深 度 学 习

刘 新 旺

Email: xinwangliu@nudt.edu.cn

国 防 科 技 大 学　　计 算 机 学 院
计 算 科 学 系　人 工 智 能 与 大 数 据 教 研 室

2020 年 12 月 22 日

Content

人工神经网络简史

- A Logical Calculus of Ideas Immanent in Nervous Activity
- “神经活动中内在思想的逻辑演算”—发表于1943年
- 作者: **麦卡洛可**（McCulloch）和**皮茨**（Pitts)
- 这篇文章也成了控制论的思想源泉之一
- 控制论的创始人**诺伯特·维纳**
- 维纳后来在哈佛任教，但不被主流数学家喜欢，最后到了隔壁的麻省理工落脚
- 维纳提出“控制论”后出了大名，在麻省理工搞了一大笔钱
- **麦卡洛可**带着**皮茨**等投奔**维纳**

人工神经网络简史

- **皮茨**和**维纳**闹翻后，拒绝麻省理工的研究生学位，对学问也心灰意冷。于1969年比他的长辈**麦卡洛可**早几个月离世，只有四十六岁
- 得**维纳**真传的人：**迈克尔·阿比卜**（Michael Arbib)，23岁就在维纳手下取得了PhD
- **阿比卜**创办了麻省大学的计算机系，并延揽一帮人工智能人马，其中有后来以“强化学习”出名的**巴托**（Andy Barto)
- 巴托的“可适应学习实验室”曾经短期收容了很多人，其中有后来的大佬级人物，如**乔丹**（Michael Jordan），乔丹在伯克利时又培养了**Andrew Ng**等一干人马

人工神经网络简史

- 1949年，神经心理学家Hebb出版《行为组织学》（Organization of Behavior)
- “Hebb规则”: 如果两个细胞总是同时激活的话，它们之间就有某种关联。同时激活的概率越高，关联度也越高
- 神经网络研究的后一个大突破是1957年。康奈尔大学实验心理学家**弗兰克·罗森布拉特**模拟实现了一种“感知机”(Perceptron)的神经网络模型
- **罗森布拉特**1962年出了本书：“Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms”
- **罗森布拉特**的名声越来越大，得到的研究经费也越来越多。国防部和海军都资助了他的研究工作，媒体对罗森布拉特也表现出了过度的关注

人工神经网络简史

- **明斯基**是人工智能的奠基人之一，是达特茅斯会议的组织者。**明斯基**在一次会议上和**罗森布拉特**大吵，认为神经网络不能解决人工智能的问题
- **明斯基**和麻省理工学院的**佩普特**教授合作，企图从理论上证明他们的观点
- 合作的成果："Perceptrons: An Introduction to Computational Geometry", 证明单层神经网络不能解决XOR（异或）问题
- **罗森布拉特**也已猜到"感知机"可能存在限制。但"感知机"的缺陷被**明斯基**以一种敌意的方式呈现，对**罗森布拉特**是致命打击

人工神经网络简史

- 所有原来的政府资助机构也逐渐停止对神经网络的研究。1971年，**罗森布拉特**四十三岁生日那天，在划船时淹死
- 表面是科学，但有证据表明**明斯基**和**罗森布拉特**以前就有瓜葛
- 他们是中学同学。布朗克斯（Bronx）科学高中是全世界最好的高中，**明斯基**是1944年毕业生，**乔姆斯基**是1945年毕业生，而**罗森布拉特**是1946年毕业生。明斯基和罗森布拉特至少有两年重叠，而且彼此认识，互相嫉妒
- 1956年的达特茅斯会议定义了“人工智能”这个词。这个会议在定义“人工智能”领域时只是提到了神经网络，那时**明斯基**是神经网络的支持者

人工神经网络简史

- 他1954年在普林斯顿的博士论文题目是“神经-模拟强化系统的理论、及其在大脑模型问题上的应用”
- **罗森布拉特**被比他大一岁的**明斯基**妒忌是自然的
  - **明斯基**所负责的麻省理工学院的人工智能实验室也在向国防部和海军申请经费
  - 大多数的圈内科学家，对**罗森布拉特**突然被塑造的明星范儿很反感
- **维德罗**（Widrow）是斯坦福大学教授，在**罗森布拉特**刚提出“感知机”时，他提出了Adaline可适应性算法
- 感知机的失败导致神经网络研究的势微，用加州理工学院的集成电路大佬**米德**（Carver Mead）的话说是“二十年大饥荒”

人工神经网络简史

- 1974年，哈佛一篇博士论文证明在神经网络多加一层，并且利用后向传播(Back-propagation)学习方法，可以解决XOR问题
- 1982年，那时在加州理工担任生物物理教授的**霍普菲尔德**，提出了一种新的神经网络，可以解决一大类模式识别问题，还可以给出一类组合优化问题的近似解
- “神经网络”在八十年代就像九十年代的互联网，后来的Web2.0，和眼下的“大数据”
- 1993年，美国电气电子工程学会IEEE开始出版“神经网络会刊”，即IEEE Transactions on Neural Networks，为该领域的高质量文章提供出版渠道。美国国防部和海军、能源部等也加大资助力度

人工神经网络简史

- **霍普菲尔德**也培养了一批后起之秀，包括**辛顿**（Geoffrey Hinton）
- **辛顿**先转往卡内基梅隆，最终到加拿大多伦多大学计算机系任教
- 他是**布尔**的外曾曾孙子，他曾祖母Ellen是布尔的女儿
- 不太为外人所知的革命家史: 中国革命的参与者、美国铁杆左派**韩丁**和**寒春**也是Ellen的孙子孙女
- 布尔的小女儿、Ellen的妹妹**伏尼契**(Ethel Lilian Voynich)是传遍苏联和中国的小说《牛虻》的作者

人工神经网络简史

- 神经网络在八十年代的光芒被后来的互联网掩盖了，但这几年又恰恰是互联网给了神经网络更大的机会
- 神经网络由一层一层的神经元构成。层数越多，就越深，所谓深度学习就是用很多层神经元构成的神经网络达到机器学习的功能
- **辛顿**就是“深度学习”的开创者，他2006年的一篇文章开辟了这个新领域
- 年过六十的辛顿不甘寂寞，和他的两个学生开了家专注深度学习的公司。谷歌出了几千万美元于2013年初收购了这家只有三名员工的公司

人工神经网络简史

- 2012年，斯坦福大学人工智能实验室主任Andrew Ng和谷歌合作建造了一个当时最大的神经网络。网络上一度疯传的谷歌猫脸识别就是用的这个参数多达十七亿的神经网络
- 借助Google大脑，无需接受人类的任何培训和指令，就可以利用内在算法从海量数据中自动提取信息，学会如何识别猫
- 2015年12月，百度在Imagenet上更深层网络(152层)的创建

Content

机器学习流程图

- 通过传感器来获得数据。然后经过**预处理、特征提取、特征选择**，再到**推理、预测或者识别**
- 绝大部分的工作聚焦于最后一个部分，也就是机器学习
- 中间的三部分，概括起来就是特征表达
- 良好的特征表达，对最终算法的准确性起关键的作用
- 然而，它一般都是人工完成的，靠人工提取特征

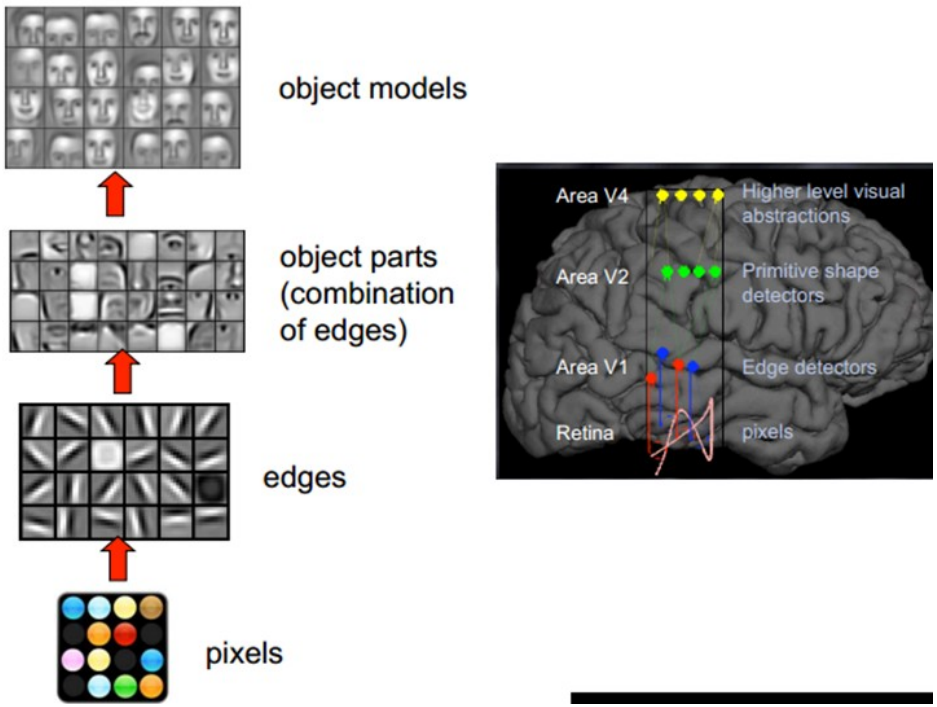
机器学习流程图

- 也出现了不少优秀的特征，如SIFT
- 然而，手工地选取特征是一件非常费力、启发式（需要专业知识）的方法，能不能选取好很大程度上靠经验和运气，而且它的调节需要大量的时间
- 既然手工选取特征不太好，那么能不能自动地学习特征呢？答案是能!
- Deep Learning就是自动地学习特征。它的一个别名“Representation Learning”

学习特征

- 那它是怎么学习的呢？怎么知道哪些特征好哪些不好呢？
- 机器学习是一门专门研究计算机怎样模拟或实现人类的学习行为的学科
- 我们人的视觉系统是怎么工作的呢？
- 1981年的诺贝尔医学奖，颁发给了David Hubel和Torsten Wiesel。主要贡献是发现了视觉系统的信息处理: **可视皮层是分级的**
- 该发现激发了人们对于神经系统的进一步思考。神经-中枢-大脑的工作过程，或许是一个**不断迭代、不断抽象**的过程

抽象与迭代



- 从原始信号摄入开始（瞳孔摄入像素Pixels）
- 接着做初步处理（大脑皮层某些细胞发现**边缘和方向**）
- 然后抽象（大脑判定，眼前的物体的**形状是圆形的**）
- 然后进一步抽象（大脑进一步**判定该物体**）

分级

- 人的视觉系统的信息处理是分级的。
- 从低级的V1区**提取边缘特征**，再到V2区的**形状或者目标部分**等，再到更高层，整个目标、目标的行为等
- 高层特征是低层特征的组合，从低层到高层特征表示越来越抽象，越来越能表现语义或者意图
- 抽象层面越高，存在的可能猜测就越少，就越利于分类

