机器学习

李成龙

安徽大学人工智能学院
“多模态认知计算”安徽省重点实验室
合肥综合性国家科学中心人工智能研究院

内容安排

- 什么是机器学习
- 机器如何学习
- 如何让机器学习的更好
- 为什么机器能学习

内容安排

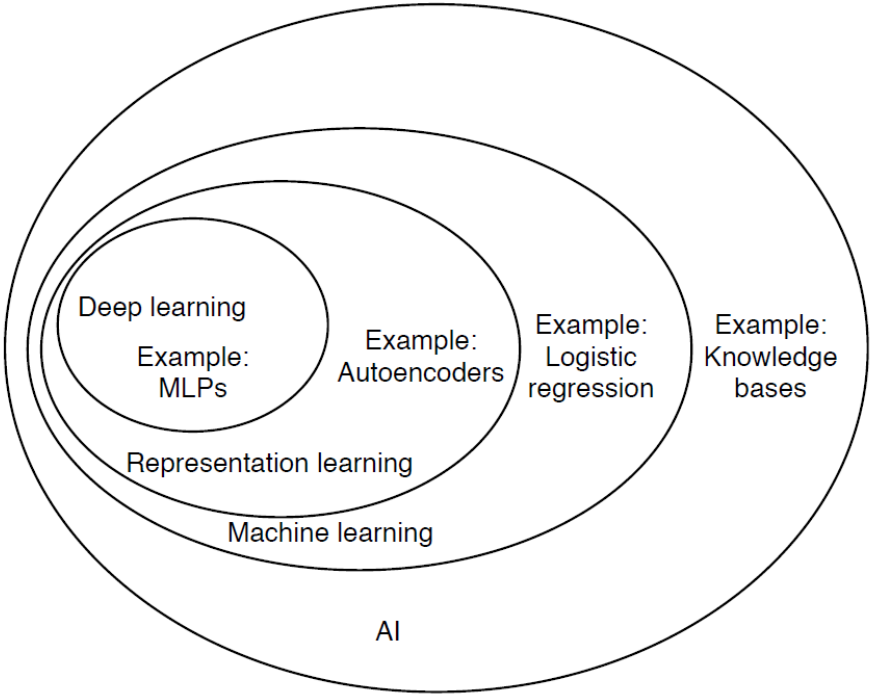
- **机器如何学习**
  - 有监督学习
    - 感知机
    - 支持向量机
    - 朴素贝叶斯分类
    - 决策树
    - 集成学习（Bagging算法与随机森林、Boosting算法）
    - 线性回归
    - 逻辑回归
    - Softmax回归
    - 神经网络与深度学习
  - 无监督学习
    - 聚类
    - 主成分分析

本节目录

本节目录

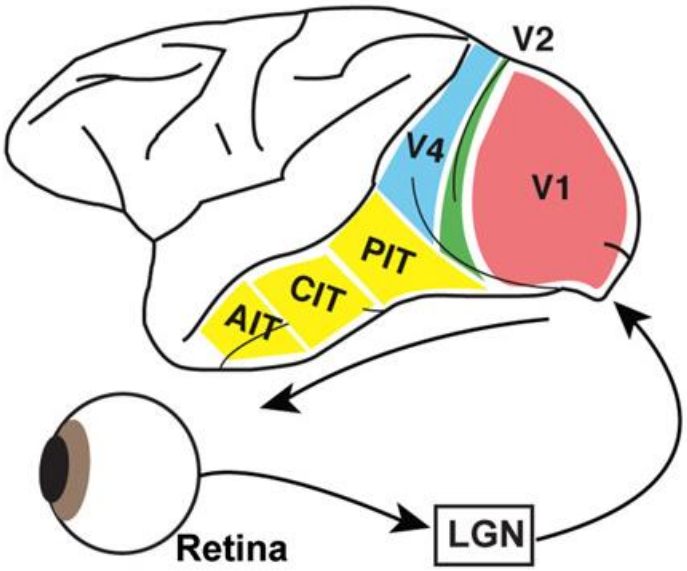
概述

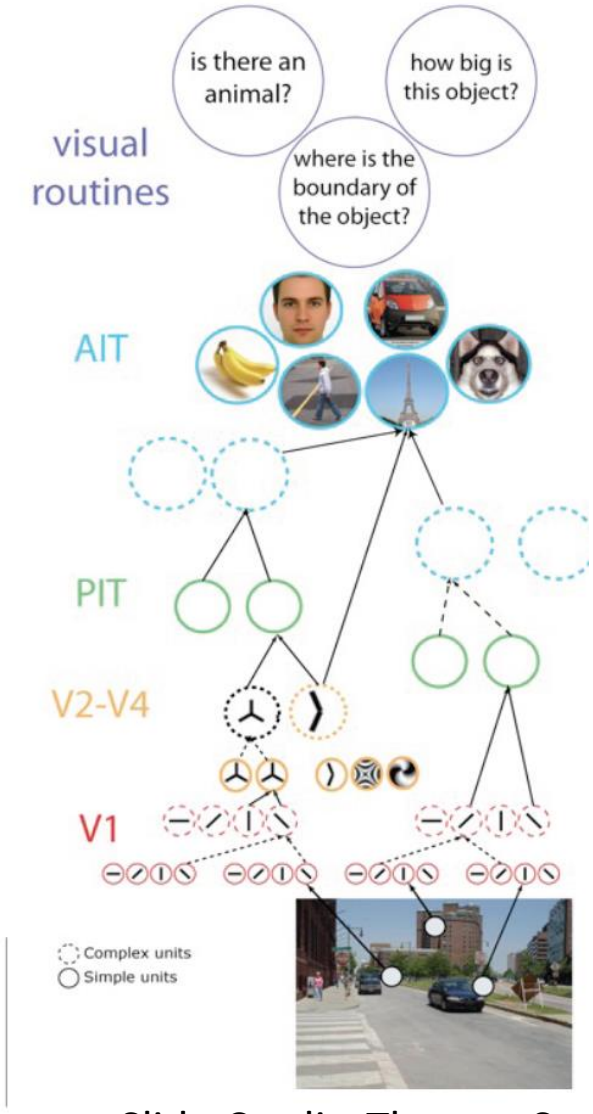
- **定义**
  - **机器学习**：研究如何通过计算的手段，利用经验来改善系统自身的性能，从而在计算机上从数据中产生“模型”，用于对新的情况给出判断
  - **深度学习**：脑启发的学习多层表示或抽象的机器学习算法



概述

- 定义

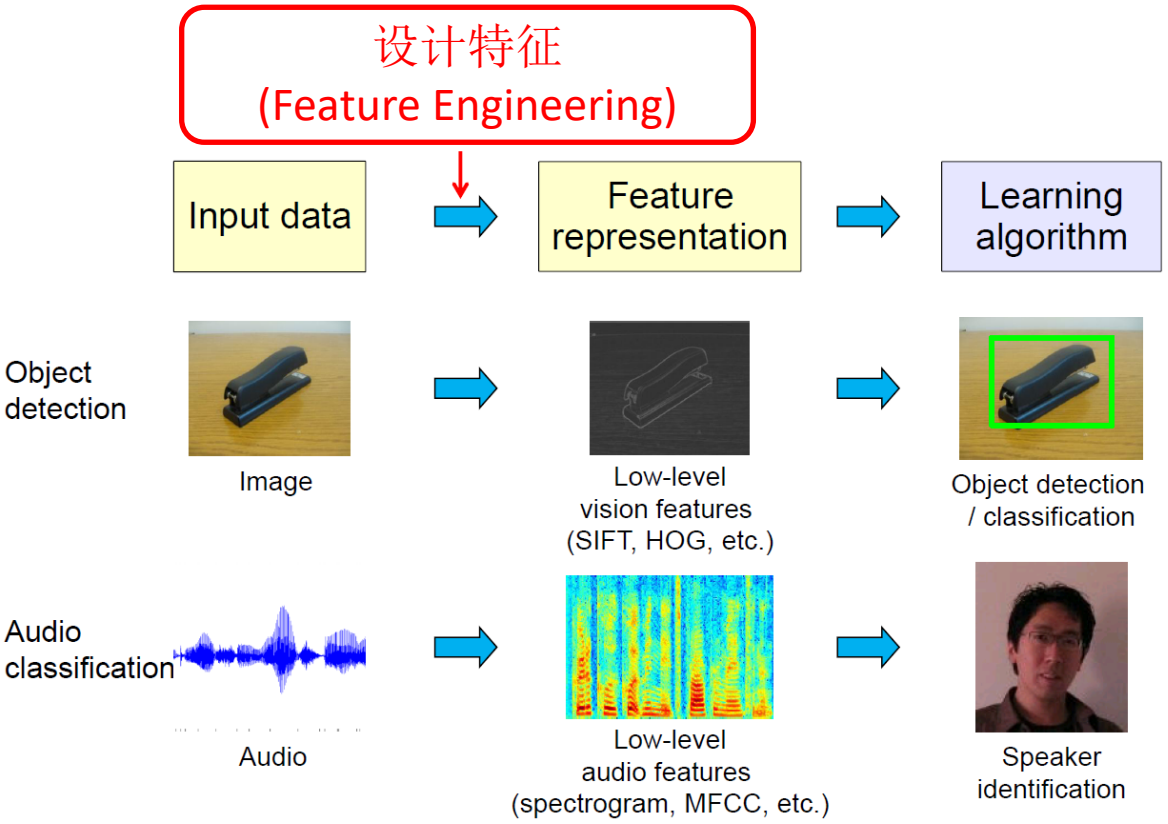




Slide Credit: Thomas Serre

概述

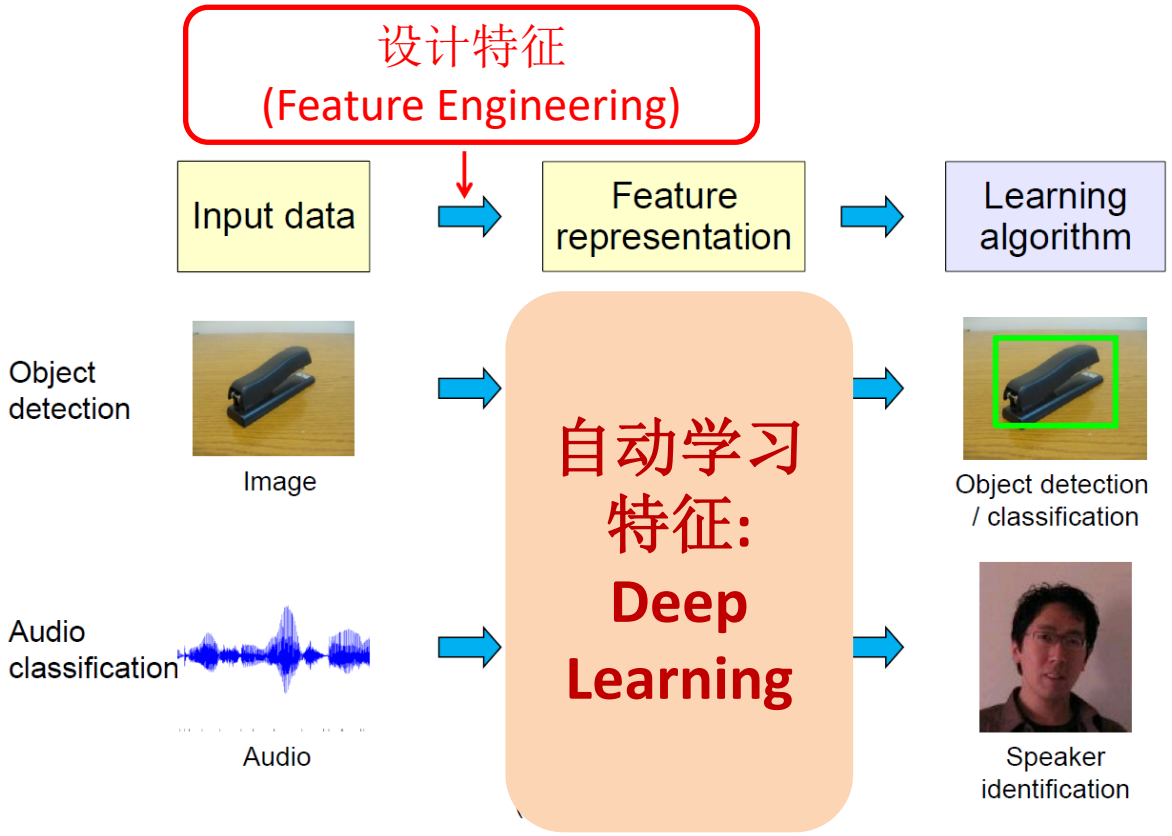
- **自动学习特征**
  - **传统做法**：给定输入数据，通过精心设计适用于任务的特征估计学习器参数



