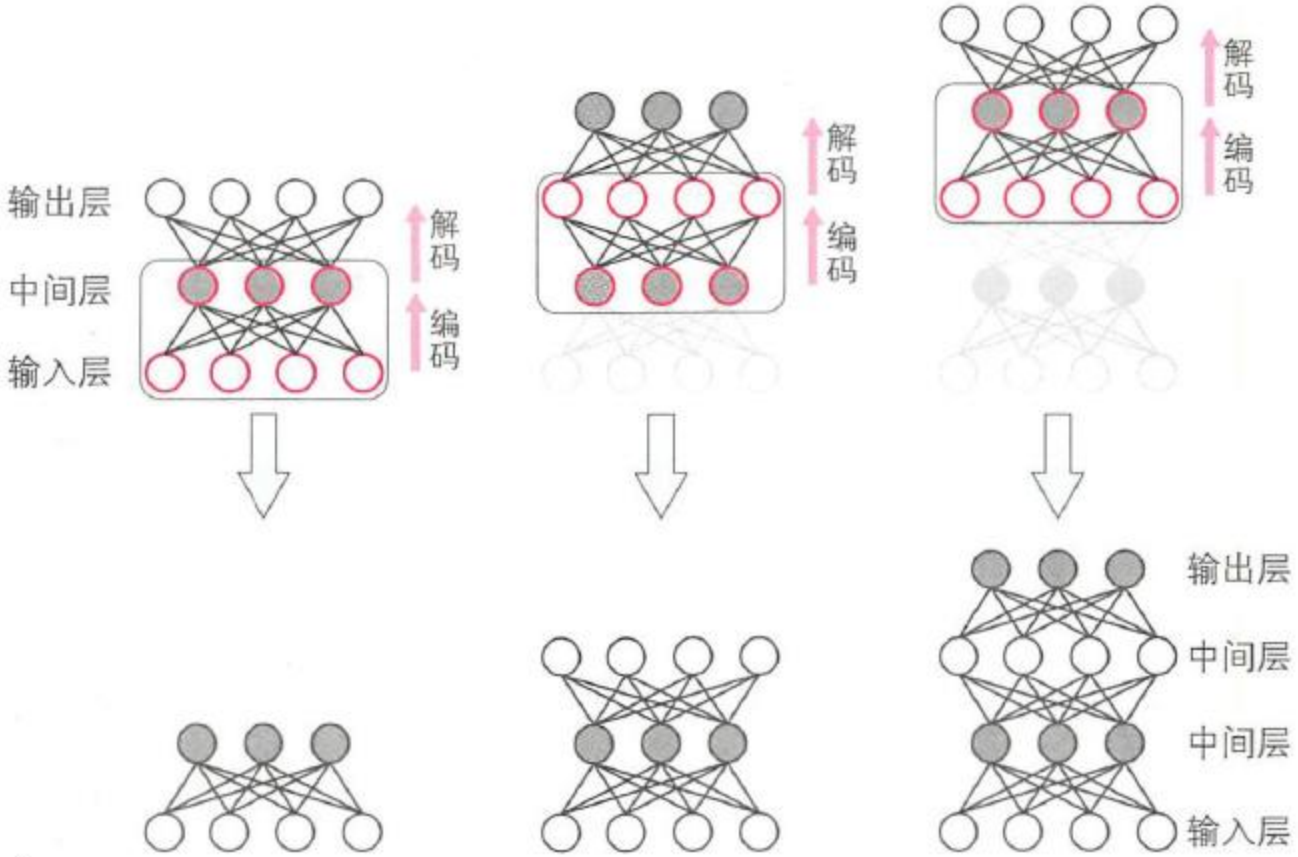
- 自编码器、降噪自编码器、稀疏自编码器都是包括编码器和解码器的三层结构。
- 但是在进行维度压缩时，可以只包括输入层和中间层。输入层和中间层多层堆叠后，就可以得到栈式自编码器 (Stacked Autoencoder)

栈式自编码器的训练

- 栈式自编码器和深度信念网络一样，都是逐层训练。但两种网络的训练方法不同，深度信念网络是利用对比散度算法逐层训练两层之间的参数。而栈式自编码器的训练过程如下

  - 首先训练第一个自编码器，然后保留第一个自编码器的编码器部分
  - 把第一个自编码器的中间层作为第二个自编码器的输入层进行训练
  - 反复地把前一个自编码器的中间层作为后一个编码器的输入层，进行迭代训练

栈式自编码器的训练



在预训练中的应用

- 栈式自编码器每层都能得到有效的参数，所以我们可以把训练后的参数作为神经网络或卷积神经网络的参数初始值，这种方法叫作预训练
- 预训练属于无监督学习，接下来需要使用有监督学习来调整整个网络的参数，这也叫作微调(fine tuning)

6 中英文术语对照

中英文术语对照

- 串扰：Crosstalk
- 能量函数： Energy function
- 模拟退火算法：Simulated Anneal
- 对数似然函数： Log-likelihood function
- 小批量：Mini-batch
- 可见变量：Visible variables
- 隐藏变量：Hidden variables
- 玻尔兹曼机：Boltzmann Machine
- 受限玻尔兹曼机： Restricted Boltzmann Machine
- 深度信念网络：Deep Belief Network

中英文术语对照

- 对比散度算法：Contrastive Divergence
- 稀疏自编码器：Sparse Autoencoder
- 栈式自编码器：Stacked Autoencoder
- 降噪自编码器：Denoising Autoencoder

谢谢！