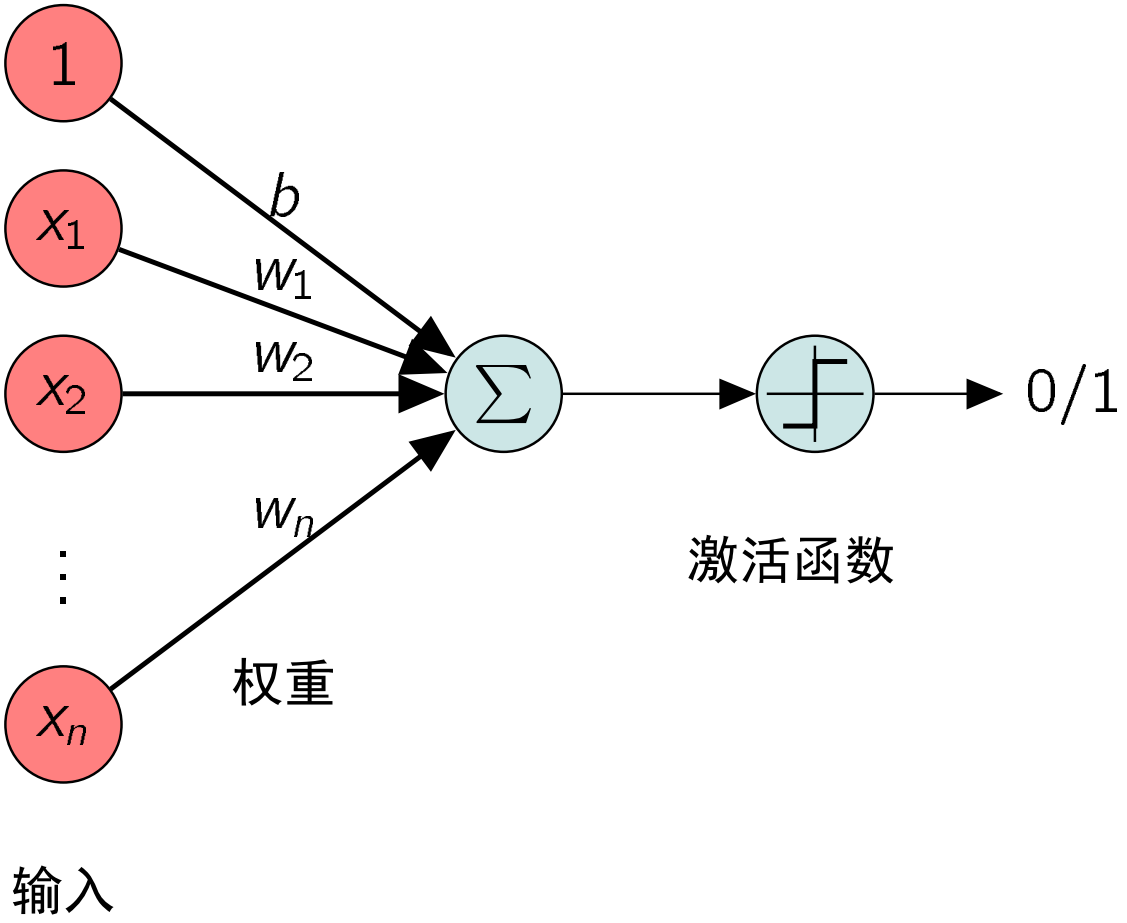
感知器

感知器也是最简单的神经网络（只有一层）。感知器是由美国计算机科学家Roseblatt于1957年提出的。

感知器可谓是最简单的人工神经网络，只有一个神经元。感知器也可以看出是线性分类器的一个经典学习算法。

感知器模型

感知器是模拟生物神经元行为的机器，有与生物神经元相对应的部件，如权重、偏置及激活函数，输出为0或1。



给定一个d维的输入$\mathbf{x} = [x_1, \cdots, x_d]^\top$，

$$\hat{y} = \begin{cases} +1 & \text{if} \quad \boldsymbol{\omega}^\top \mathbf{x} + b > 0 \\ -1 & \text{if} \quad \boldsymbol{\omega}^\top \mathbf{x} + b \leq 0 \end{cases} \tag{1}$$

其中$\boldsymbol{\omega} \in \mathbb{R}^d$是权重向量，$b$是偏置。$\boldsymbol{\omega}$和$b$是未知的，需要从给定的训练数据集中学习得到。

不失一般性，定义$\boldsymbol{\omega} \triangleq [\boldsymbol{\omega}^\top, b]^\top$, $\mathbf{x} \triangleq [\mathbf{x}^\top, 1]^\top$，公式1可以简写为：

$$\hat{y} = \begin{cases} +1 & \text{if} \quad \boldsymbol{\omega}^\top \mathbf{x} > 0 \\ -1 & \text{if} \quad \boldsymbol{\omega}^\top \mathbf{x} \leq 0 \end{cases} \tag{2}$$

两类感知器算法Rosenblatt [1958]

输入: 训练集: $(\mathbf{x}_i, y_i), i = 1, \cdots, N$，迭代次数：$T$

输出: $\mathbf{w}$

初始化：$\mathbf{w}_0 = 0$;

$k = 0$;

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

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

    选取一个样本$(\mathbf{x}_i, y_i)$，**if** $\mathbf{w}^T(y_i \mathbf{x}_i) < 0$ **then**

      $\mathbf{w}_{k+1} = \mathbf{w}_k + y_i \mathbf{x}_i$，;

      $k = k + 1$;

    end

  end

end

return $\mathbf{w}_k$;

Novikoff[1963]证明对于两类问题，如果训练集是线性可分的，那么感知器算法可以在有限次迭代后收敛。然而，如果训练集不是线性分隔的，那么这个算法则不能确保会收敛。

感知器收敛性

对于任何线性可分的训练集$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$，假设$R$是所有样本中输入向量的模的最大值，即$R = \max_i \|\mathbf{x}_i\|$，那么在感知器学习算法中，总共的预测错误次数$K \leq \frac{R^2}{\gamma^2}$。

Content

人工神经网络模型主要考虑网络连接的拓扑结构、神经元的特征、学习规则等。目前，已有近40种神经网络模型。

- 前馈神经网络：也经常称为多层感知器（Multilayer Perceptron，MLP）。
- 反馈神经网络：网络内神经元间有反馈，这种神经网络的信息处理是状态的变换，可以用动力学系统理论处理。

人工神经元使用一个非线性的激活函数，输出一个活性值。假定神经元接受d个输入$\mathbf{x} = [x_1, \cdots, x_d]^\top$，用状态$z$表示一个神经元所获得的输入信号$\mathbf{x}$的加权和，输出为该神经元的活性值$a$。具体定义如下：

$$\begin{aligned} z &= \boldsymbol{\omega}^\top \mathbf{x} + b \\ a &= f(z) \end{aligned} \tag{3}$$

其中$\boldsymbol{\omega}$是d维的权重向量，b是偏置。典型的激活函数$f(\cdot)$有Sigmoid型函数等。

神经元示例

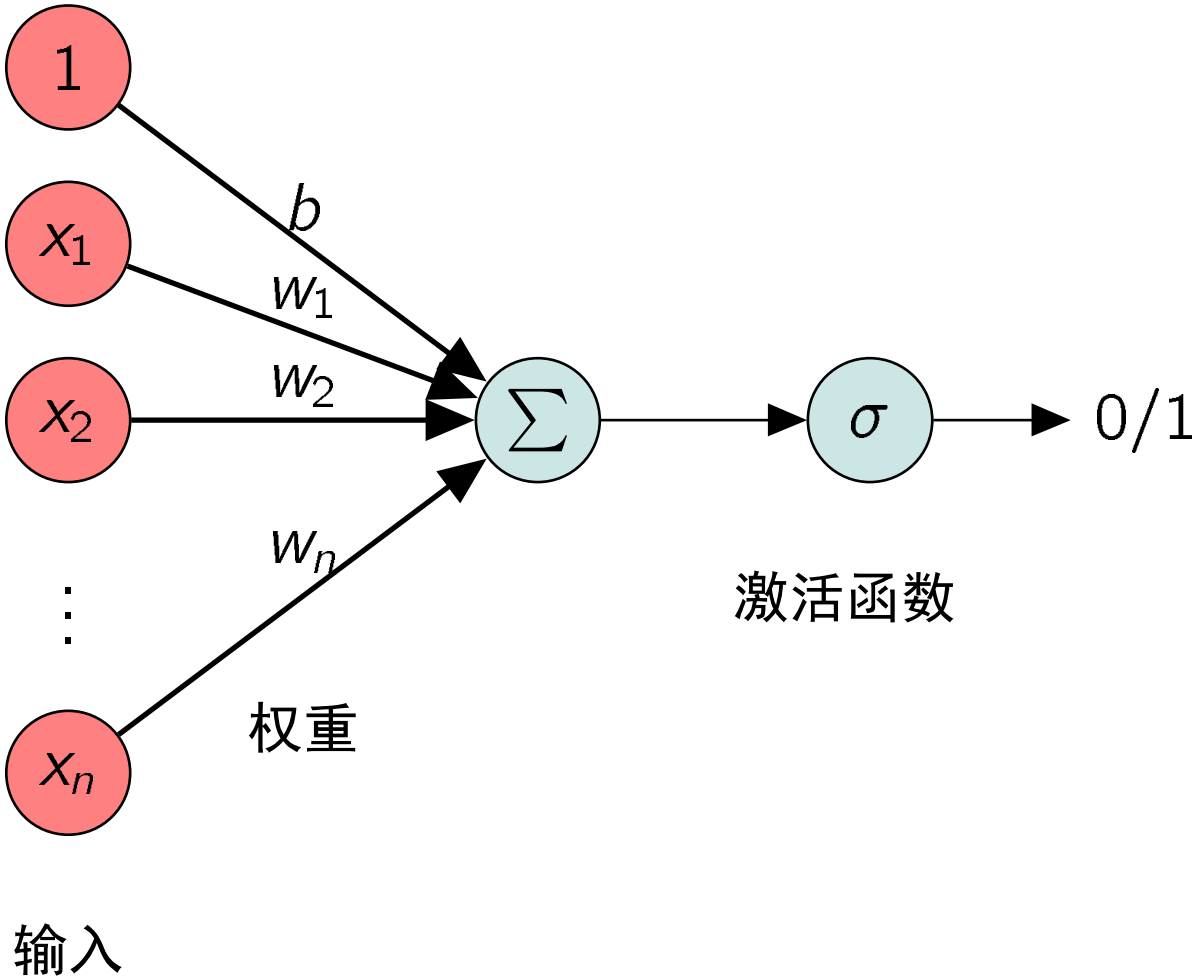
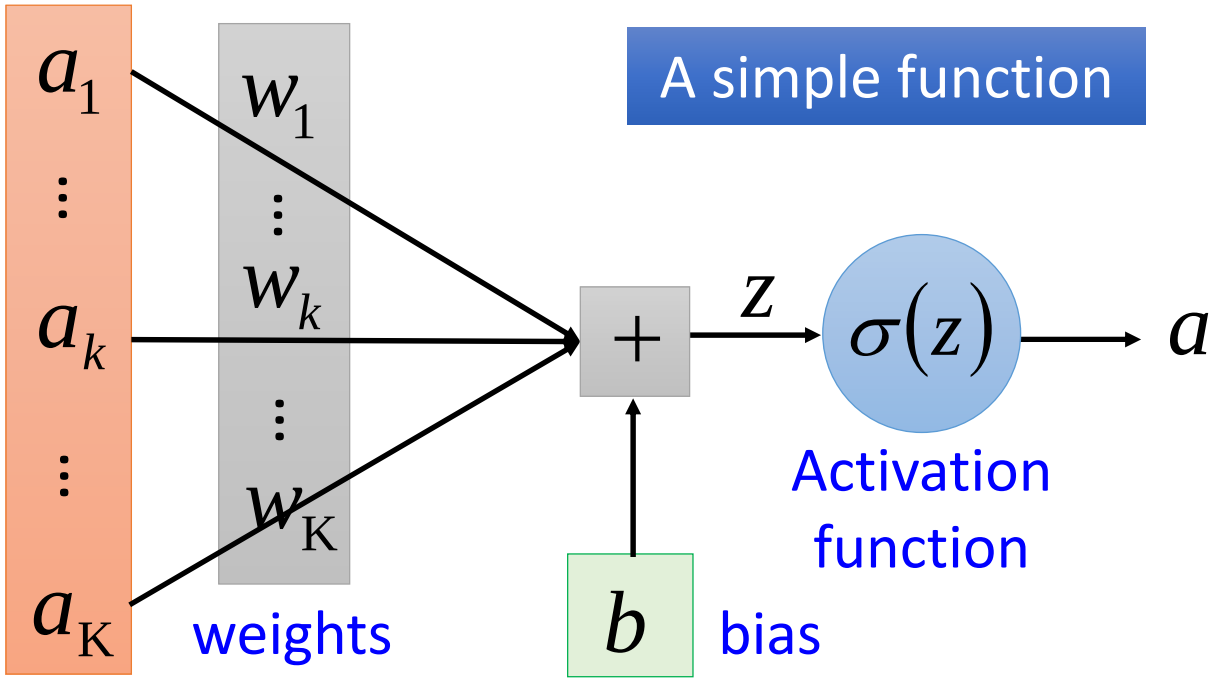