Slide Credit: Honglak Lee

概述

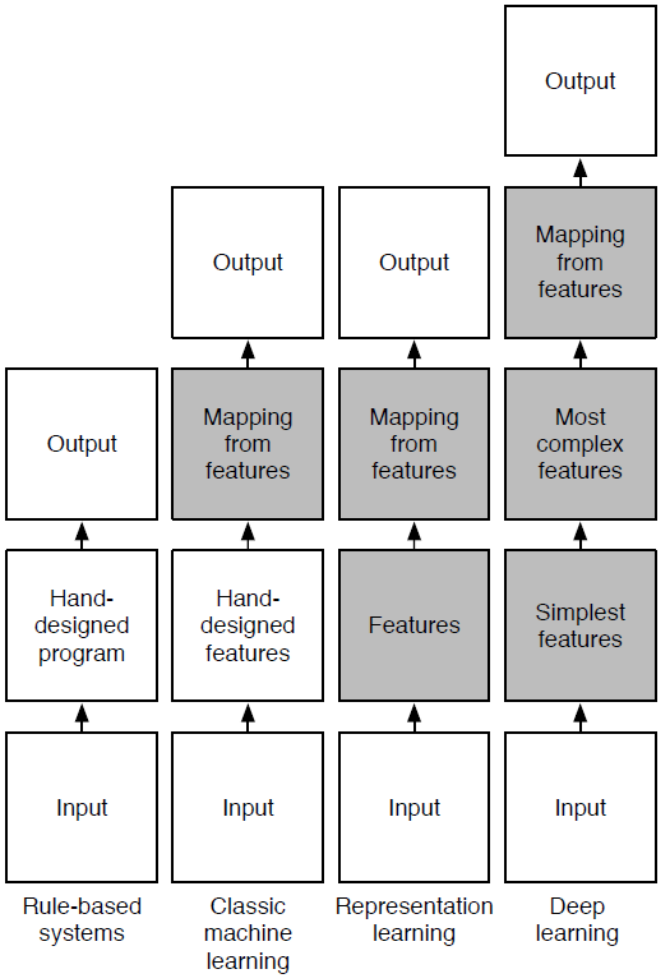
- **自动学习特征**
  - **深度学习**：设计面向任务的损失函数，通过数据驱动的方式自动学习特征和学习器参数



Slide Credit: Honglak Lee

概述

- 自动学习特征
  - 深度学习：



概述

- **深度学习与浅层学习**
  - 神经网络模型作为一个通用逼近函数具有非常强大的拟合能力，由于随着网络层数的加深，网络模型通常会变得难以收敛且计算量巨大，故具有强大拟合能力的浅层网络是很长一段时期最主要的研究对象
  - 对于大多数神经网络模型而言，其根本目的在于模拟适用于某个任务的映射函数，如针对于二分类问题的MLP模型其实相当于从样本输入到样本标签类别的映射函数

概述

- **深度学习与浅层学习**
  - 从理论上看，神经元数目足够多的神经网络模型可以逼近任意函数
    - 通用近似定理[Cbbenko, 1989, Hornik et al., 1989]：令φ是一个非常数、有界、单调递增的连续函数，I_d是一个d维的单位超立方体$[0,1]^d$，$C(I_d)$是定义在I_d上的连续函数集合。对于任何一个函数f$\in C(I_d)$，存在一个整数m和一组实数v_i，$b_i \in R$以及实数向量$\boldsymbol{w}_i \in R^d$，以至于我们可以定义函数

$$F(\boldsymbol{x}) = \sum_{i=1}^{m} v_i \varphi(\mathbf{w}^T \boldsymbol{x}_i + bi)$$

作为函数f的近似实现，即

$$\left|\left|F(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x})\right|\right| < \varepsilon, \boldsymbol{x} \in I_d$$

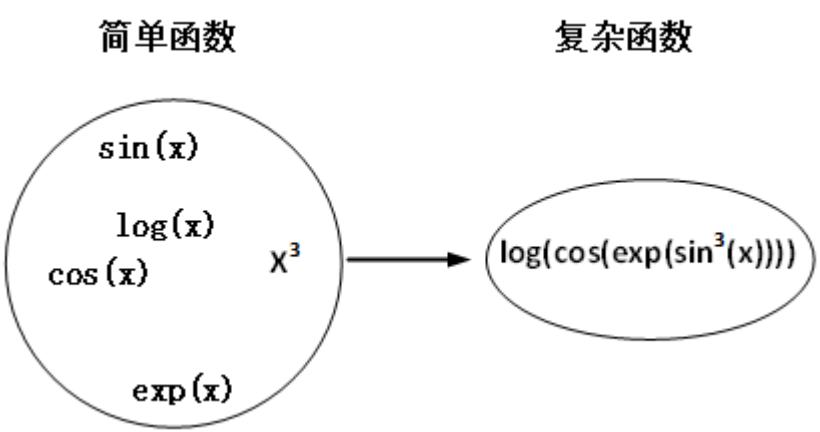
其中$\varepsilon > 0$是一个很小的正数

概述

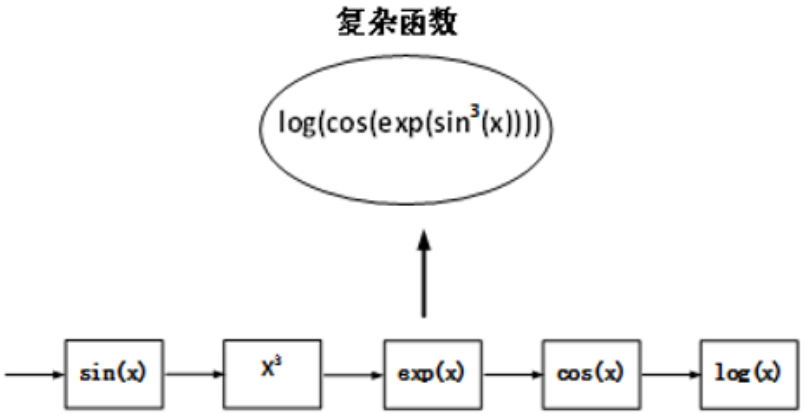
- **深度学习与浅层学习**
  - 神经网络模型增加神经网络的隐含层层数比直接增加某一隐含层的结点数目更能提高模型的拟合能力，这是因为添加隐含层不仅增加了模型的数据处理神经元数目，还添加了一层嵌套的非线性映射函数

概述

- **深度学习与浅层学习**
  - 以使用简单函数逼近复杂函数过程为例。下图表示使用简单函数逼近复杂函数的一个简单实例，若使用单层多结点模型逼近这一复杂函数，则表示形式通常较为复杂，如下图所示。若用多层模型，则可较为简单的表示该复杂函数



简单函数对复杂函数的逼近



多层模型表示复杂函数

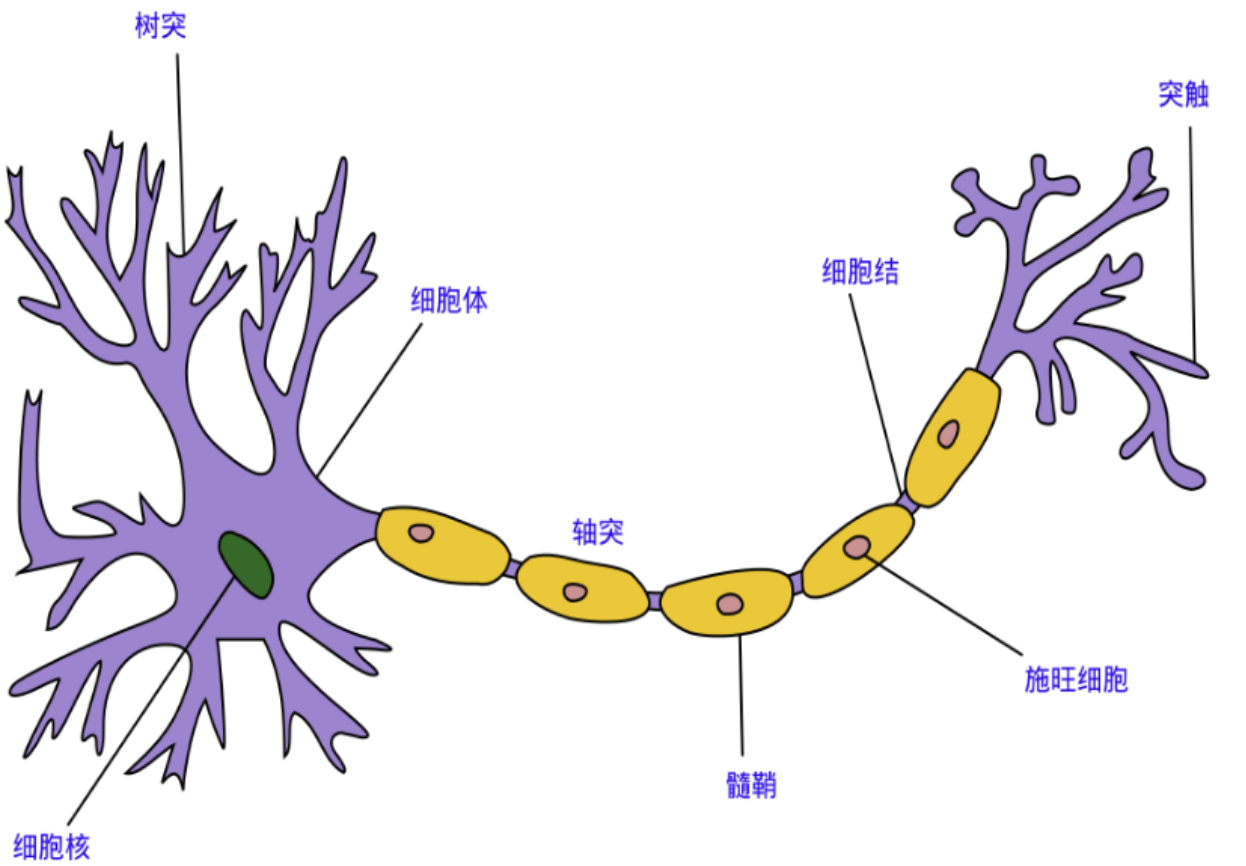
概述

- **深度学习与浅层学习**
  - 数据处理层数较少的神经网络模型容量较低，基于此类模型的机器学习一般统称为浅层学习
  - 虽然从理论上讲浅层学习模型可以逼近任意函数，但其模型容量或灵活性远不及具有较深层次的网络模型，难以满足对复杂任务求解的需求
  - 计算机硬件巨大进步和大数据技术的发展使得对较深层次网络模型的训练构造成为可能，可通过深度学习技术构造深度网络模型用于解决比较复杂的实际问题