图: 人工神经元模型

Neural Network (神经网络)

Neuron (神经元)

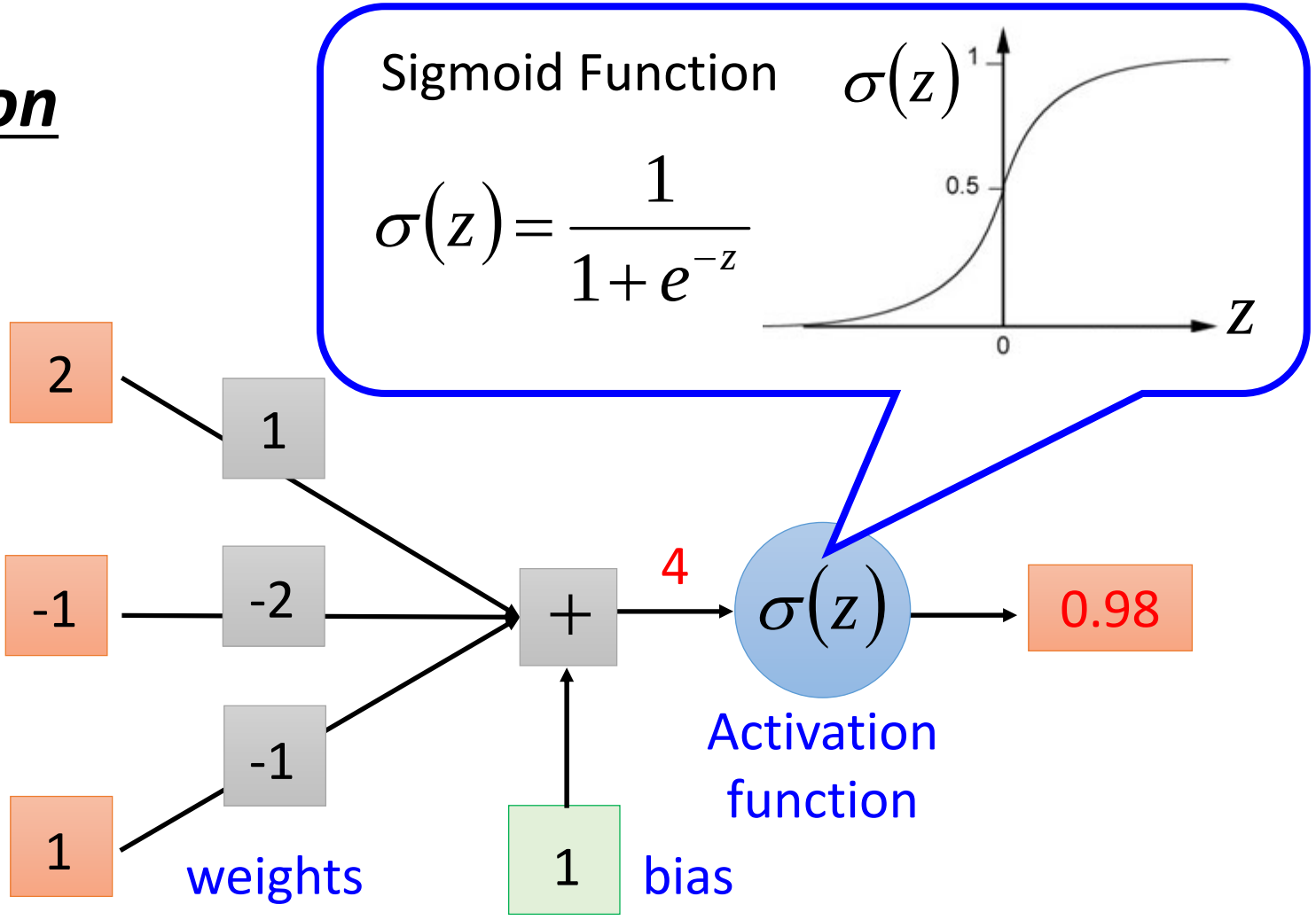
$$z = a_1 w_1 + \ldots + a_k w_k + \ldots + a_K w_K + b$$



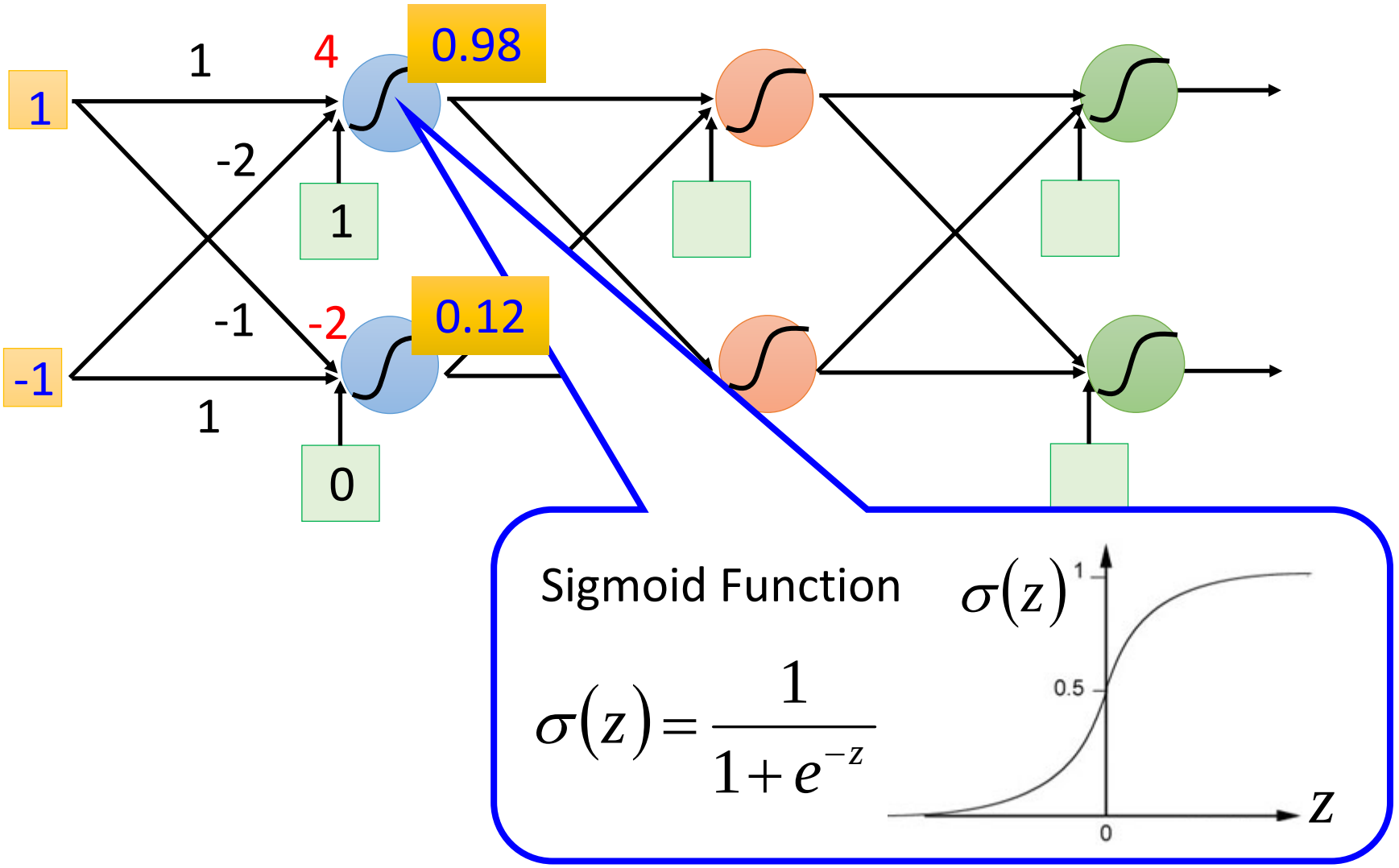
Weights and biases are called network parameters

Neural Network (神经网络)