本节目录

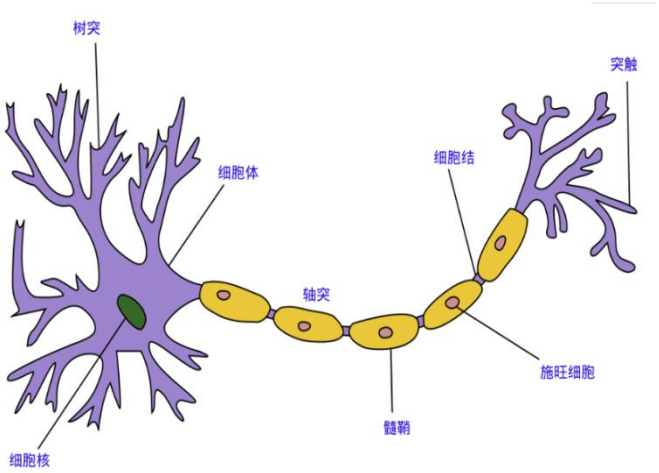
人工神经元

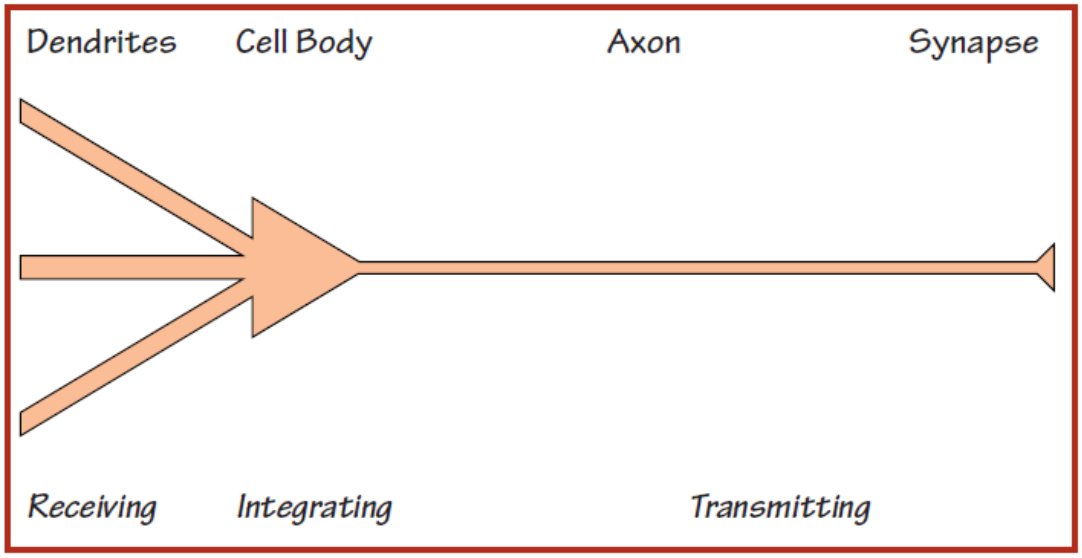
- 生物神经元



人工神经元

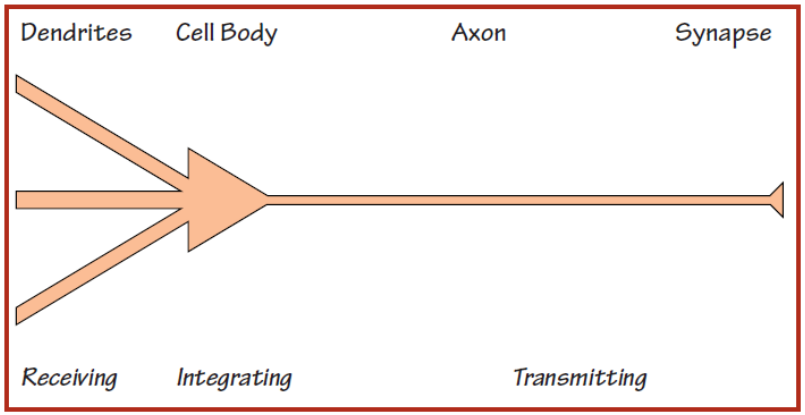
- 生物神经元

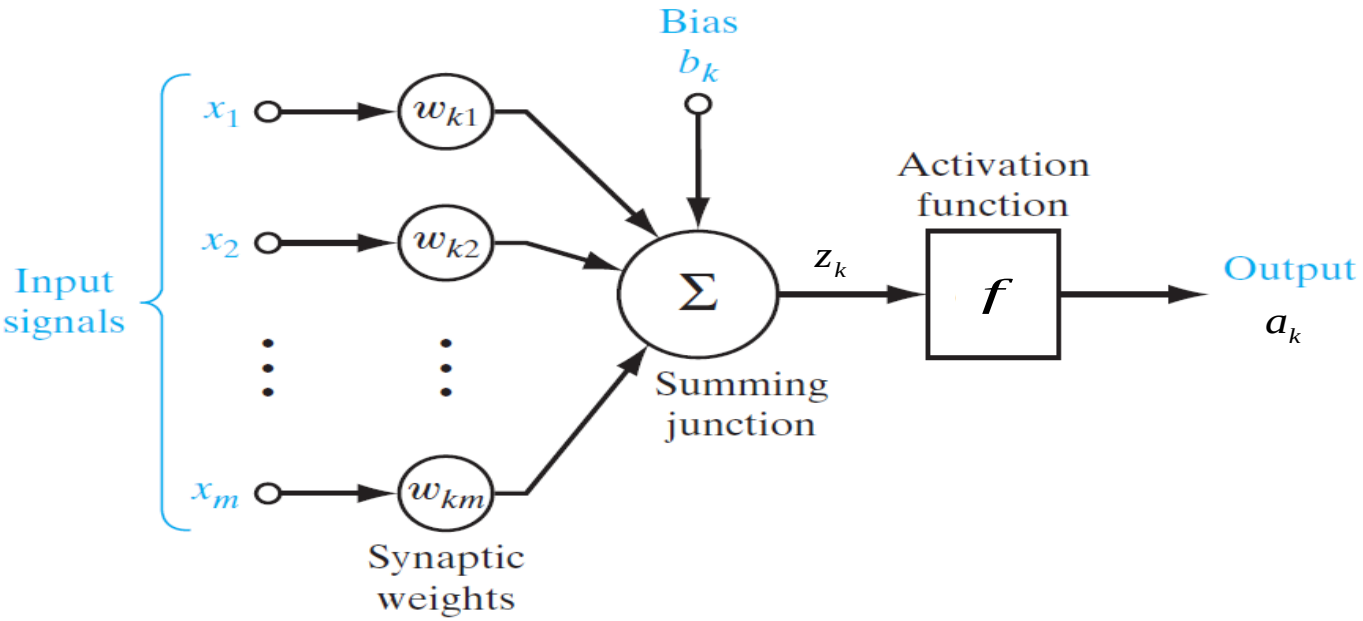

树突
细胞体
轴突
细胞结
突触
施旺细胞
髓鞘
细胞核


Dendrites
Cell Body
Axon
Synapse
Receiving
Integrating
Transmitting

人工神经元

- 人工神经元


Dendrites
Cell Body
Axon
Synapse
Receiving
Integrating
Transmitting


Bias
b_k
x_1
x_2
x_m
w_{k1}
w_{k2}
w_{km}
Input signals
Synaptic weights
Σ
Summing junction
z_k
Activation function
f
Output
a_k

人工神经元

- **激活函数**
  - 激活函数f的作用是对神经元增加非线性，通常将输出范围限制在[0,1]或[−1,1]。常用的激活函数主要有阈值函数、Sigmoid函数等。如下分段函数就是一种常用阈值函数：

$$f(t)=\begin{cases}1\,,t\geq 0\\0\,,t<0\end{cases}$$

  - 阈值函数的输出只能取0或1，分别表示该神经元处于抑制或兴奋状态，故通常称该激活函数为单极性阈值函数
  - 若将神经元处于抑制状态表示为-1，即有：

$$f(t)=\mathrm{sgn}(t)=\begin{cases}1\,,t\geq 0\\-1\,,t<0\end{cases}$$

  - 这种阈值函数则称之为双极性阈值函数

人工神经元

- **激活函数**
  - Sigmoid函数是一种最常用的激活函数，可将人工神经元的输出限制在区间(0,1)内

$$f(t) = \text{Sigmoid}(t) = \frac{1}{1 + e^{-t}}$$

  - 可对Sigmoid函数进行适当改造以调整输出范围，获得新的激活函数。例如，令：

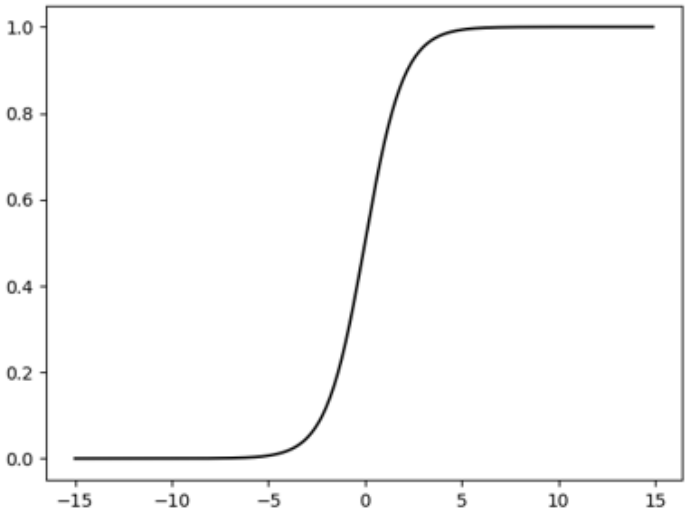
$$f(t) = 2\text{Sigmoid}(t) - 1$$

  - 则可得到一个输出范围是$(-1,1)$的激活函数，通常称该激活函数为tanh函数，tanh函数的具体形式如下：

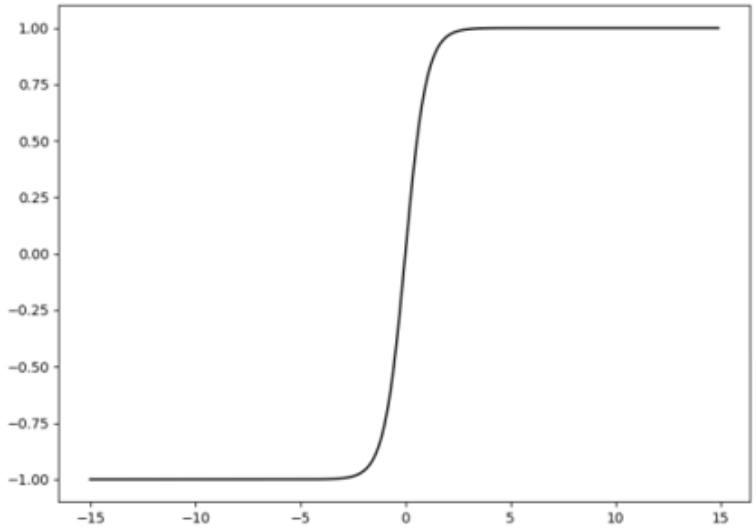
$$f(t) = \tanh(t) = \frac{e^{t} - e^{-t}}{e^{t} + e^{-t}}$$

人工神经元

- **激活函数**
  - tanh函数将Sigmoid函数图像在竖直方向上拉伸了两倍并向下平移了一个单位，有时会更便于进行模型参数求解（均值为0）



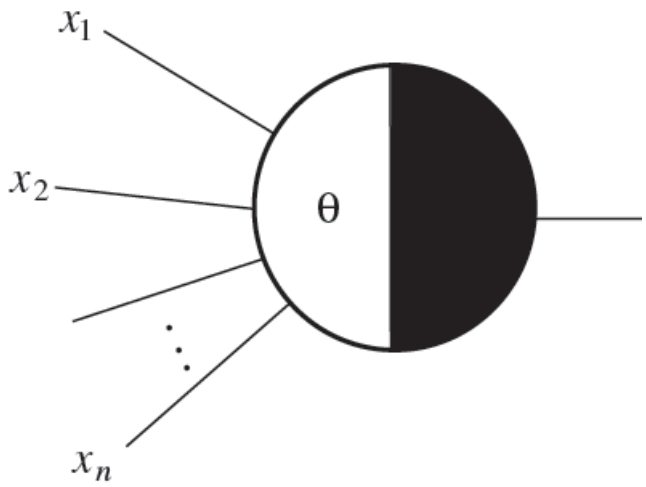
Sigmoid激活函数



tanh激活函数

人工神经元

- 阈值逻辑单元TLU

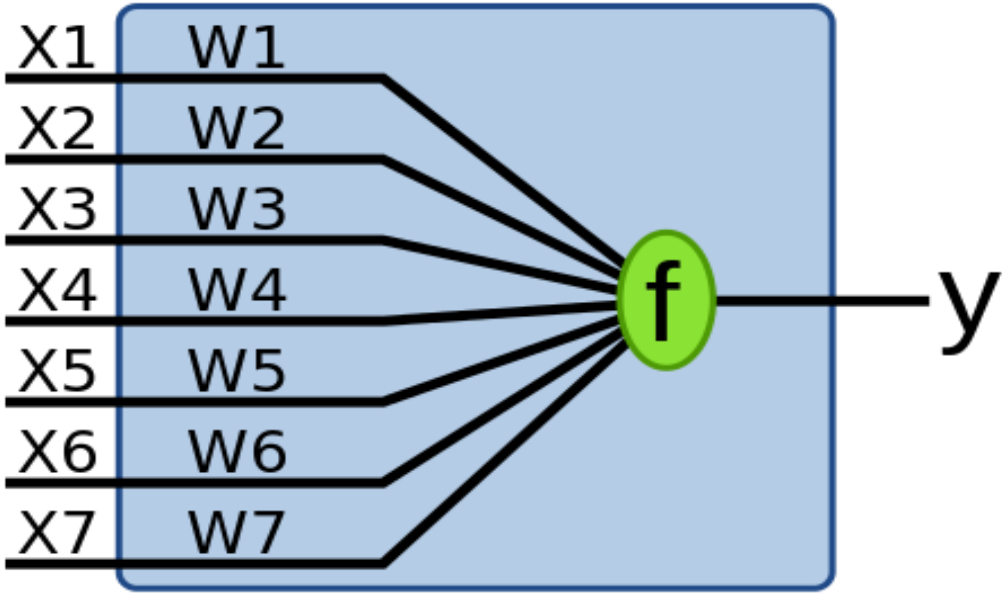


$$y = \begin{cases} 1 & w\sum_{i=1}^{n} x_i \geq \theta \\ 0 & otherwise \end{cases}$$

McCulloch–Pitts,1943

人工神经元

- 感知机（Perceptron）



$$y = f(\mathbf{x};\mathbf{w},b) = \begin{cases} 1 & \mathbf{w}^T\mathbf{x}+b \geq 0 \\ 0 & \mathbf{w}^T\mathbf{x}+b < 0 \end{cases}$$

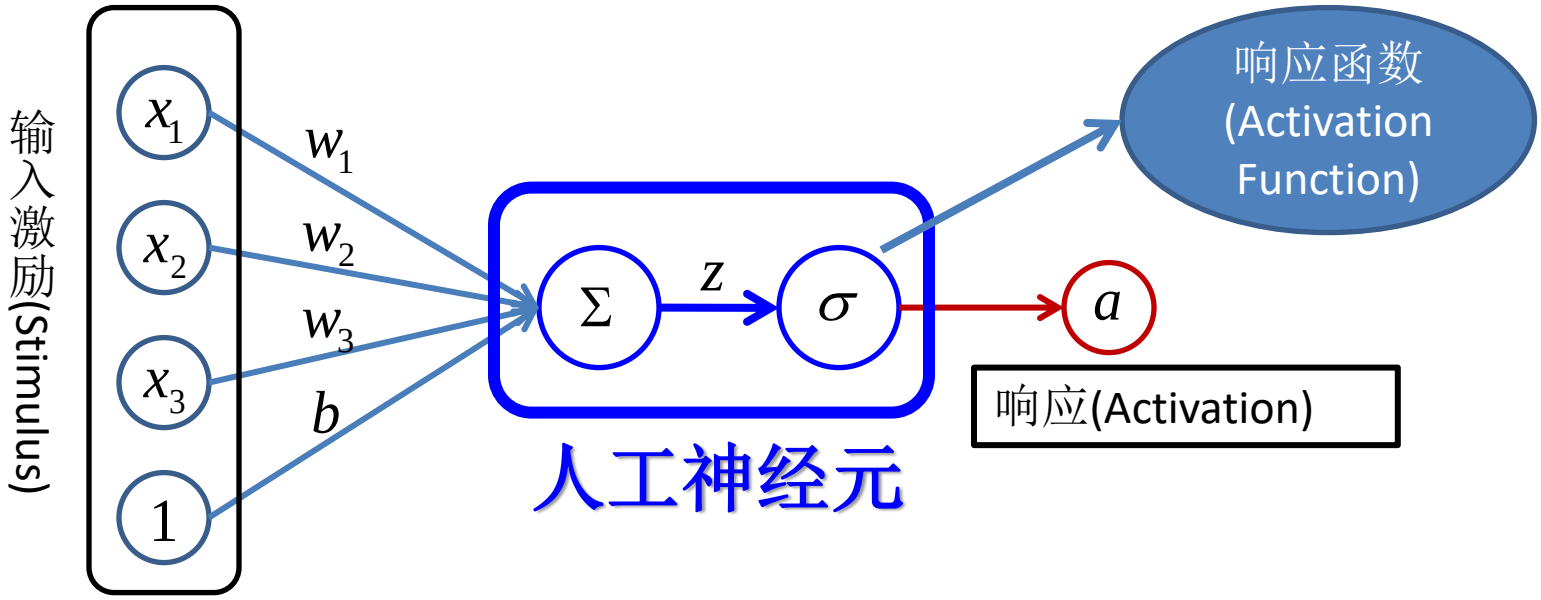
Rosenblat,1958

人工神经元

- **逻辑回归（Logistic Regression）**

$$a = P(y=1|\mathbf{x}) = \sigma\left(\mathbf{w}^T\mathbf{x}+b\right) = \frac{1}{1+\exp\left[-\left(\mathbf{w}^T\mathbf{x}+b\right)\right]}$$

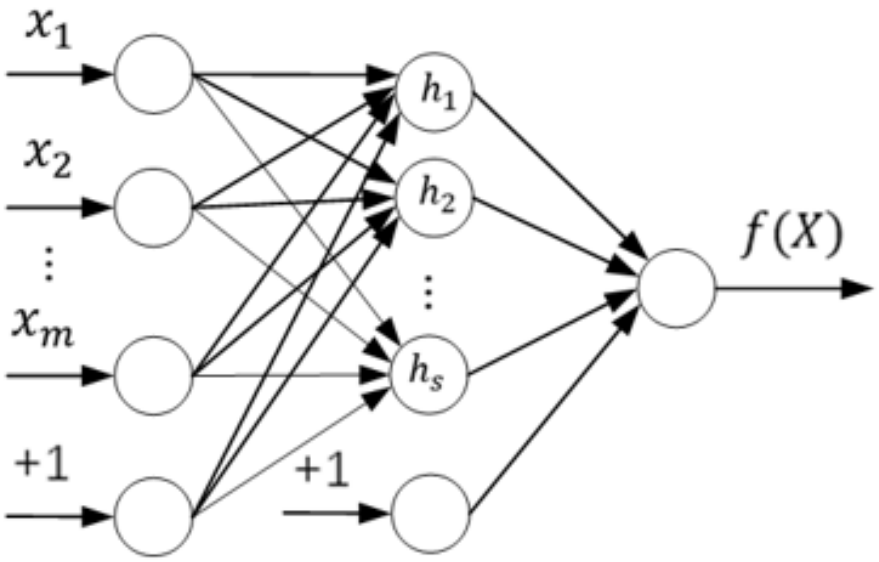
$$\mathbf{x} \to z = \mathbf{w}^T\mathbf{x}+b \to a = \sigma(z)$$



本节目录

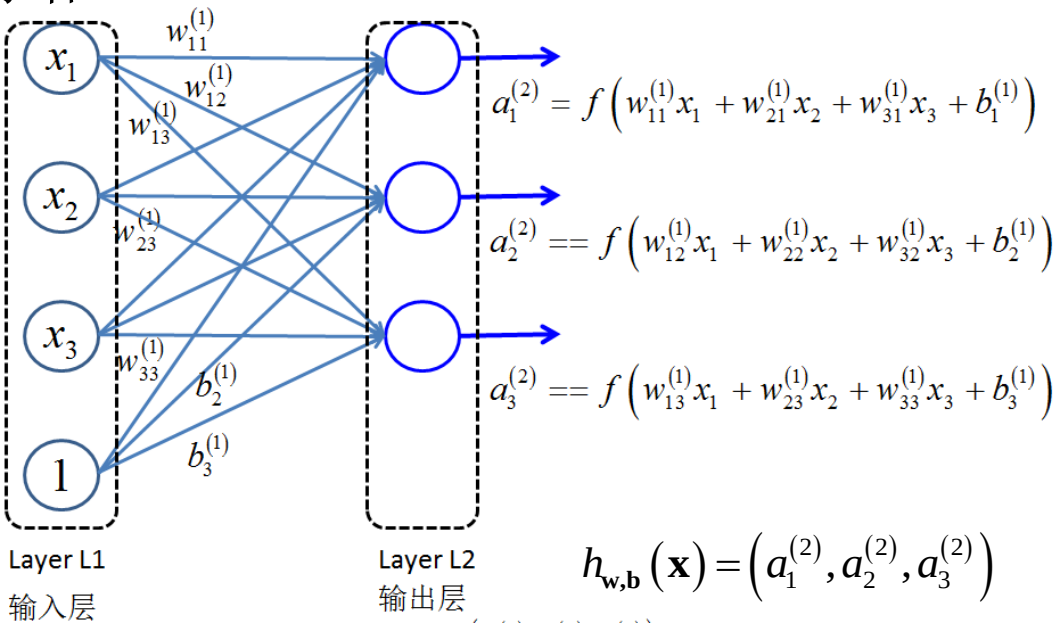
神经网络

- **多层感知机**
  - 通常将没有环路或回路的人工神经网络称为前馈网络模型。感知机是一种最简单的前馈网络模型
  - 在感知机模型的基础之上添加隐含层，通常将此类模型称为多层感知机模型（MLP）



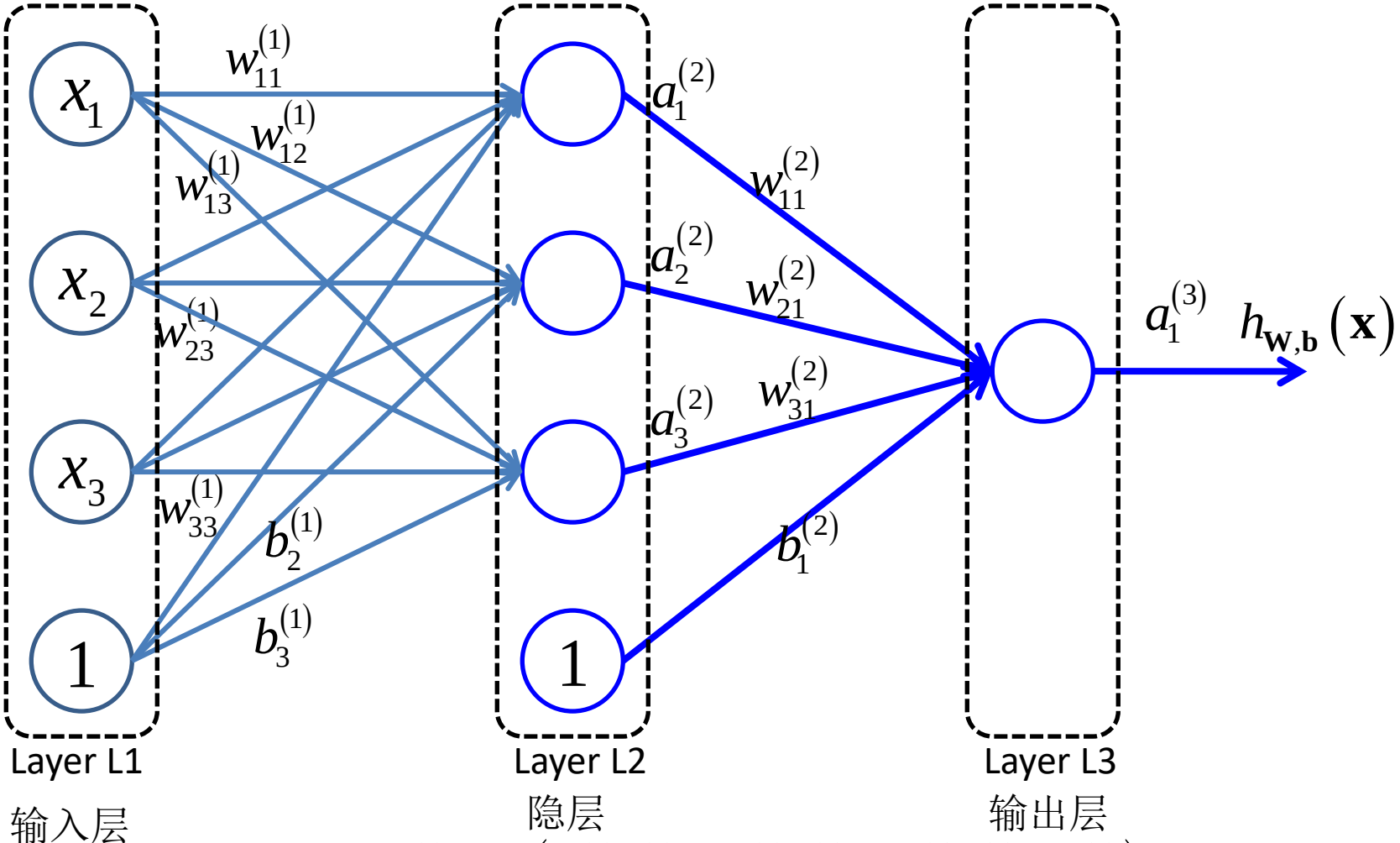
神经网络

- **单隐层感知机**
  - MLP模型中信息处理神经元的激活函数通常为Sigmoid函数。故MLP模型的隐含层可将数据通过非线性映射表示在另一个空间当中，并将模型输出限制在区间(0,1)当中
  - MLP模型中隐含层的层数为一层的神经网络称为单隐层神经网络