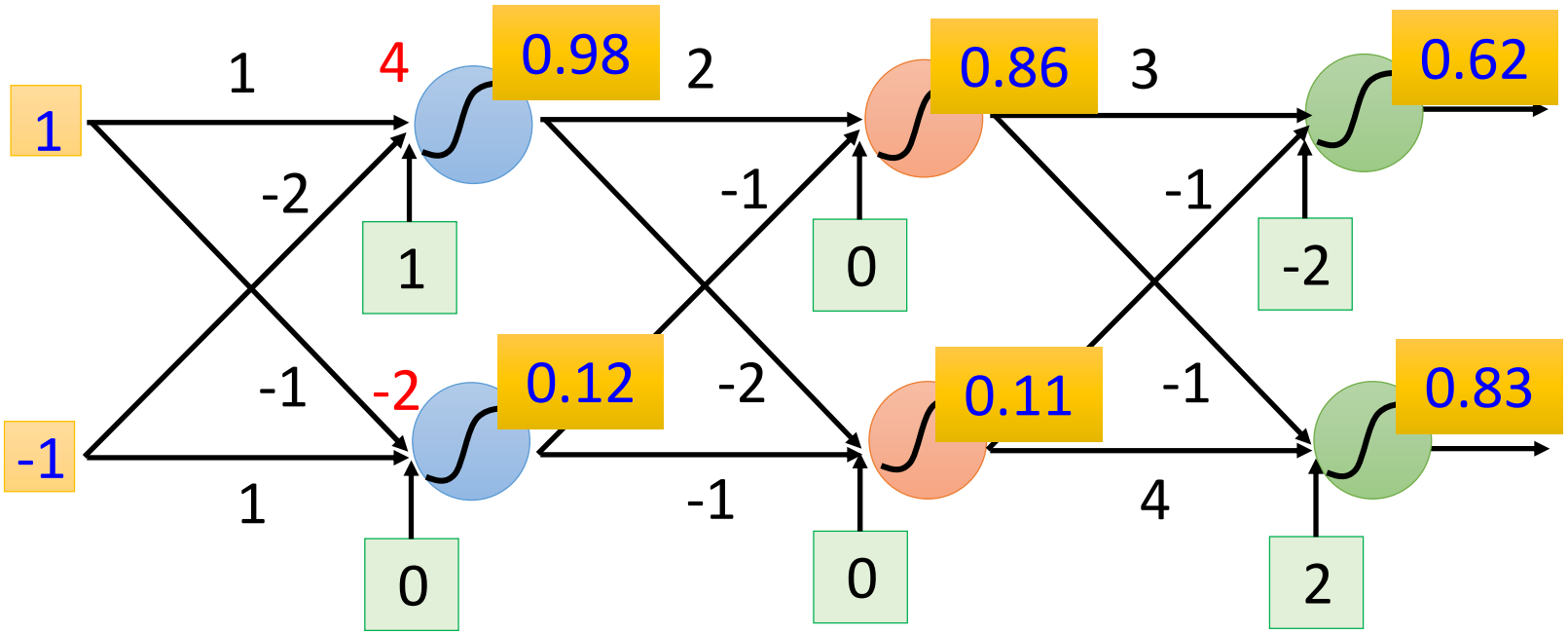
Neuron



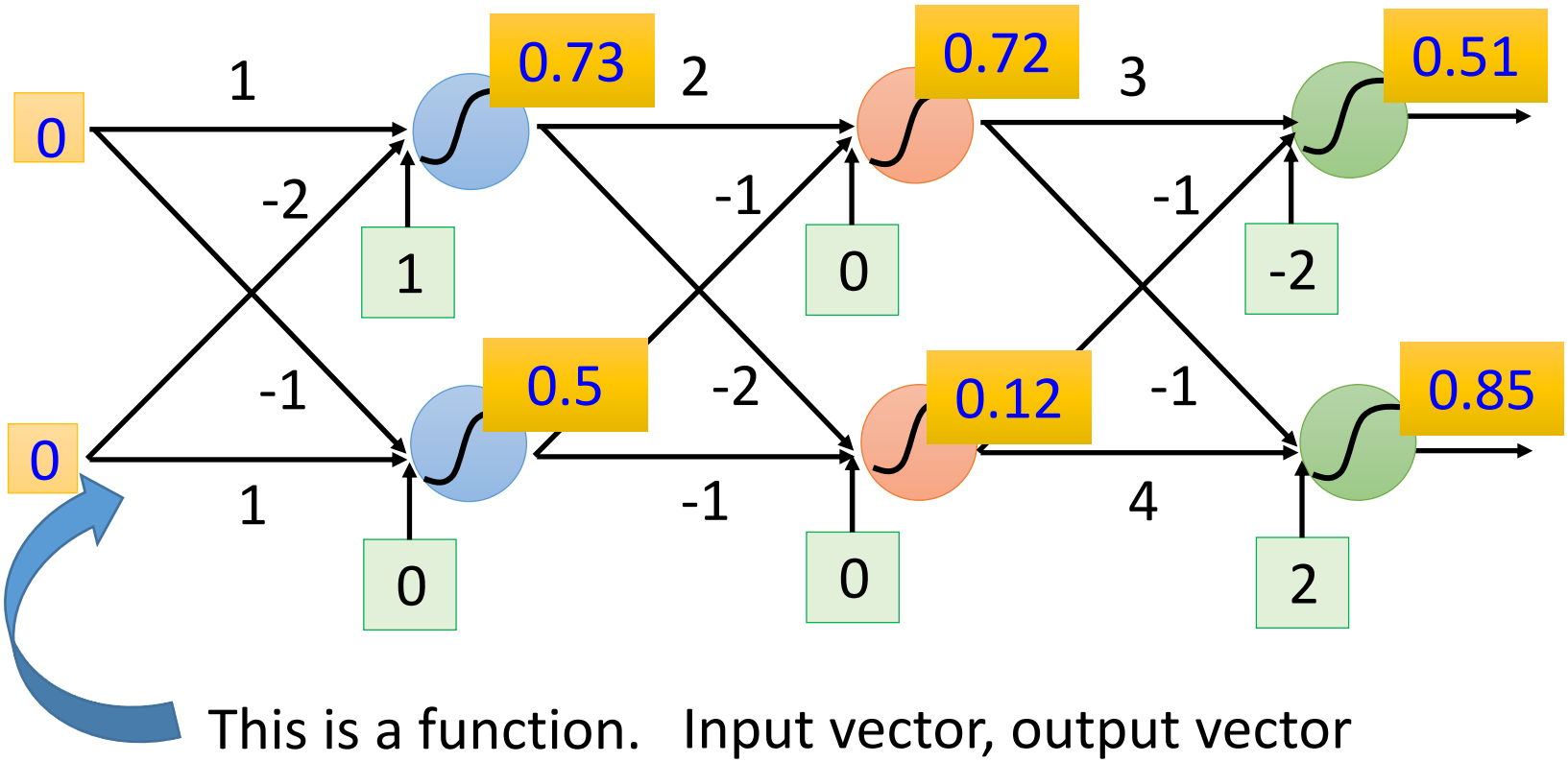
Neural Network (神经网络)



Neural Network (神经网络)



Neural Network (神经网络)



This is a function. Input vector, output vector

$$f\left(\begin{bmatrix}1\\-1\end{bmatrix}\right)=\begin{bmatrix}0.62\\0.83\end{bmatrix}\quad f\left(\begin{bmatrix}0\\0\end{bmatrix}\right)=\begin{bmatrix}0.51\\0.85\end{bmatrix}$$

传统神经网络中最常用的激活函数分别是Sigmoid型函数。Sigmoid型函数是指一类S型曲线函数，常用的Sigmoid型函数有logistic函数$\sigma(x)$和tanh函数。

$$\begin{aligned} \sigma(x) &= \frac{1}{1+e^{-x}} \\ \tanh(x) &= \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \end{aligned} \tag{4}$$

tanh函数可以看作是放大并平移的logistic函数: $\tanh(x) = 2\sigma(2x) - 1$。

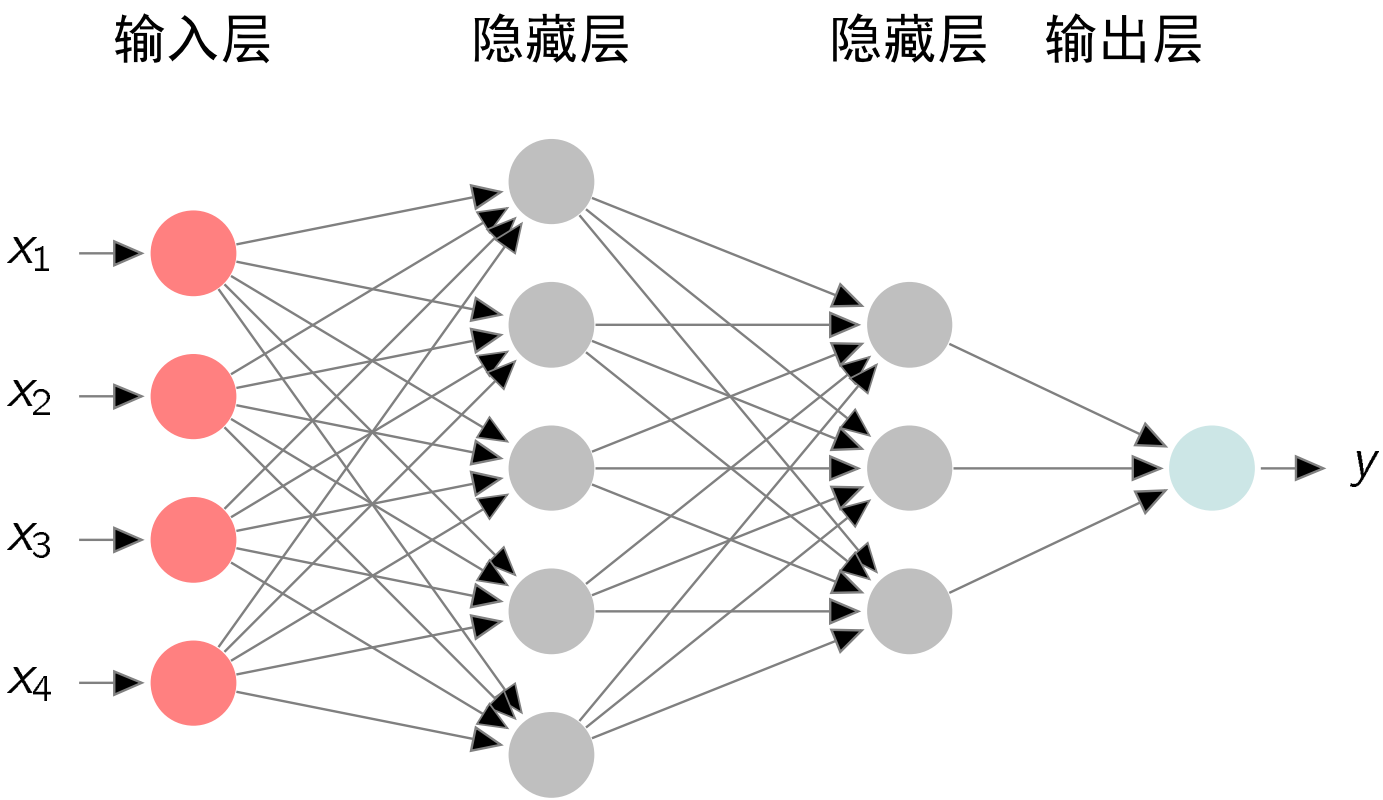
rectifier函数[Glorot et al., 2011]，定义为

$$\text{rectifier}(x) = \max(0, x) \tag{5}$$

rectifier函数被认为有生物上的解释性。神经科学家发现神经元具有单侧抑制、宽兴奋边界、稀疏激活性等特性。

采用rectifier函数的单元也叫作**修正线性单元**（rectified linear unit，ReLU）[Nair and Hinton, 2010]。

在前馈神经网络中，各神经元分别属于不同的层。整个网络中无反馈，信号从输入层向输出层单向传播。



给定一个前馈神经网络，我们用下面的记号来描述这样网络

- L: 表示神经网络的层数
- n^l: 表示第l层神经元的个数
- $f_l()$: 表示第l层神经元的激活函数
- $\mathbf{W}^{(l)} \in \mathbb{R}^{n^l \times n^{l-1}}$: 表示第$l-1$层到第$l$层的权重矩阵
- $\mathbf{b}^{(l)} \in \mathbb{R}^{n^l}$: 表示第$l-1$层到第$l$层的偏置
- $\mathbf{z}^{(l)} \in \mathbb{R}^{n^l}$: 表示第$l$层神经元的状态
- $\mathbf{a}^{(l)} \in \mathbb{R}^{n^l}$: 表示第$l$层神经元的活性值。

前馈神经网络通过下面公式进行信息传播:

$$\begin{aligned} \mathbf{z}^{(l)} &= \mathbf{W}^{(l)^\top} \mathbf{a}^{(l-1)} + b^{(l)} \\ \mathbf{a}^{(l)} &= f_l(\mathbf{z}^{(l)}) \end{aligned} \tag{6}$$

公式(6)也可以合并写为：

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)^\top} f_{(l-1)}(\mathbf{z}^{(l-1)}) + b^{(l)} \tag{7}$$

这样，前馈神经网络可以通过逐层的信息传递，得到网络最后的输出$\mathbf{a}^{(L)}$

$$\mathbf{x} = \mathbf{a}^{(0)} \to \mathbf{z}^{(1)} \to \mathbf{a}^{(1)} \to \mathbf{z}^{(2)} \to \cdots \to \mathbf{a}^{(L-1)} \to \mathbf{z}^{(L)} \to \mathbf{a}^{(L)} = \mathbf{y} \tag{8}$$

给定一组样本$\{(\mathbf{x}_i, y_i)\}_{i=1}^n$，用前馈神经网络的输出为$f(\mathbf{x}|\mathbf{W}, \mathbf{b})$，目标函数为

$$\begin{aligned} J(\mathbf{W}, \mathbf{b}) &= \sum_{i=1}^n \ell(f(\mathbf{x}_i|\mathbf{W}, \mathbf{b}), y_i) + \frac{\lambda}{2}\|\mathbf{W}\|_{\mathrm{F}}^2 \\ &= \sum_{i=1}^n J(\mathbf{W}, \mathbf{b}; \mathbf{x}_i, y_i) + \frac{\lambda}{2}\|\mathbf{W}\|_{\mathrm{F}}^2 \end{aligned} \tag{9}$$

其中$\mathbf{W}$和$\mathbf{b}$包含了每一层的权重矩阵和偏置向量，$\|\mathbf{W}\|_{\mathrm{F}}^2 = \sum_{l=1}^L \sum_{i=1}^{n^{l+1}} \sum_{j=1}^{n^l} \left(W_{ij}^{(l)}\right)^2$

学习的目标是最小化$J(\mathbf{W},\mathbf{b})$。如果采用梯度下降方法，可以用如下方法更新参数：

$$\begin{aligned}
\mathbf{W}^{(l)} &= \mathbf{W}^{(l)} - \alpha\frac{\partial J(\mathbf{W},\mathbf{b})}{\partial \mathbf{W}^{(l)}} \\
&= \mathbf{W}^{(l)} - \alpha\left(\sum_{i=1}^{n}\left(\frac{\partial J(\mathbf{W},\mathbf{b};\mathbf{x}_i,y_i)}{\partial \mathbf{W}^{(l)}}\right) + \lambda\mathbf{W}^{(l)}\right) \\
\mathbf{b}^{(l)} &= \mathbf{b}^{(l)} - \alpha\frac{\partial J(\mathbf{W},\mathbf{b})}{\partial \mathbf{b}^{(l)}} \\
&= \mathbf{b}^{(l)} - \alpha\sum_{i=1}^{n}\left(\frac{\partial J(\mathbf{W},\mathbf{b};\mathbf{x}_i,y_i)}{\partial \mathbf{b}^{(l)}}\right)
\end{aligned} \tag{10}$$