神经网络

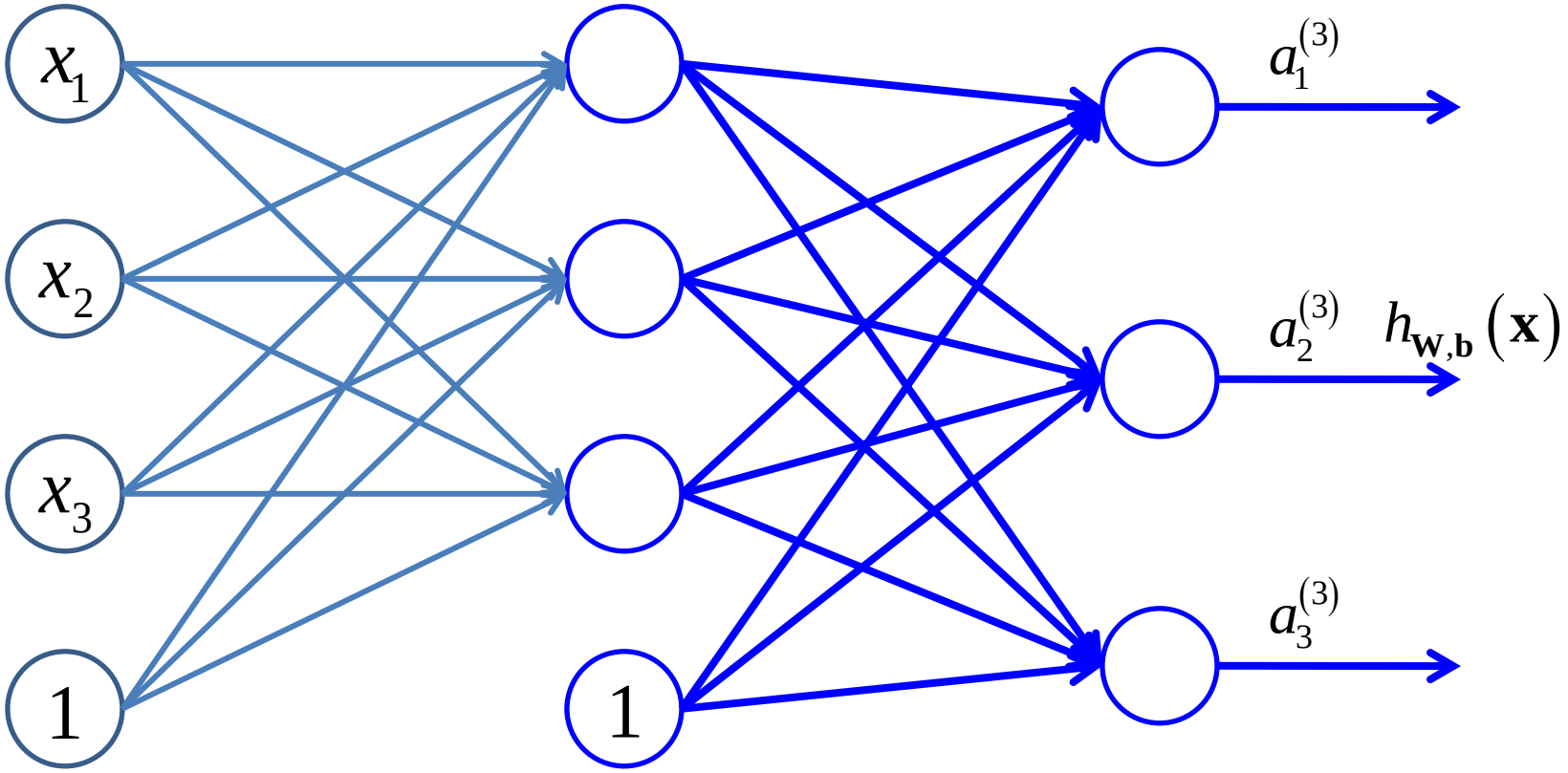
- 单隐层感知机



$$h_{\mathbf{w},\mathbf{b}}(\mathbf{x}) = a_1^{(3)} = f\left(w_{11}^{(2)}a_1^{(2)} + w_{21}^{(2)}a_2^{(2)} + w_{31}^{(2)}a_3^{(2)} + b_1^{(2)}\right)$$

神经网络

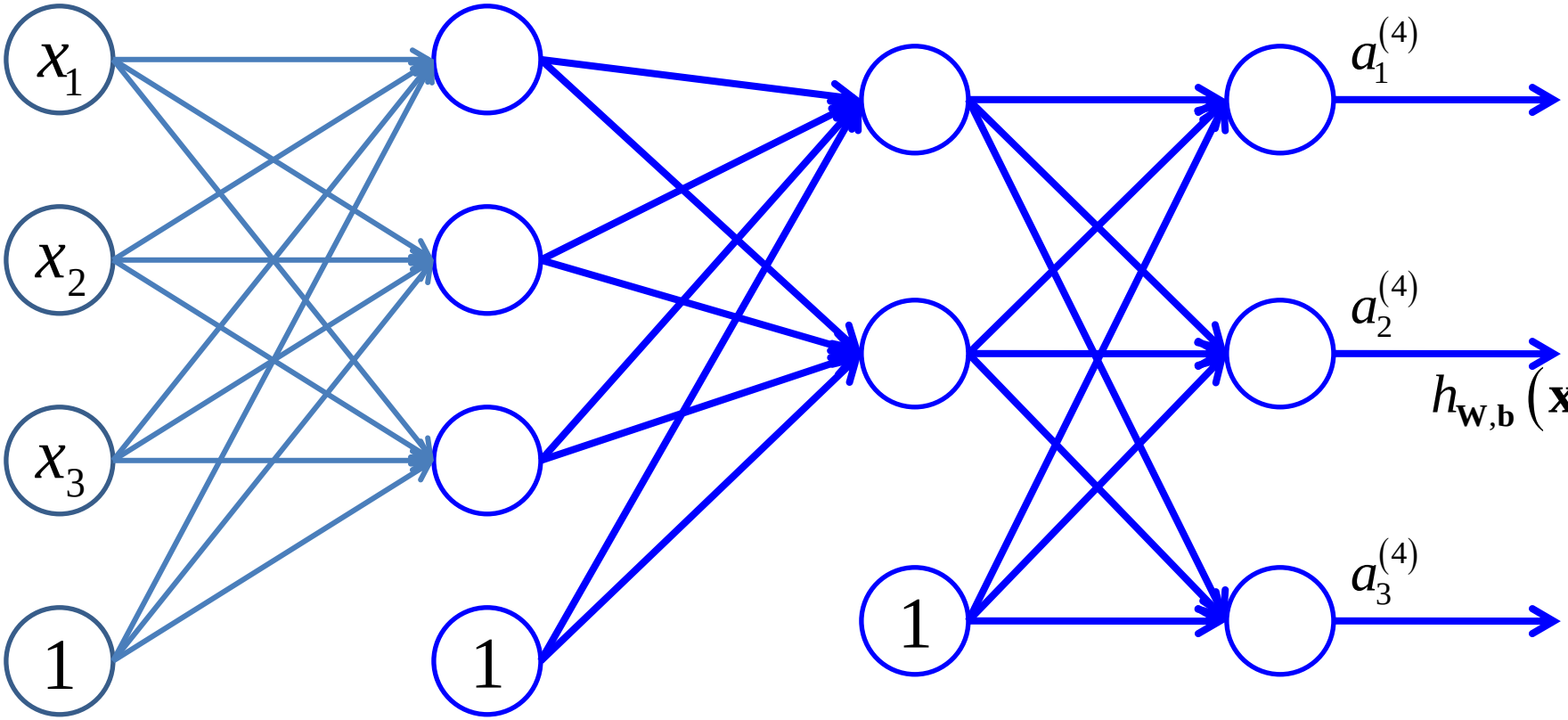
- 单隐层感知机



$$\mathbf{h}_{\mathbf{w},\mathbf{b}}(\mathbf{x}) = \begin{pmatrix} a_1^{(3)} & a_2^{(3)} & a_3^{(3)} \end{pmatrix}^T$$

神经网络

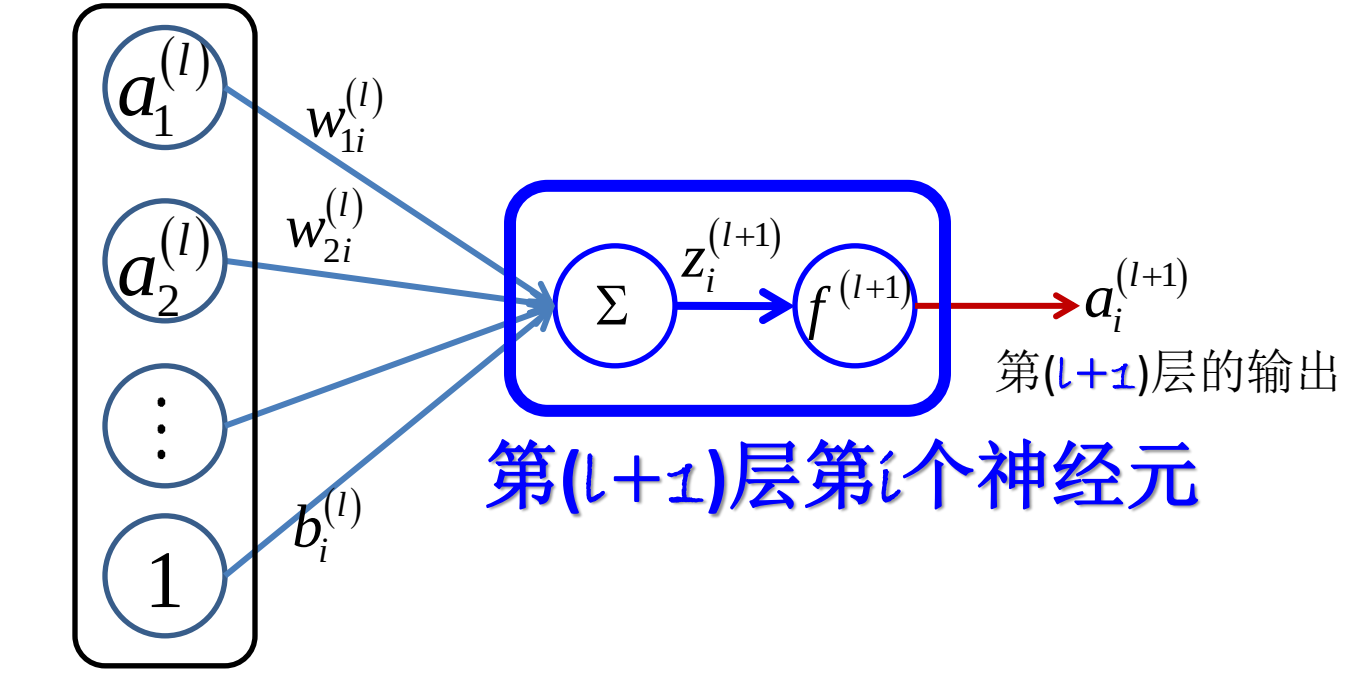
- 多层感知机



$$\mathbf{h}_{\mathbf{w,b}}(\mathbf{x}) = \begin{pmatrix} a_1^{(4)} & a_2^{(4)} & a_3^{(4)} \end{pmatrix}^T$$

神经网络

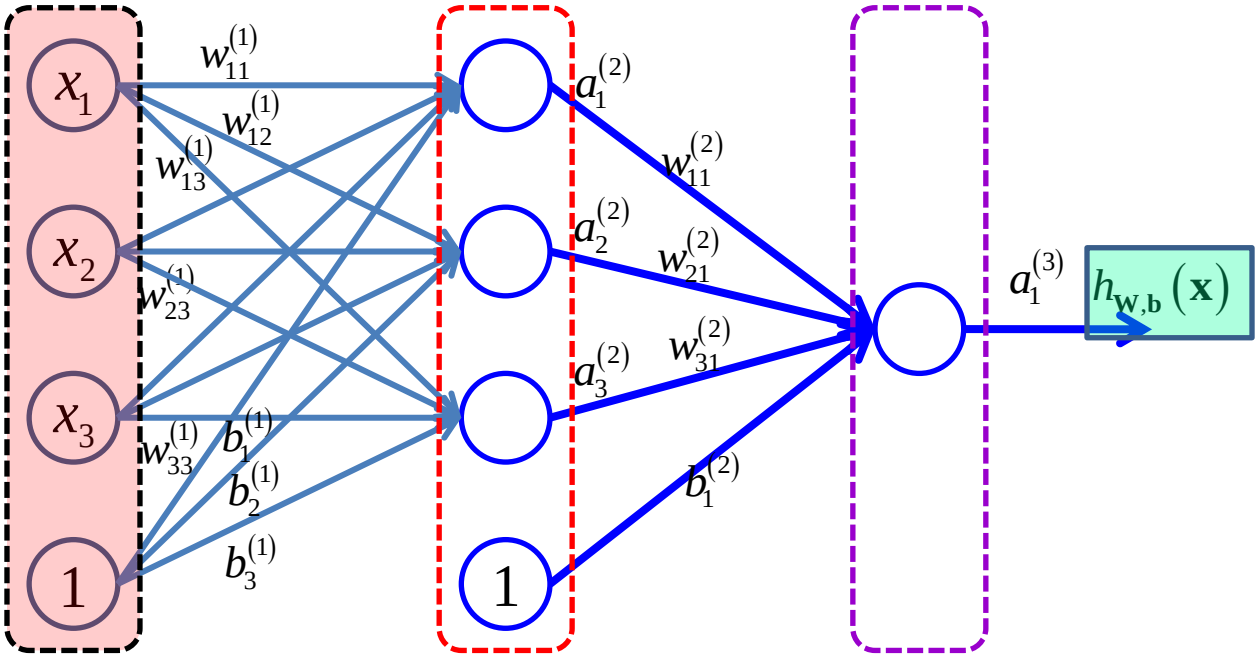
- 多层感知机



$$a_i^{(1)} = x_i; a_i^{(l+1)} = f^{(l+1)}\left(\sum_{j=1}^{N_l} w_{ji}^{(l)} a_j^{(l)} + b_i^{(l)}\right)$$