α是参数的更新(学习)率。

$\frac{\partial J(\mathbf{W},\mathbf{b};\mathbf{x},y)}{\partial \mathbf{W}^{(l)}}$怎么计算？

根据链式法则，$\frac{\partial J(\mathbf{W},\mathbf{b};\mathbf{x},y)}{\partial \mathbf{W}_{ij}^{(l)}}$可以写为

$$\frac{\partial J(\mathbf{W},\mathbf{b};\mathbf{x},y)}{\partial \mathbf{W}_{ij}^{(l)}} = \left(\frac{\partial J(\mathbf{W},\mathbf{b};\mathbf{x},y)}{\partial \mathbf{z}^{(l)}}\right)^{\top} \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{W}_{ij}^{(l)}} \tag{11}$$

对于第l层，定义一个误差项$\delta^{(l)} = \frac{\partial J(\mathbf{W},\mathbf{b};\mathbf{x},y)}{\partial \mathbf{z}^{(l)}} \in \mathbb{R}^{(n^l)}$为目标函数关于第$l$层的神经元$\mathbf{z}^{(l)}$的偏导数，表示第$l$层神经元对最终误差的影响。

因为$\mathbf{z}^{(l)} = \mathbf{W}^{(l)\top}\mathbf{a}^{(l-1)} + b^{(l)}$，

$$\frac{\partial \mathbf{z}^{(l)}}{\partial W_{ij}^{(l)}} = \frac{\partial (W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)})}{\partial W_{ij}^{(l)}} = \begin{bmatrix} 0 \\ \vdots \\ a_j^{(l-1)} \\ \vdots \\ 0 \end{bmatrix} \leftarrow \text{第 } i \text{ 行}$$

因此，

$$\frac{\partial J(\mathbf{W}, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{W}_{ij}^{(l)}} = \delta_i^{(l)} a_j^{(l-1)} \tag{12}$$

公式(11)可以写为

$$\frac{\partial J(\mathbf{W},\mathbf{b};\mathbf{x},y)}{\partial \mathbf{W}^{(l)}} = \delta^{(l)} \left(a^{(l-1)}\right)^{\top} \tag{13}$$

同理可得，

$$\frac{\partial J(\mathbf{W},\mathbf{b};\mathbf{x},y)}{\partial \mathbf{b}^{(l)}} = \delta^{(l)} \tag{14}$$

计算第l层的误差项$\delta^{(l)}$

$$\begin{aligned}\delta^{(l)} &\triangleq \frac{\partial J(W,\mathbf{b};\mathbf{x},y)}{\partial \mathbf{z}^{(l)}} \\ &= \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \cdot \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \cdot \frac{\partial J(W,\mathbf{b};\mathbf{x},y)}{\partial \mathbf{z}^{(l+1)}} \\ &= \mathbf{diag}(f_l'(\mathbf{z}^{(l)})) \cdot (W^{(l+1)})^{\top} \cdot \delta^{(l+1)} \\ &= f_l'(\mathbf{z}^{(l)}) \odot ((W^{(l+1)})^{\top}\delta^{(l+1)}),\end{aligned}$$

其中$\odot$是向量的点积运算符，表示每个元素相乘。

从上式可以看出，第l层的误差项可以通过第$l+1$层的误差项计算得到。这就是误差的**反向传播**(Backpropagation，BP)。

反向传播算法的含义：**第l层每个神经元的误差项是所有与该神经元相连的第$l+1$层的神经元的误差项**的权重之和，再乘上该神经元激活函数的梯度。

在计算出每一层的误差项之后，就可以得到每一层参数的梯度。因此，前馈神经网络的训练过程可以分为以下三步：(1)先前馈计算每一层的状态和激活值，直到最后一层；(2)反向传播计算每一层的误差；(3)计算每一层参数的偏导数，并更新参数。该训练过程被称为**反向传播算法**。

卷积神经网络(Convolutional Neural Networks，CNN)是一种前馈神经网络。卷积神经网络是受生物学上感受野(Receptive Field)的机制而提出的。**感受野**主要是指听觉系统和视觉系统中神经元的一些性质。比如在视觉神经系统中，一个神经元的感受野是指视网膜上的特定区域，只有这个区域内的刺激才能够激活该神经元。

卷积神经网络有三个结构上的特性：**局部连接，权重共享以及(空间或时间上的)次采样**。这些特性使得卷积神经网络具有一定程度上的**平移、缩放和扭曲不变性**。

卷积是分析数学中一种重要的运算。这里只考虑离散序列的情况。一维卷积经常用在信号处理中。给定一个输入信号序列$x_t\ (t=1,\cdots,n)$和滤波器$f_k\ (k=1,\cdots,m)$，一般情况下滤波器的长度m远小于信号序列长度n。

卷积的输出为:

$$y_t=\sum_{k=1}^{m}f_k x_{t-k+1} \tag{15}$$

当滤波器$f_k=1/m$时，卷积相当于信号序列的移动平均。

卷积的结果按输出长度不同可以分为两类：一类是宽卷积，输出长度n+m-1，对于不在$[1,n]$范围之外的x_t用零补齐(zero-padding)。一类是窄卷积，输出长度$n-m+1$，不补零。

两维卷积经常用在图像处理中。给定一个图像$X_{ij}\,(1 \leq i \leq M,\, 1 \leq j \leq N)$和滤波器$f_{ij}\,(1 \leq i \leq m,\, 1 \leq j \leq n)$，一般$m \ll M,\, n \ll N$。卷积的输出为：

$$y_{ij} = \sum_{u=1}^{m} \sum_{v=1}^{n} f_{uv} X_{i-u+1,j-v+1} \tag{16}$$

在图像处理中，常用的均值滤波(mean filter)就是当前位置的像素值设为滤波器窗口中所有像素的平均值，也就是$f_{uv} = \frac{1}{mn}$。

卷积层：用卷积来代替全连接

在全连接前馈神经网络中，如果第l层有$n^{(l)}$个神经元，第$l-1$层有$n^{(l-1)}$个神经元，连接边有$n^{(l)} \times n^{(l-1)}$个，即权重矩阵有$n^{(l)} \times n^{(l-1)}$个参数。当m和n都很大时，权重矩阵的参数非常多，训练的效率会非常低。

如果采用卷积来代替全连接，第l层的每一个神经元都只和第$l-1$层的一个局部窗口内的神经元相连，构成一个局部连接网络。第l层的第i个神经元的输入定义为:

$$\begin{aligned} a_i^{(l)} &= f_l\left(\sum_{j=1}^{m} {W_j^{(l)}}^{\top} a_{i-m+j}^{(l-1)} + b^{(l)}\right) \\ &= f_l\left({\mathbf{W}^{(l)}}^{\top} \mathbf{a}_{i-m+1:i}^{(l-1)} + b^{(l)}\right) \end{aligned} \tag{17}$$

其中$\mathbf{W}^{(l)} \in \mathbb{R}^m$为$m$维滤波器，$\mathbf{a}_{i-m+1:i}^{(l-1)} = [\mathbf{a}_{i-m+1}^{(l-1)}, \cdots, \mathbf{a}_i^{(l-1)}]^{\top}$

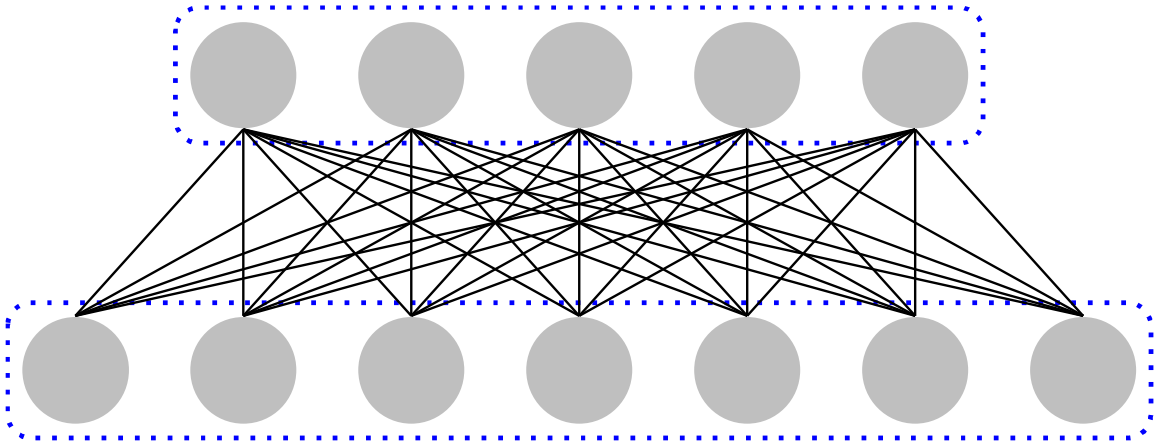
上述公式也可以写为：

$$\mathbf{a}^{(l)} = f_l\left(\mathbf{W}^{(l)} \bigotimes \mathbf{a}^{(l-1)} + b^{(l)}\right) \tag{18}$$

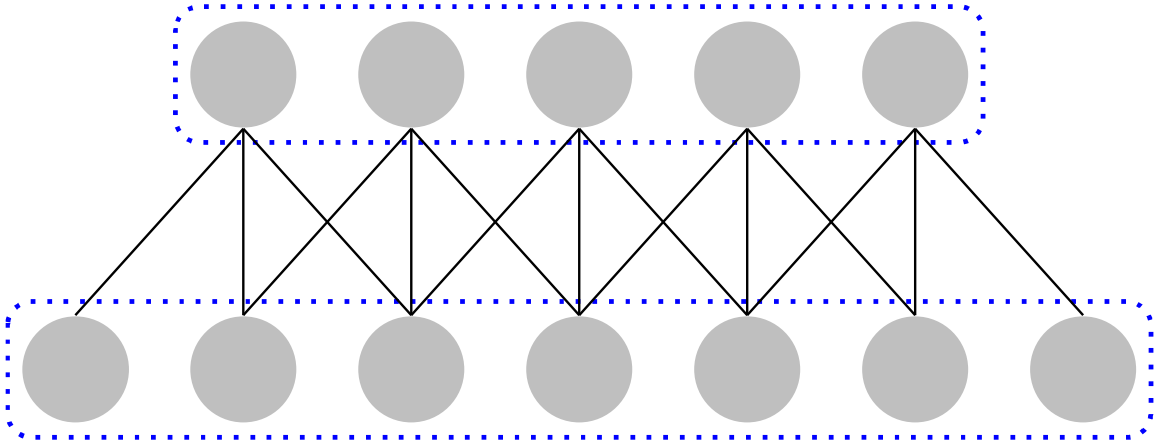
$\bigotimes$表示卷积运算。

从上式可以看出，$\mathbf{W}^{(l)}$对于所有的神经元都是相同的。这即是卷积层的另外一个特性: **权值共享**。在卷积层里，我们只需要$m+1$个参数。另外，第$l+1$层的神经元个数不是任意选择的，而是满足$n^{(l+1)} = n^{(l)} - m + 1$。

全连接层和卷积层



(a) 全连接层



(b) 卷积层

在图像处理中，图像是以二维矩阵的形式输入到神经网络中，因此我们需要二维卷积。假设$\mathbf{X}^{(l)} \in \mathbb{R}^{w_l \times h_l}$和$\mathbf{X}^{(l-1)} \in \mathbb{R}^{w_{l-1} \times h_{l-1}}$分别是第$l$层和第$l-1$层的神经元活性值。$\mathbf{X}^{(l)}$的每一个元素为：

$$\mathbf{X}_{s,t}^{(l)} = f_l \left(\sum_{i=1}^{u} \sum_{j=1}^{v} \mathbf{W}_{i,j}^{(l)} \cdot \mathbf{X}_{s-i+u,t-j+v}^{(l-1)} + b^{(l)} \right) \tag{19}$$