神经网络

- 多层感知机



$$L=3, N_3=1: a_i^{(1)}=x_i; a_i^{(2)}=f\left(\sum_{j=1}^{N_1} w_{ji}^{(1)} a_j^{(1)}+b_i^{(1)}\right);$$

$$h(x)=a_1^{(3)}=f\left(\sum_{j=1}^{N_2} w_{j1}^{(2)} a_j^{(2)}+b_1^{(2)}\right)=f\left(\sum_{j=1}^{N_2}\left[w_{ji}^{(2)} f\left(\sum_{m=1}^{N_1} w_{mj}^{(1)} x_m+b_j^{(1)}\right)\right]+b_1^{(2)}\right)$$

神经网络

- 多层感知机

$$L=4, N_4=1:$$

$$h(x)=a_1^{(4)}=f\left(\sum_{j=1}^{N_3}\left[w_{j1}^{(3)} f\left(\sum_{m=1}^{N_2}\left[w_{mj}^{(2)} f\left(\sum_{n=1}^{N_1} w_{nm}^{(1)} x_n + b_m^{(1)}\right)\right] + b_j^{(2)}\right)\right] + b_1^{(3)}\right)$$

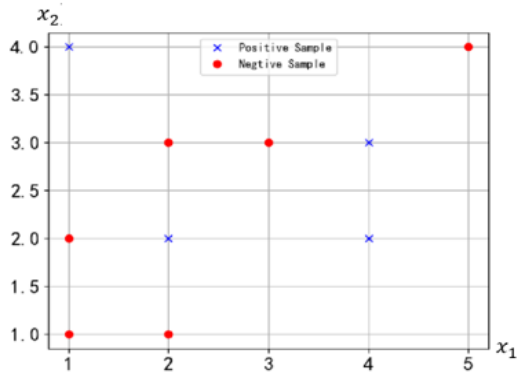
$$h(x)=f^{(k+5)}\left(\cdots f^{(k+2)}\left(\cdots f^{(k)}\left(\cdots x\cdots\right)\cdots\right)\cdots\right)$$

通过不同函数的复合(composition)构造出一个复杂函数

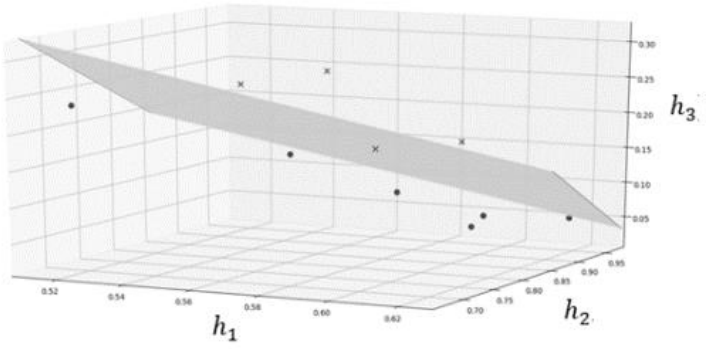
对输入特征x做多层次变换，得到输出h(x)

神经网络

- **多层感知机**
  - 通过多层的函数复合可以把原线性不可分的数据映射后线性可分。以输入向量为$X=(1,x_1,x_2)^T$且隐含层输出向量为$\boldsymbol{h}'=(h_1,h_2,h_3)^T$的MLP模型为例，使用训练数据集$D$构造MLP模型，$D$中原始数据分布情况如左图所示，使用该数据集训练好的MLP模型可通过隐含层将原始数据分布映射为如右图所示的分布



原始数据分布



映射后数据分布

本节目录

反向传播算法

- **经验风险最小化**
  - 损失函数
    - 回归
    - 二分类
    - 多分类
  - 经验风险
  - 梯度下降法

$$l\left(y,h\left(x;\mathbf{W},\mathbf{b}\right)\right)=\frac{1}{2}\left\|h\left(x\right)-y\right\|^2$$

$$l\left(y,h\left(x;\mathbf{W},\mathbf{b}\right)\right)=-y\ln h\left(x\right)-\left(1-y\right)\ln\left(1-h\left(x\right)\right)$$

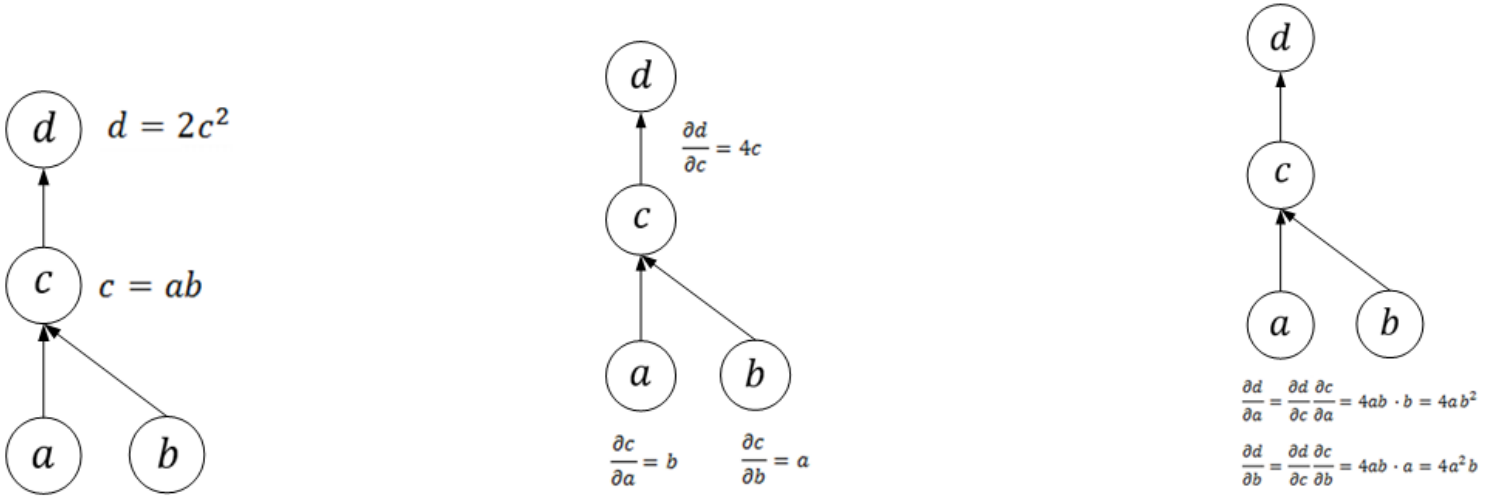
$$l\left(y,h\left(x;\mathbf{W},\mathbf{b}\right)\right)=-\sum_{c=1}^{K}\delta\left(y=c\right)\ln h_k\left(x\right)$$

$$J\left(\mathbf{W},\mathbf{b}\right)=\frac{1}{m}\sum_{i=1}^{m}l\left(y^{(i)},h\left(x^{(i)};\mathbf{W},\mathbf{b}\right)\right)$$

$$\mathbf{W}:=\mathbf{W}-\alpha\frac{\partial J\left(\mathbf{W},\mathbf{b}\right)}{\partial\mathbf{W}};\mathbf{b}:=\mathbf{b}-\alpha\frac{\partial J\left(\mathbf{W},\mathbf{b}\right)}{\partial\mathbf{b}}$$

反向传播算法

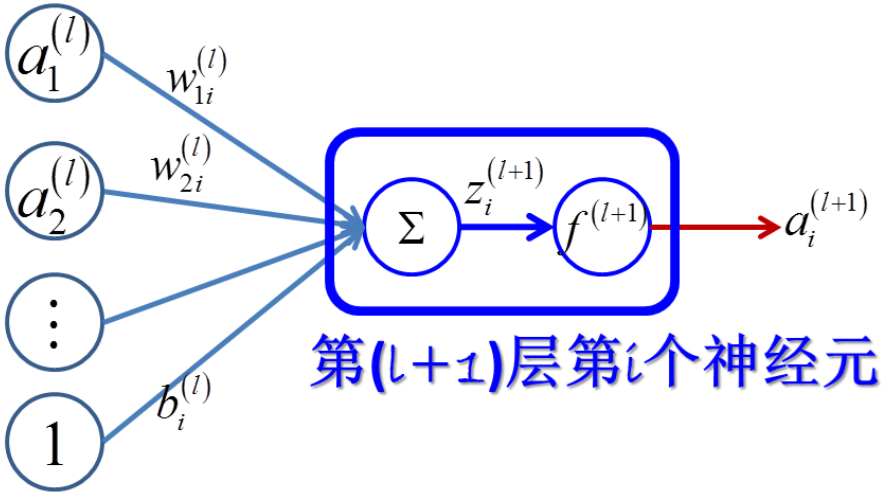
- **梯度的逐层求解**
  - 对于神经网络这样包含多个数据处理层的前馈网络模型而言，考虑使用反向传播算法对误差和参数所对应的梯度进行逐层求解



梯度的逐层求解过程

反向传播算法

- 反向传播推导



$$a_i^{(l+1)} = f^{(l+1)}\left(\sum_{j=1}^{N_l} w_{ji}^{(l)} a_j^{(l)} + b_i^{(l)}\right) = f^{(l+1)}\left(z_i^{(l+1)}\right)$$

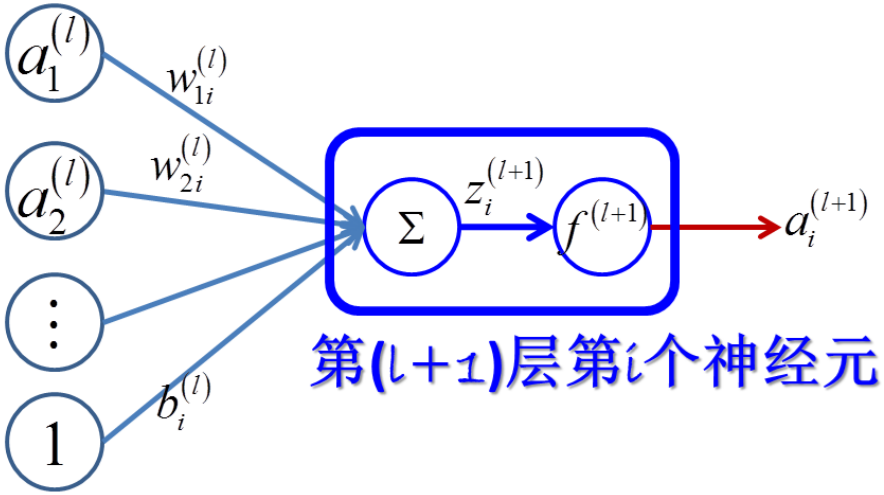
$$z_i^{(l+1)} = w_i^{(l)T} a^{(l)} + b_i^{(l)}$$

$$w_i^{(l)} = \begin{pmatrix} w_{1i}^{(l)} & w_{2i}^{(l)} & \cdots & w_{N_l i}^{(l)} \end{pmatrix}^T$$

$$a^{(l)} = \begin{pmatrix} a_1^{(l)} & a_2^{(l)} & \cdots & a_{N_l}^{(l)} \end{pmatrix}^T$$