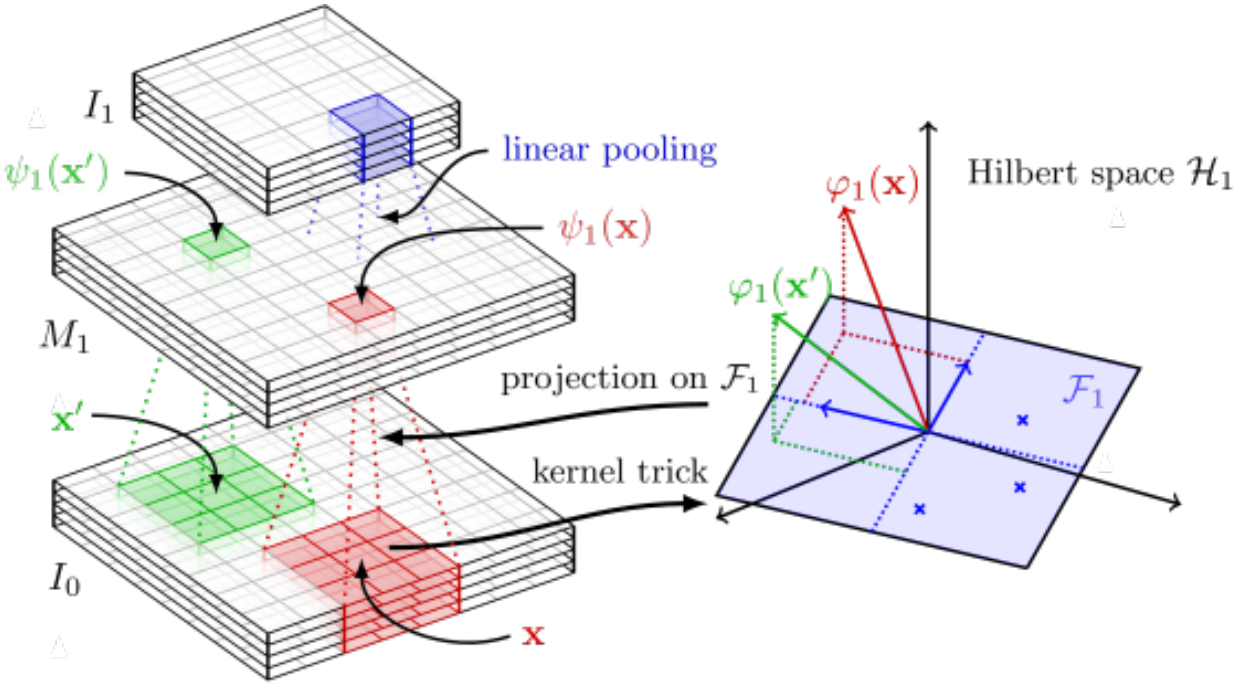
其中$\mathbf{W}^{(l)} \in \mathbb{R}^{u \times v}$为两维滤波器，$b^l$为第$l$层偏置。第$l$层的神经元个数为$w_l \times h_l$，并且$w_l = w_{l-1} - u + 1$, $h_l = h_{l-1} - v + 1$。

(19)式可以写为：

$$\mathbf{X}^{(l)} = f_l \left(\mathbf{W}^{(l)} \otimes \mathbf{X}^{(l-1)} + b^{(l)} \right) \tag{20}$$

为了增强卷积层的**表示能力**，可以使用K 个不同的滤波器来得到K组输出，每一组输出都共享一个滤波器。

如果把滤波器看成一个特征提取器，每一组输出都可以看成是输入图像经过特征抽取后得到的特征表示。因此，卷积神经网络中每一组输出也被称为特征映射(Feature Map)。



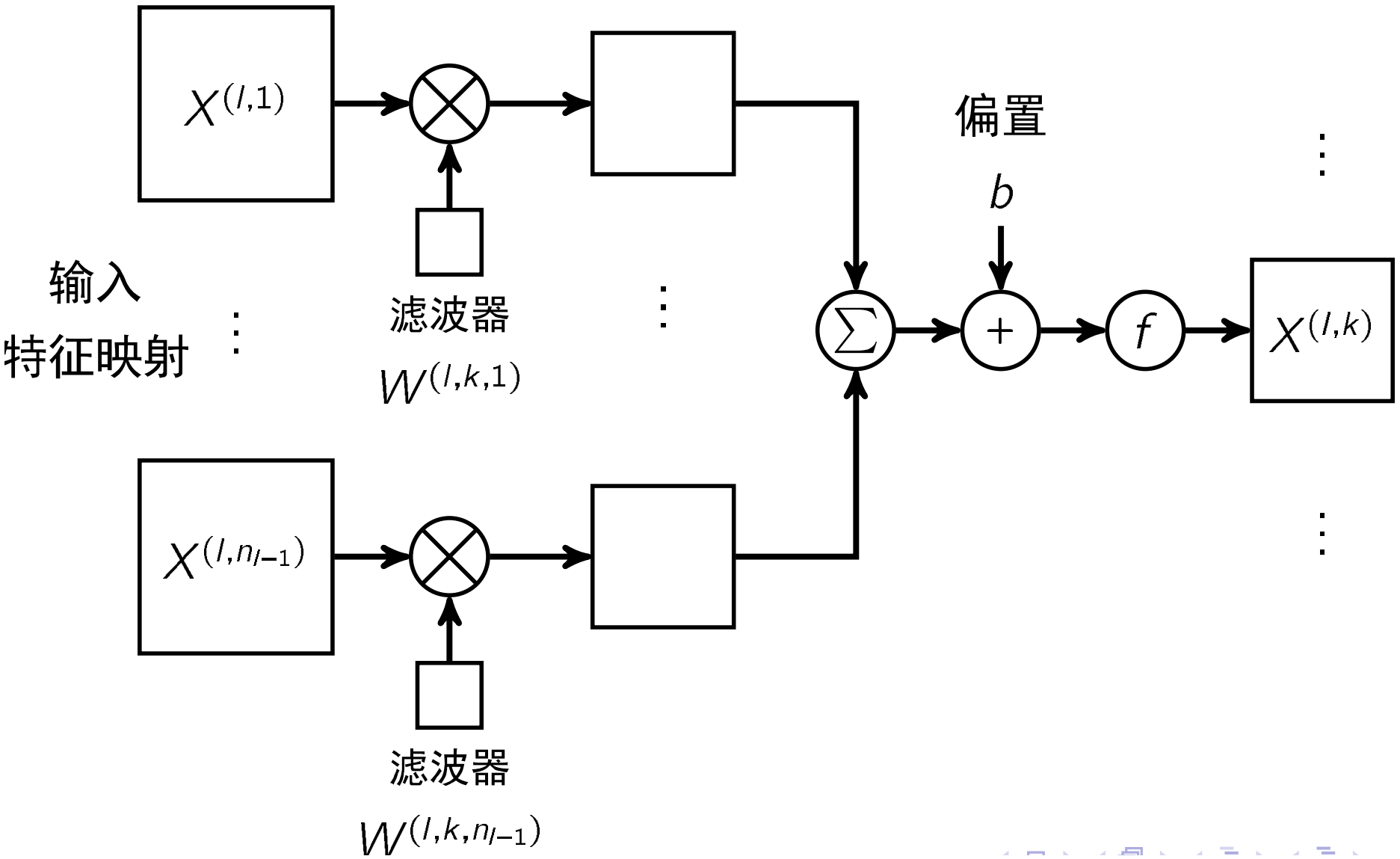
不失一般性，假设第$l-1$层的特征映射组数为n_{l-1}，每组特征映射的大小为$m_{l-1}=w_{l-1}\times h_{l-1}$。则第$l-1$层的总神经元个数：$n_{l-1}\times m_{l-1}$。

记第l层的特征映射组数为n_l。假设第l层的第k组特征映射$\mathbf{X}^{(l,k)}$的输入为第$l-1$层的所有组特征映射。第l层的第k组特征映射$\mathbf{X}^{(l,k)}$为：

$$\mathbf{X}^{(l,k)}=f_l\left(\sum_{p=1}^{n_{l-1}}\left(\mathbf{W}^{(l,k,p)}\otimes\mathbf{X}^{(l-1,p)}\right)+b^{(l,k)}\right) \tag{21}$$

其中，$\mathbf{W}^{(l,k,p)}$表示第$l-1$层的第p组特征向量到第l层的第k组特征映射所需的滤波器。

两维卷积层的映射关系



第l层的每一组特征映射都依赖于第$l-1$层的所有特征映射，即**不同层的特征映射之间是全连接的关系**。事实上，这种全连接关系不是必须的。可以让第l层的每一组特征映射都依赖于**前一层的少数几组**特征映射。

定义一个连接表T来描述不同层的特征映射之间的连接关系。如果第l层的第k组特征映射依赖于前一层的第p组特征映射，则$T_{p,k}=1$，否则为0。

$$\mathbf{X}^{(l,k)}=f_l\left(\sum_{p=1,T_{p,k}=1}^{n_{l-1}}\left(\mathbf{W}^{(l,k,p)}\otimes\mathbf{X}^{(l-1,p)}\right)+b^{(l,k)}\right) \tag{22}$$

假如连接表T的非零个数为K，每个滤波器的大小为$u\times v$，那么共需要$K\times(u\times v)+n_l$参数。

卷积层虽然可以显著减少连接的个数，但是**每一个特征映射的神经元个数**并没有显著减少。此时，如果后面接一个分类器，分类器的输入维数依然很高，很容易出现过拟合。

为了解决这个问题，卷积神经网络一般会在卷积层之后再加上一个池化(Pooling)操作，即子采样(Su b-sampling)，构成一个子采样层。

子采样层可以来大大降低特征的维数，避免过拟合。

对于卷积层得到的一个特征映射$\mathbf{X}^{(l)}$，可以将$\mathbf{X}^{(l)}$划分为多个区域$R_k, (k=1,\cdots,K)$，这些区域可以重叠，也可以不重叠。一个子采样函数down$(\cdot)$定义为：

$$\mathbf{X}^{(l+1)} = f\left(\mathbf{Z}^{(l+1)}\right) = f\left(\mathbf{W}^{(l+1)} \cdot \text{down}(\mathbf{X}^{l}) + b^{(l+1)}\right) \tag{23}$$

其中down$(\mathbf{X}^l)$是指子采样后的特征映射，$\mathbf{W}^{(l+1)}$和$b^{(l+1)}$分别是可训练的权重和偏置参数。

子采样函数down$(\cdot)$一般是取区域内所有神经元的最大值（Maximum Pooling）或平均值（Average Pooling）。

$$\text{pool}_{\max}(R_k) = \max_{i\in R_k} a_i \qquad \text{pool}_{\text{avg}}(R_k) = \frac{1}{|R_k|}\sum_{i\in R_k} a_i \tag{24}$$

子采样的作用还在于可以使得下一层的神经元对一些小的形态改变保持不变性。

卷积神经网络示例：LeNet-5

LeNet-5虽然提出时间比较早，但是是一个非常成功的神经网络模型。基于LeNet-5的手写数字识别系统在90年代被美国很多银行使用，用来识别支票上面的手写数字。