反向传播算法

- 反向传播推导



误差

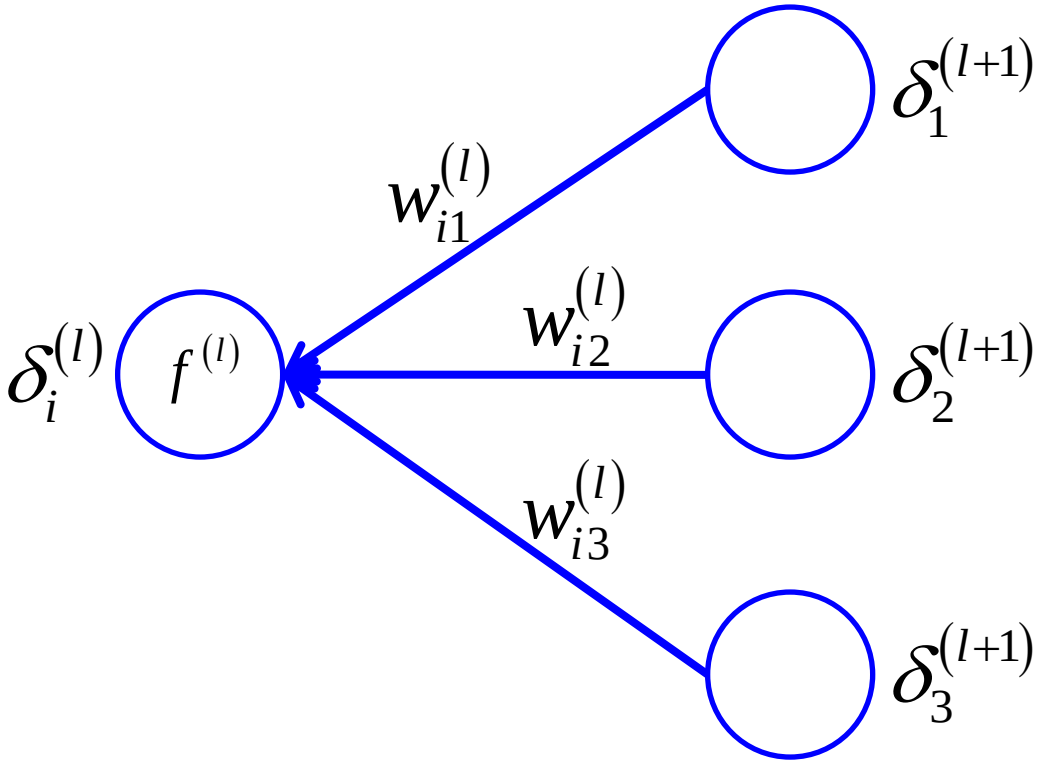
$$\delta_i^{(l)} \equiv \frac{\partial J}{\partial z_i^{(l)}} = \frac{\partial J}{\partial a_i^{(l)}} \cdot \frac{\partial a_i^{(l)}}{\partial z_i^{(l)}} = \frac{\partial J}{\partial a_i^{(l)}} \cdot f'\left(z_i^{(l)}\right), \quad \frac{\partial J}{\partial a_i^{(l)}} = \sum_{j=1}^{N_{l+1}} \frac{\partial J}{\partial z_j^{(l+1)}} \cdot \frac{\partial z_j^{(l+1)}}{\partial a_i^{(l)}}$$

$$\delta_i^{(l)} = f'\left(z_i^{(l)}\right) \cdot \left[\sum_{j=1}^{N_{l+1}} w_{ij}^{(l)} \delta_j^{(l+1)}\right]$$

$$\frac{\partial J}{\partial b_i^{(l)}} = \delta_i^{(l+1)} \qquad \frac{\partial J}{\partial w_{ji}^{(l)}} = \delta_i^{(l+1)} a_j^{(l)}$$

反向传播算法

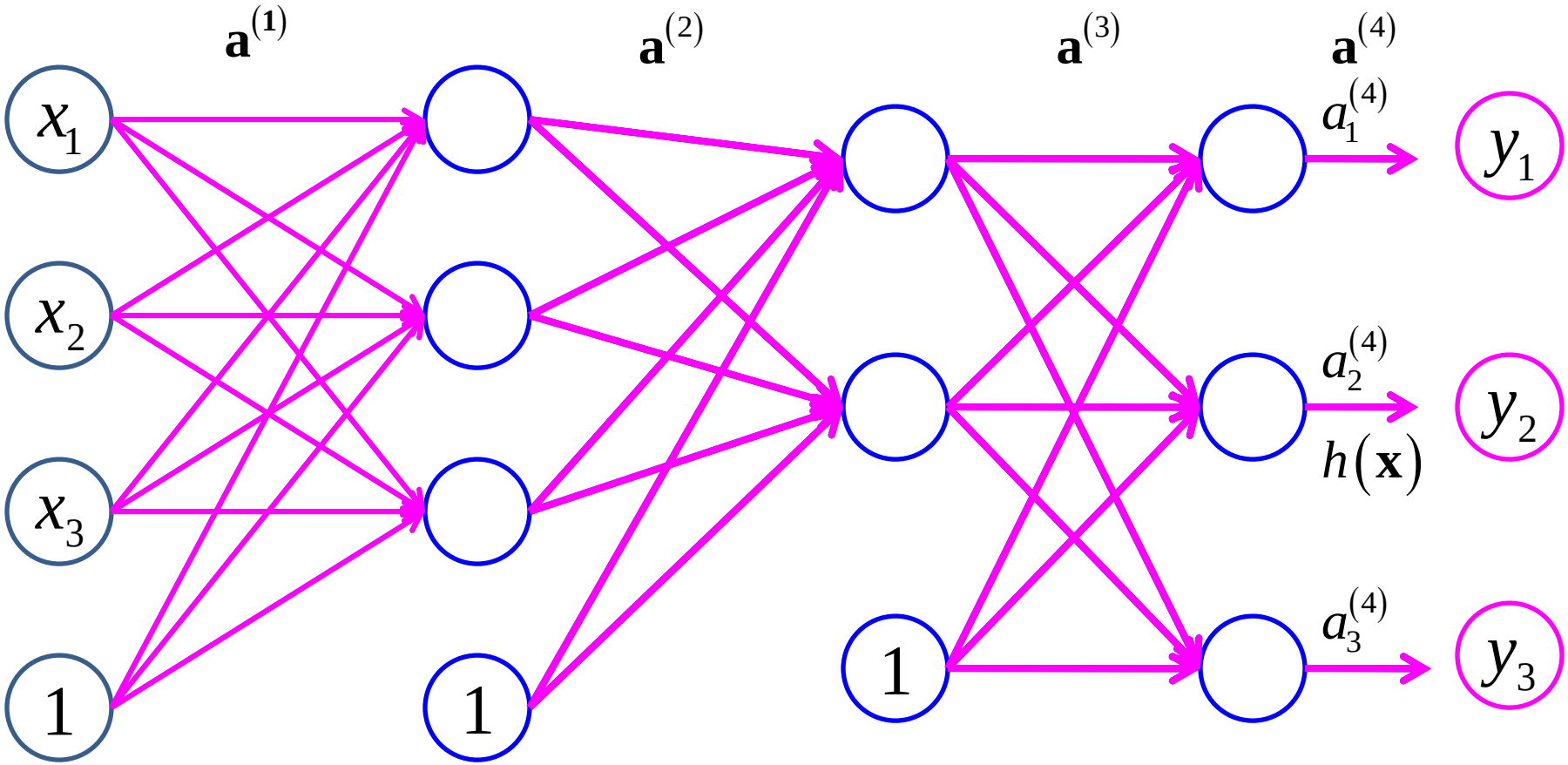
- 反向传播推导



$$\delta_i^{(l)} = f'\left(z_i^{(l)}\right) \cdot \left[\sum_{j=1}^{N_{l+1}} w_{ij}^{(l)} \delta_j^{(l+1)}\right]$$

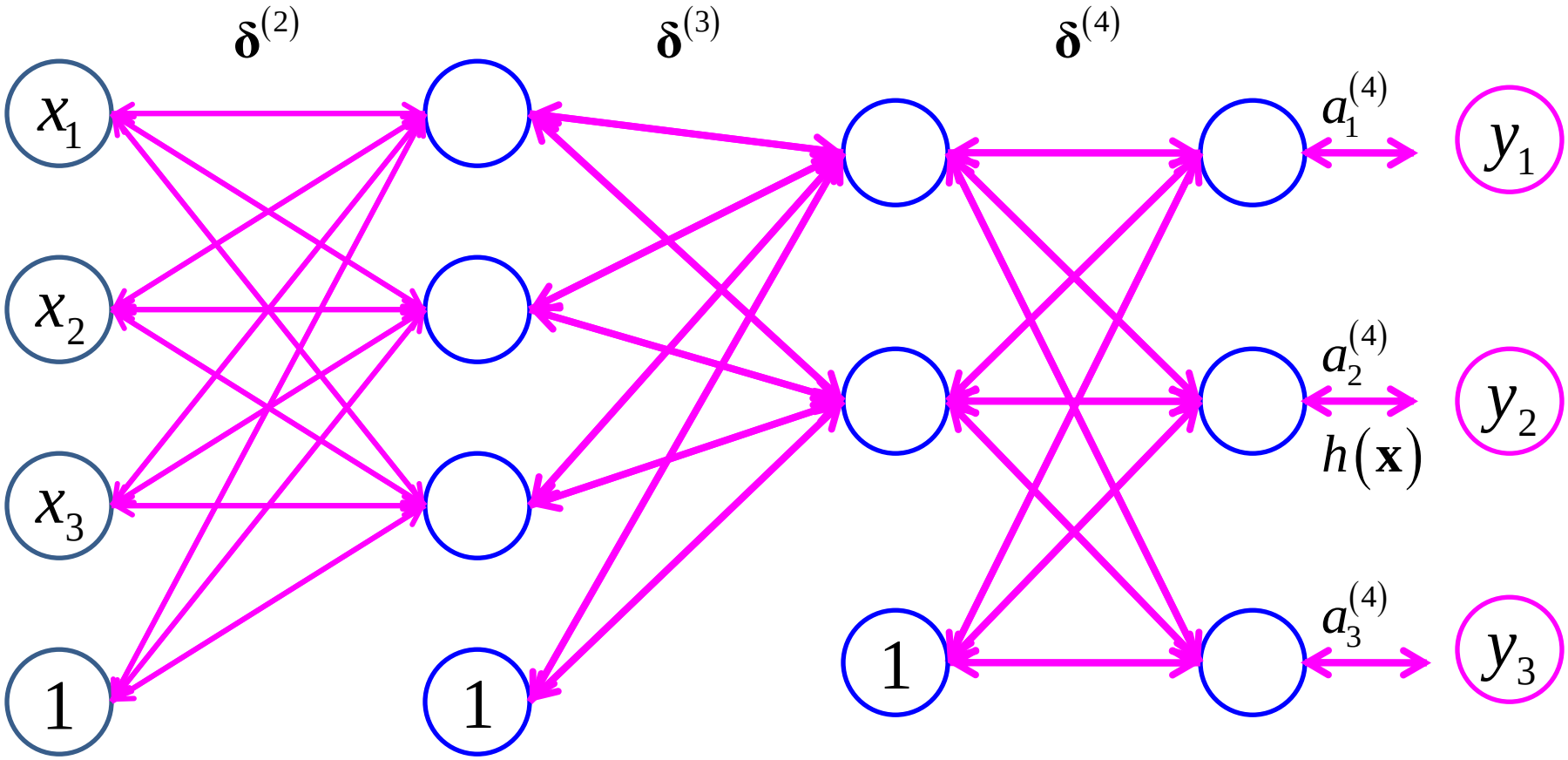
反向传播算法

- 前向传播过程



反向传播算法

- 误差后向传播过程



$$\mathbf{W} := \mathbf{W} - \alpha \frac{\partial J(\mathbf{W},\mathbf{b})}{\partial \mathbf{W}} \qquad \mathbf{b} := \mathbf{b} - \alpha \frac{\partial J(\mathbf{W},\mathbf{b})}{\partial \mathbf{b}}$$

$$J(W,b;x,y)$$

反向传播算法

- **算法流程**
  - 第一步：初始化W,b
  - 第二步：
  - Do{ 1.前向传播：

$$for\ l=1:L-1\quad z_j^{(l+1)}\left(x^{(n)}\right)=\sum_{i=1}^{N_l} w_{ij}^{(l)} a_i^{(l)}\left(x^{(n)}\right)+b_j^{(l)};\quad a_j^{(l+1)}\left(x^{(n)}\right)=f^{(l+1)}\left(z_j^{(l+1)}\left(x^{(n)}\right)\right)$$

2.误差后向传播：

$$l=L:\quad \delta_j^{(L)}\left(x^{(n)}\right)=\frac{\partial J\left(W,b;x^{(n)},y^{(n)}\right)}{\partial a_j^{(L)}} f^{(L)\prime}\left(z_i^{(L)}\left(x^{(n)}\right)\right)$$

$$for\ l=L-1:2\quad \left\{\begin{array}{c} \delta_i^{(l)}\left(x^{(n)}\right)=f^{(l)\prime}\left(z_i^{(L)}\left(x^{(n)}\right)\right)\left[\sum_{j=1}^{N_{l+1}} w_{ij}^{(l)}\delta_j^{(l+1)}\left(x^{(n)}\right)\right] \\ \frac{\partial J\left(W,b;x^{(n)},y^{(n)}\right)}{\partial b_j^{(l)}}=\delta_j^{(l+1)}\left(x^{(n)}\right);\frac{\partial J\left(W,b;x^{(n)},y^{(n)}\right)}{\partial w_{ij}^{(l)}}=\delta_j^{(l+1)}a_i^{(l)}\left(x^{(n)}\right) \end{array}\right\}$$

$$l=1:\quad \frac{\partial J\left(W,b;x^{(n)},y^{(n)}\right)}{\partial b_j^{(1)}}=\delta_j^{(2)}\left(x^{(n)}\right);\quad \frac{\partial J\left(W,b;x^{(n)},y^{(n)}\right)}{\partial w_{ij}^{(1)}}=\delta_j^{(2)}\left(x^{(n)}\right)x_i^{(n)}$$