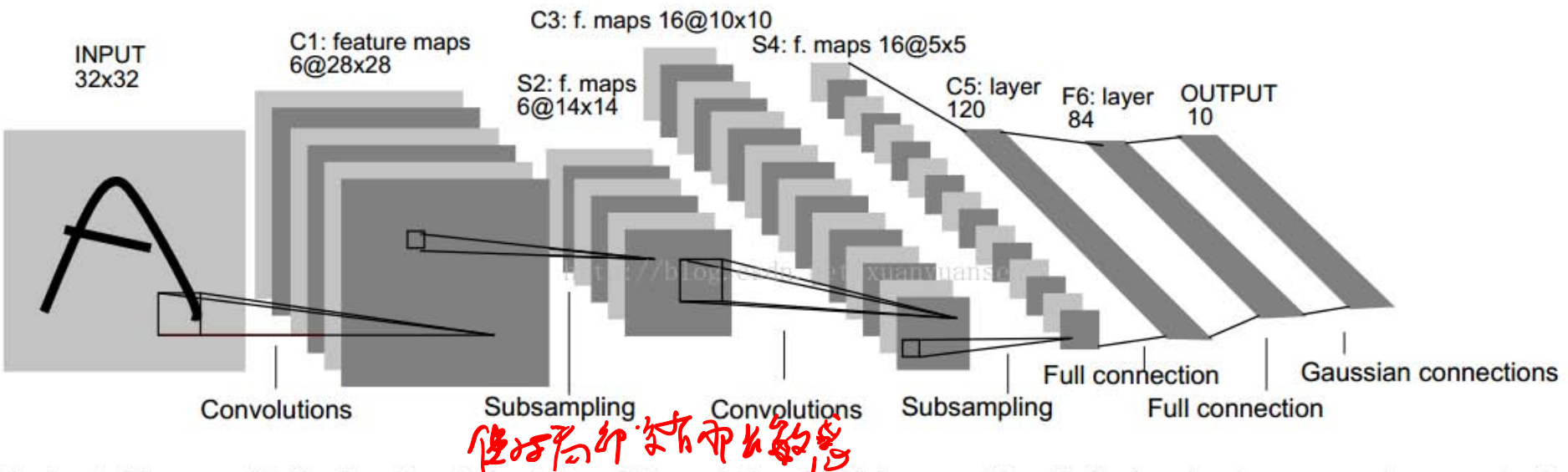


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

不计输入层，LeNet-5共有7层，每一层的结构为：

1. 输入层：输入图像大小为$32 \times 32 = 1024$。
2. C1层：这一层是卷积层。滤波器的大小是$5 \times 5 = 25$，共有6个滤波器。得到6组大小为$28 \times 28 = 784$的特征映射。
3. S2层：这一层为子采样层。由C1层每组特征映射中的2×2 邻域点次采样为1个点，也就是4个数的平均。

1. C3层：这一层是卷积层。由于S2层也有多组特征映射，需要一个连接表来定义不同层特征映射之间的依赖关系。LeNet-5的连接表如下图所示。这样的连接机制的基本假设是：C3层的最开始的6个特征映射依赖于S2 层的特征映射的每3个连续子集。接下来的6个特征映射依赖于S2层的特征映射的每4个连续子集。再接下来的3个特征映射依赖于S2层的特征映射的每4个不连续子集。最后1个特征映射依赖于S2层的所有特征映射。这样共有16个滤波器，大小是$5 \times 5 = 25$。得到16组大小为$10 \times 10 = 100$的特征映射。
2. S4层：这一层是一个子采样层，由2×2邻域点次采样为1个点，得到16组5×5大小的特征映射。

1. C5层：是一个卷积层，得到120组大小为1×1的特征映射。每个特征映射与S4层的全部特征映射相连。有$120 \times 16 = 1920$ 个滤波器，大小是$5 \times 5 = 25$。
2. F6层：是一个全连接层，有84个神经元。
3. 输出层：输出层由10个欧氏径向基函数(Radial Basis Function，RBF)函数组成。

连接表

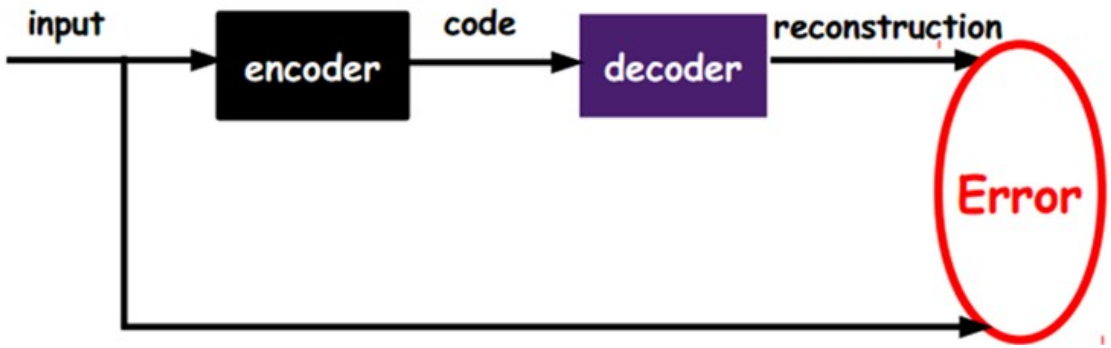
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

图: LeNet-5 中 C3 层的连接表。图片来源：[LeCun et al., 1998]

Content

给定无标签数据，用非监督学习学习特征

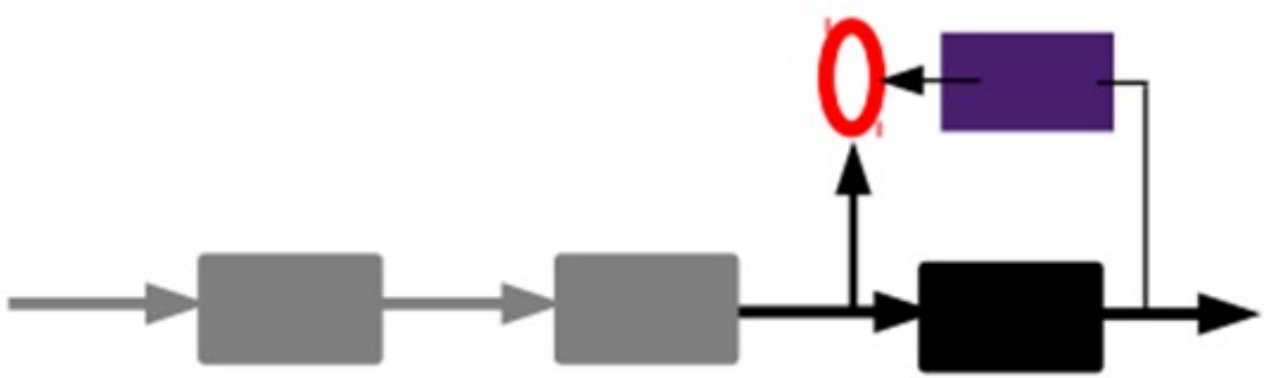
- 给定一个神经网络，假设其输出与输入是相同的，然后训练调整其参数，得到每一层的连接权重。
- 自然地，可以得到输入的几种不同表示(每一层代表一种特征表示)，这些表示就是特征
- 自动编码器就是一种尽可能复现输入信号的神经网络



通过调整编码器和解码器的参数，使得重构误差最小，得到输入信号的一个表示，即编码。

通过编码器产生特征，逐层训练

- 将第一层输出的编码当成第二层的输入信号，同样最小化重构误差，就会得到第二层的参数，并且得到第二层输入的编码，得到原输入信息的第二个表达了。
- 其他层依此方法炮制



有监督微调

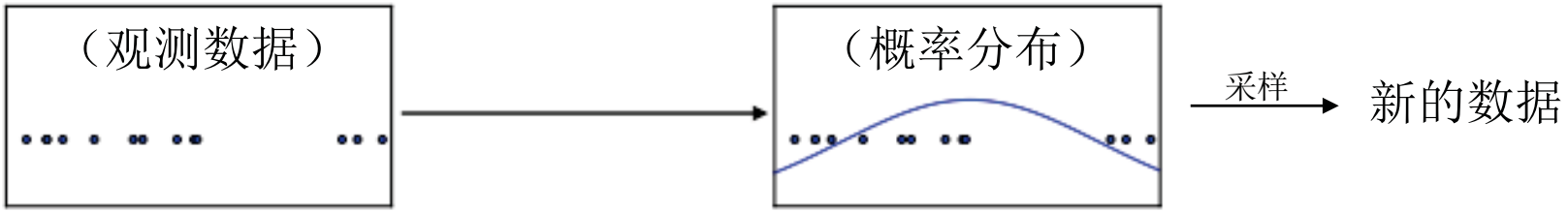
- 经过上面的方法，可以得到多层表示。
- 每一层都会得到原始输入的不同的表达。
- 需要多少层需要根据试验调整。
- 此时，自动编码器只是学习到了一个可以良好地代表输入数据的特征，该特征可最大程度上代表原输入信号。

需要将最后层的特征编码输入到分类器，通过有标签样本，通过监督学习进行微调。

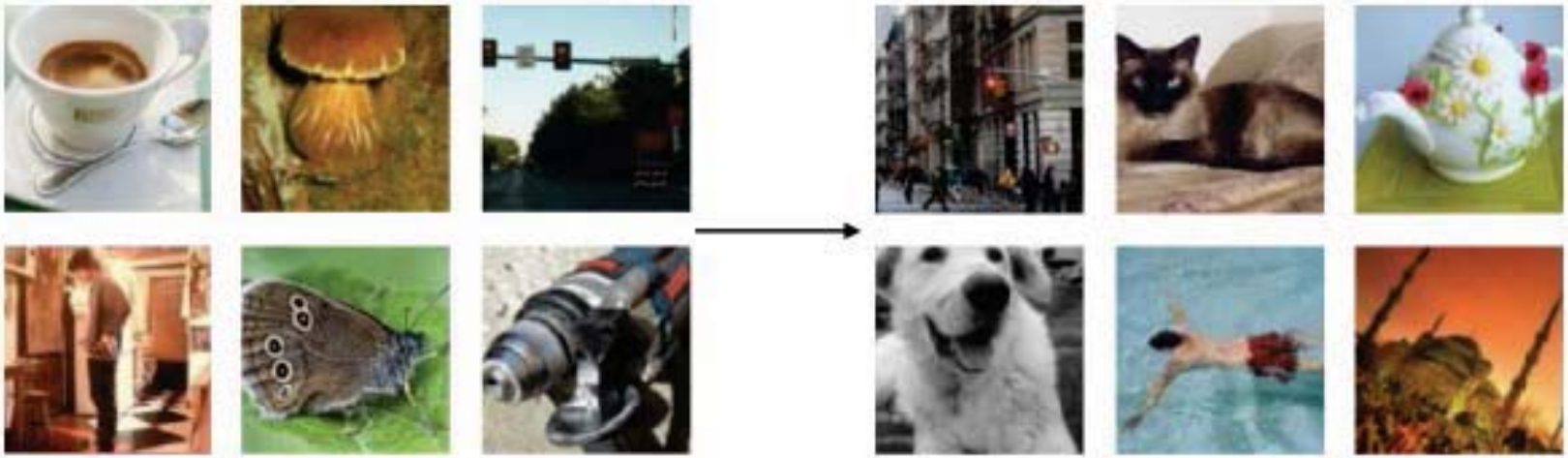
在自动编码器的基础上加上ℓ_1的正则化限制，可得到稀疏自动编码器算法。

Content

概率生成模型



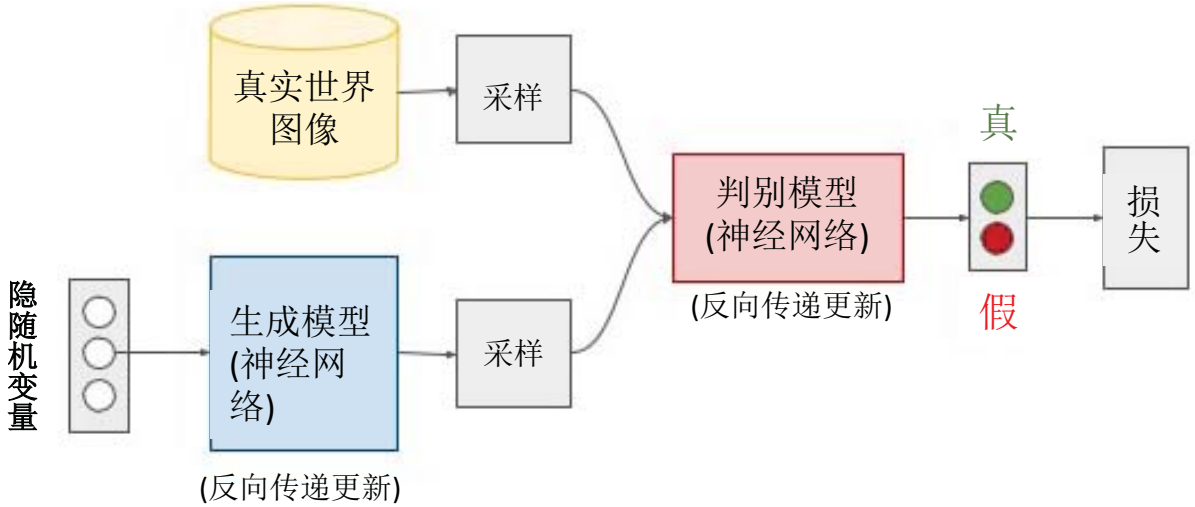
图像生成

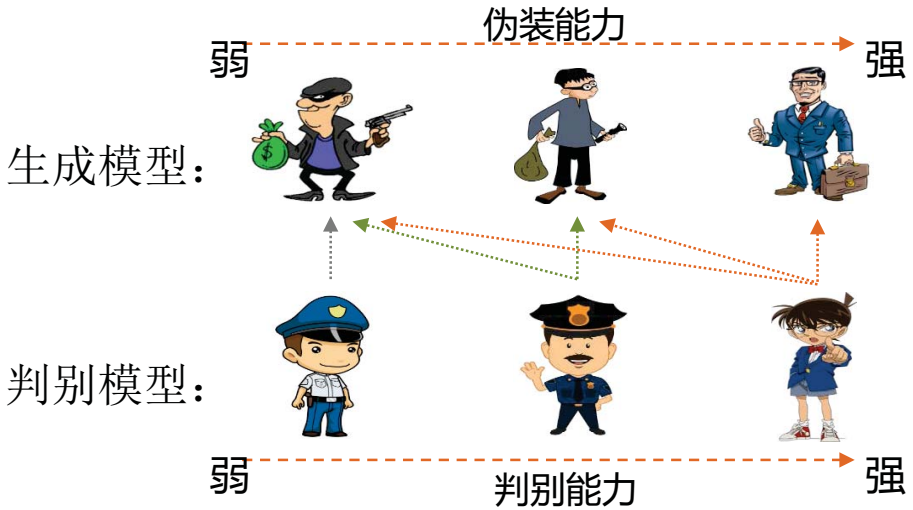


训练样本 (真实世界样本)

生成样本 (模型生成样本)

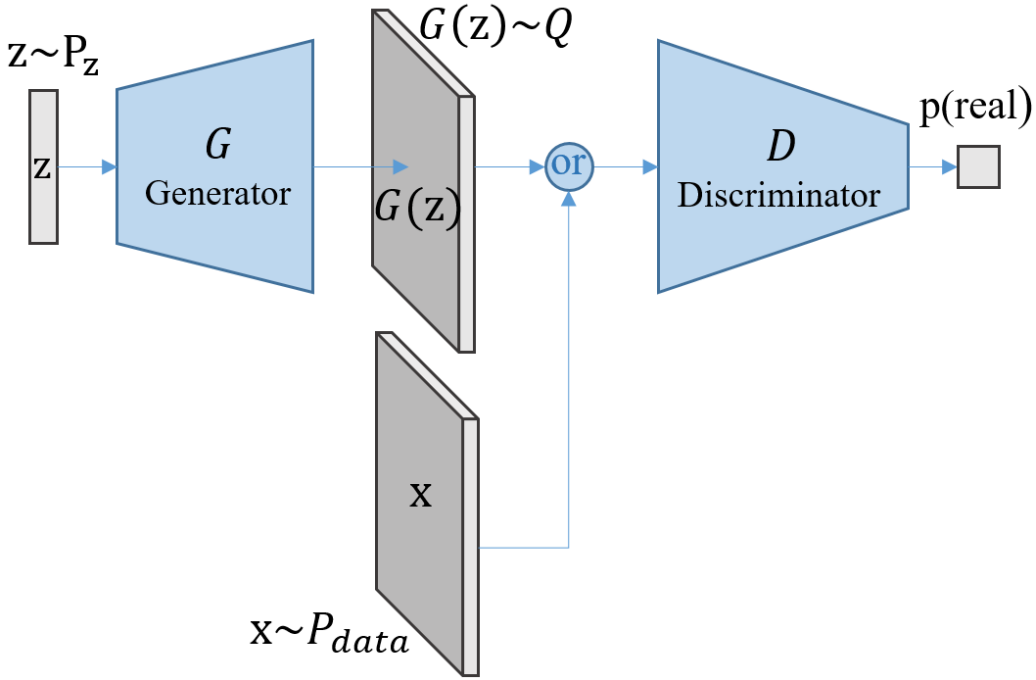
GAN的基本框架


真实世界
图像
采样
判别模型
(神经网络)
真
损失
隐随机变量
生成模型
(神经网络)
采样
(反向传递更新)
假
(反向传递更新)


伪装能力
弱
强
生成模型：
判别模型：
弱
判别能力
强

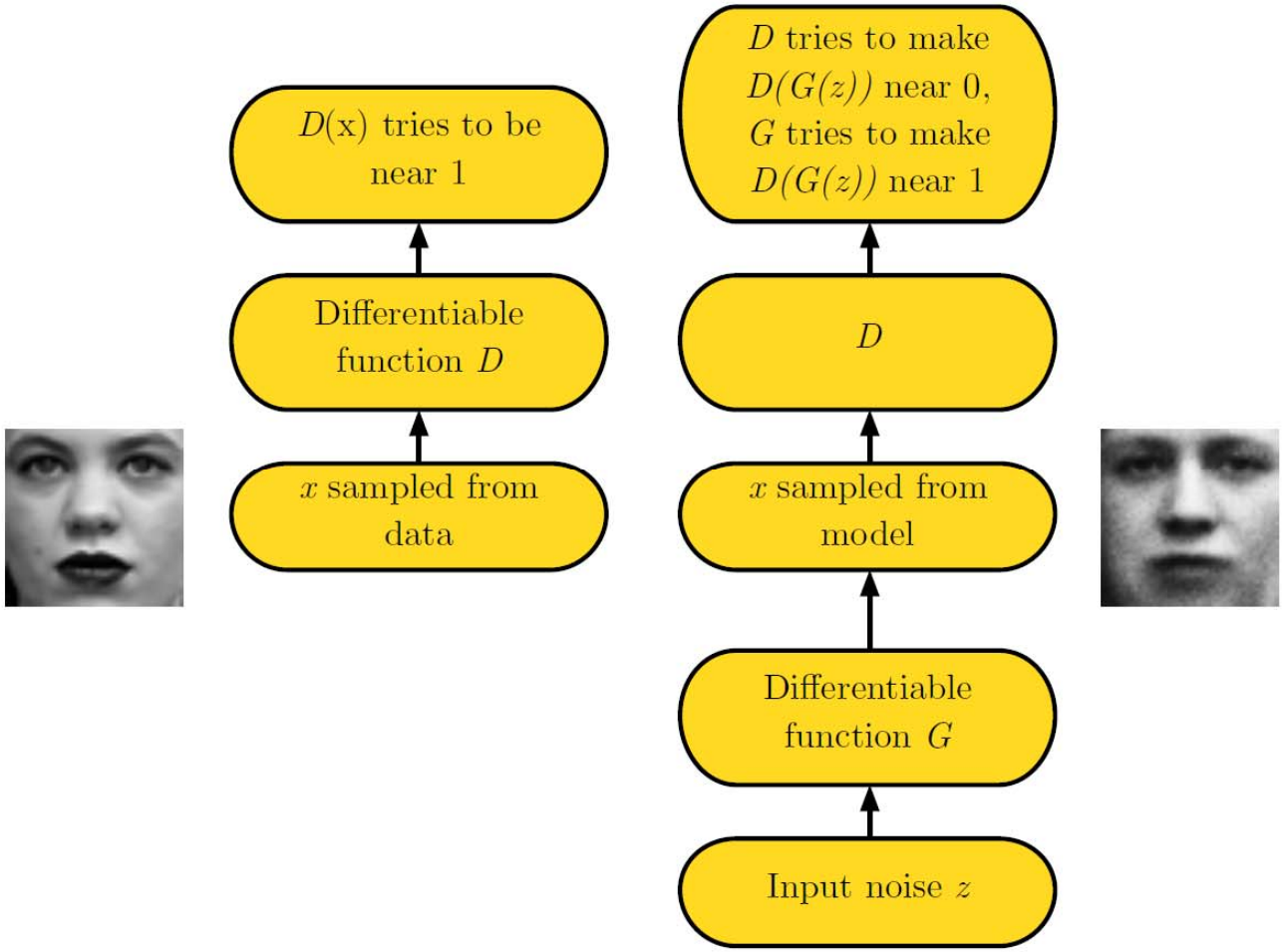
生成对抗网络(GAN)：框架

G: 生成函数
　生成与真实样本相似的样本
　z: 噪声
D: 判别函数
　区分样本来自于真实样本还是来自于生成样本
　D(x): x来自于真实样本的概率



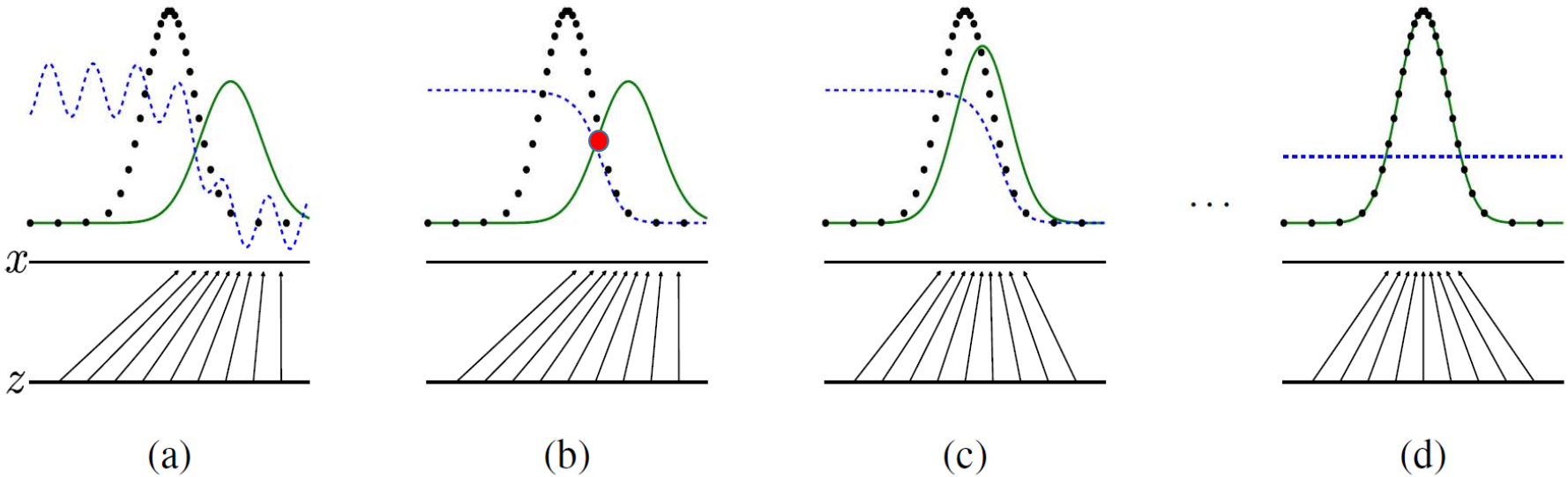
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

生成对抗网络(GAN)：框架



(Goodfellow 2016)

生成对抗网络(GAN)：训练过程示意



z: 噪声，x: 样本，黑色点线: 真实样本的分布，绿色实线: 生成样本的分布，蓝色虚线: 判别函数
(a): 随机初始化判别函数和生成函数
(b): 优化判别函数，增强其判别样本是否来自于真实样本的能力，例如图中的交点
(c): 优化生成函数，使其具有更强的伪装真实样本的能力
(d): 不断交替优化判别函数和生成函数，达到平衡点

生成对抗网络(GAN)：随机梯度下降训练

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.