3.更新权值：

$$w_{ij}^{(l)}:=w_{ij}^{(l)}-\alpha\frac{1}{m}\sum_{n=1}^{m}\frac{\partial J\left(W,b;x^{(n)},y^{(n)}\right)}{\partial w_{ij}^{(l)}};\quad b_j^{(l)}:=b_j^{(l)}-\alpha\frac{1}{m}\sum_{n=1}^{m}\frac{\partial J\left(W,b;x^{(n)},y^{(n)}\right)}{\partial b_j^{(l)}}$$

  - }Until convergence

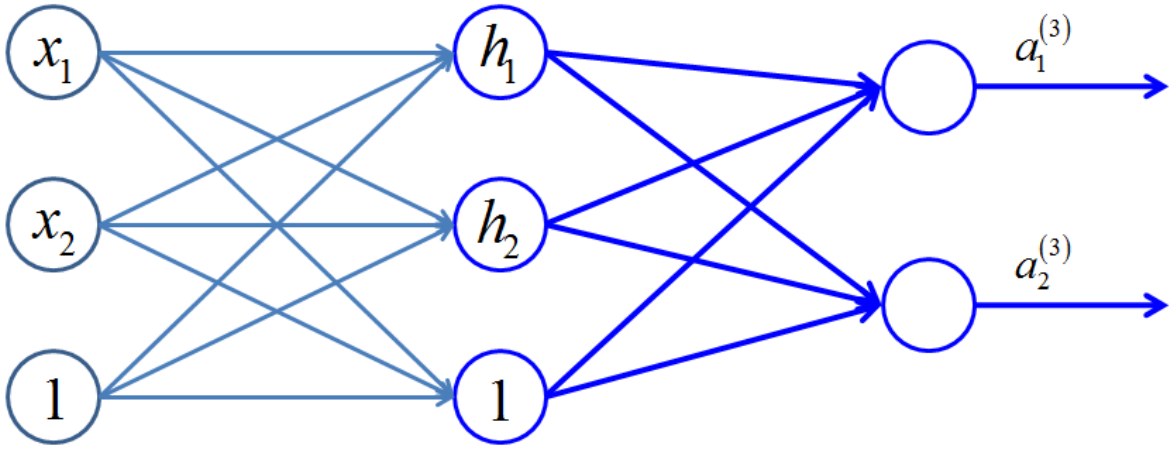
反向传播算法

例题1：现有如图所示的神经网络模型，其中各神经元均使用Sigmoid激活函数，假设当前网络的权重向量$\boldsymbol{W}$取值为：

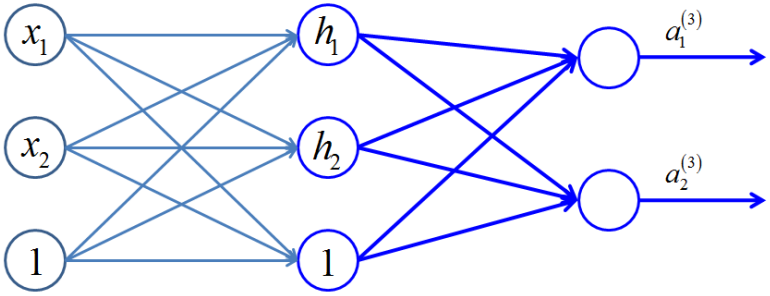
$$\boldsymbol{W}=\left(w_{11}^{(1)},w_{12}^{(1)},w_{21}^{(1)},w_{22}^{(1)},b_1^{(2)},b_2^{(2)},w_{11}^{(2)},w_{12}^{(2)},w_{21}^{(2)},w_{22}^{(2)},b_1^{(3)},b_2^{(3)}\right)^T$$

$$=(0.2,03,0.4,0.5,0.3,0.3,0.7,0.5,0.8,0.3,0.5,0.5)^T$$

试用样本$(X,\boldsymbol{y})=[(0.4,0.6),(0.6,0.6)]$完成对连接权重的一次更新



反向传播算法



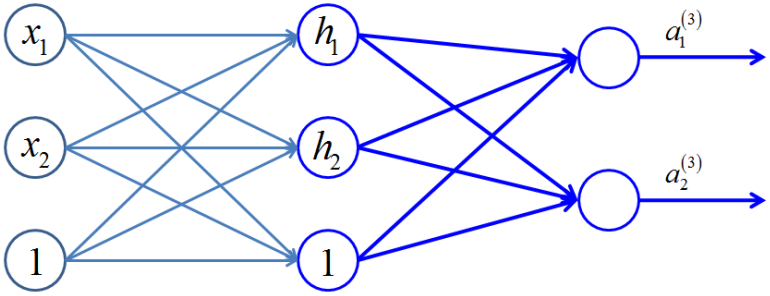
- 假设用于模型优化的目标函数为：

$$J(\boldsymbol{W}) = \sum_{j=1}^{2} J_j(\boldsymbol{W}) = \frac{1}{2}\sum_{j=1}^{2}\left(y_j - f_j(X)\right)^2$$

- 将数据输入该网络可求得隐含层的输出为$h_1 = 0.6502$，$h_2 = 0.6726$，网络的最终输出为$y_1 = 0.8166$，$y_2 = 0.7363$
- 对于隐含层到输出层的连接权重$w_{11}^{(2)}, w_{12}^{(2)}, w_{21}^{(2)}, w_{22}^{(2)}$，对应的梯度计算公式如下：

$$\frac{\partial J}{\partial \mathrm{w}_{uv}^{(l)}} = -h_u \cdot \sigma'\left(\boldsymbol{w}_v^{(2)^T}\boldsymbol{h}\right) \cdot \left(y_v - f_v(X)\right)$$

反向传播算法



- 以$w_{11}^{(2)}$为例，有：

$$\frac{\partial \mathrm{J}}{\mathrm{w}_{11}^{(2)}} = -h_1 \cdot \sigma'\left(\mathrm{w}_1^{(2)^{\mathrm{T}}}\boldsymbol{h}\right) \cdot (\mathrm{y}_1 \mathrm{f}_1(\mathrm{X}))$$

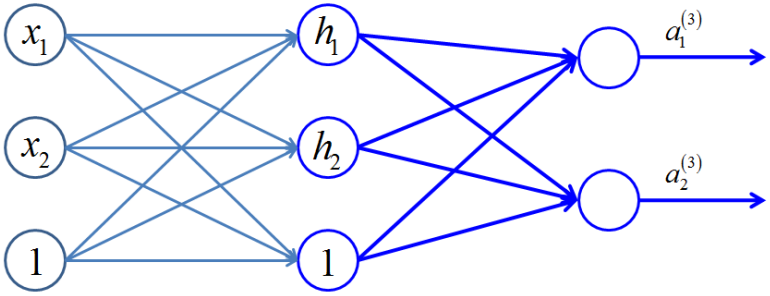
$$= -0.6502 \times \sigma(0.6502 \times 0.7 + 0.6726 \times 0.8 + 0.5) \times$$

$$(1 - \sigma(0.6502 \times 0.7 + 0.6726 \times 0.8 + 0.5))0.6 - 0.(8166) \approx 0.0211$$

- 取参数更新步长为0.1，故将$w_{11}^{(2)}$做如下更新为：

$$\mathrm{w}_{11}^{(2)}\mathrm{w}_{11}^{(2)} - \frac{\partial \mathrm{J}}{\mathrm{w}_{11}^{(\mathrm{l})}} = 0.7 - 0.1 \times 0.0211 = 0.6979$$

反向传播算法



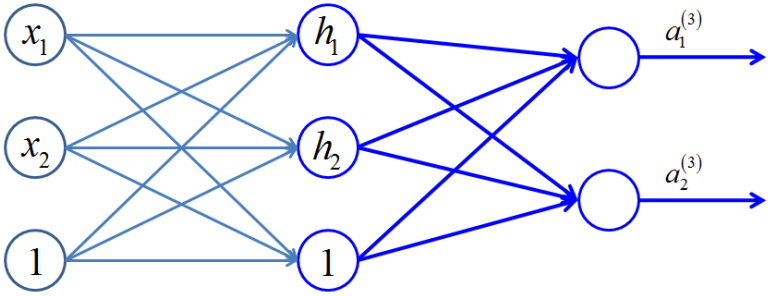
- 同理可求得更新一次之后$w_{12}^{(2)} = 0.7979$，$w_{21}^{(2)} = 0.4982$，$w_{22}^{(2)} = 0.2982$
- 对于输入层到隐含层的连接权重而言，其梯度计算公式如下：

$$\frac{\partial J}{\partial \mathrm{w}_{uv}^{(l)}} = \frac{\partial J}{\partial h_{\mathrm{v}}} \cdot \sigma\left(\mathrm{w}_{\mathrm{v}}^{(\mathrm{l})^{\mathrm{T}}} X_i\right)\left(1 - \sigma\left(\mathrm{w}_{\mathrm{v}}^{(\mathrm{l})^{\mathrm{T}}} X_i\right)\right) \cdot \mathrm{x}_{\mathrm{iu}}$$

其中$\partial J / \partial h_v$为：

$$-\sum_{j=1}^{k}\left[h_v \cdot \sigma'\left(\mathrm{w}_{\mathrm{j}}^{(\mathrm{l})^{\mathrm{T}}} \boldsymbol{h}\right) \cdot \sum_{i=1}^{n}\left(y_{ij} - f_j(X_i)\right)\right]$$

反向传播算法



- 以$w_{11}^{(1)}$为例，有：

$$\frac{\partial J}{\partial \mathrm{w}_{11}^{(1)}} = -\sum_{j=1}^{2}\left[h_1 \cdot \sigma'\left(\mathrm{w}_{\mathrm{j}}^{(1)^{\mathrm{T}}}\boldsymbol{h}\right)\cdot\sum_{i=1}^{n}\left(y_{ij}-f_j(X_i)\right)\right]\sigma\left(\mathrm{w}_1^{(1)^{\mathrm{T}}}X\right)\left(1-\sigma\left(\mathrm{w}_1^{(1)^{\mathrm{T}}}X\right)\right)\cdot \mathrm{x}_1$$

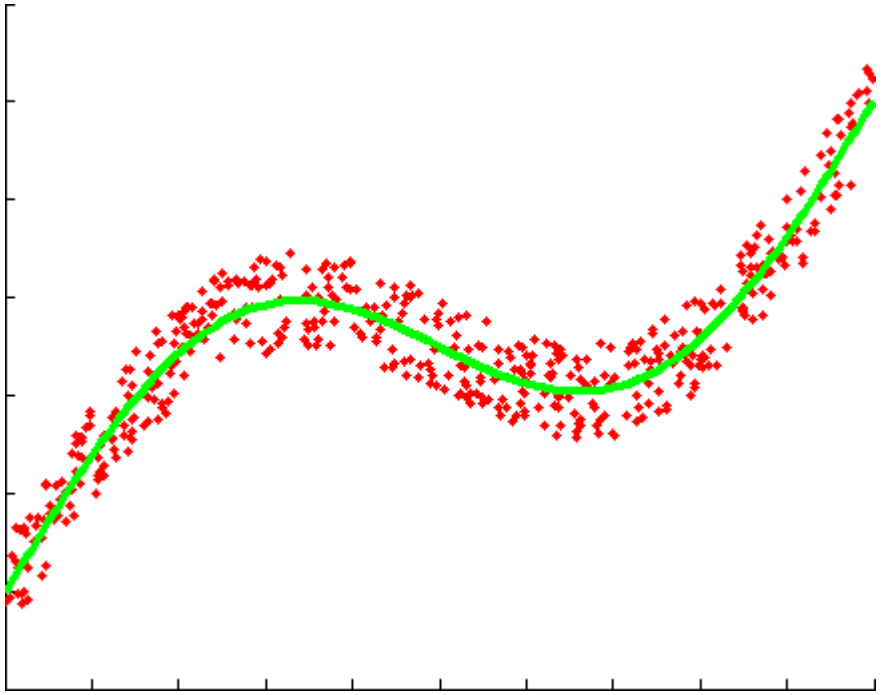
- 代入数据可求得$\partial J/\partial \mathrm{w}_{11}^{(1)} \approx -0.053$，由梯度下降法公式有：

$$\mathrm{w}_{11}^{(1)} = \mathrm{w}_{11}^{(1)} - \alpha\frac{\partial J}{\partial \mathrm{w}_{11}^{(1)}} = 0.2053$$

- 同理可得其它参数一次更新后取值$\mathrm{w}_{12}^{(1)} = 0.4098$，$\mathrm{w}_{21}^{(1)} = 0.3044$，$\mathrm{w}_{22}^{(1)} = 0.5026$

反向传播算法

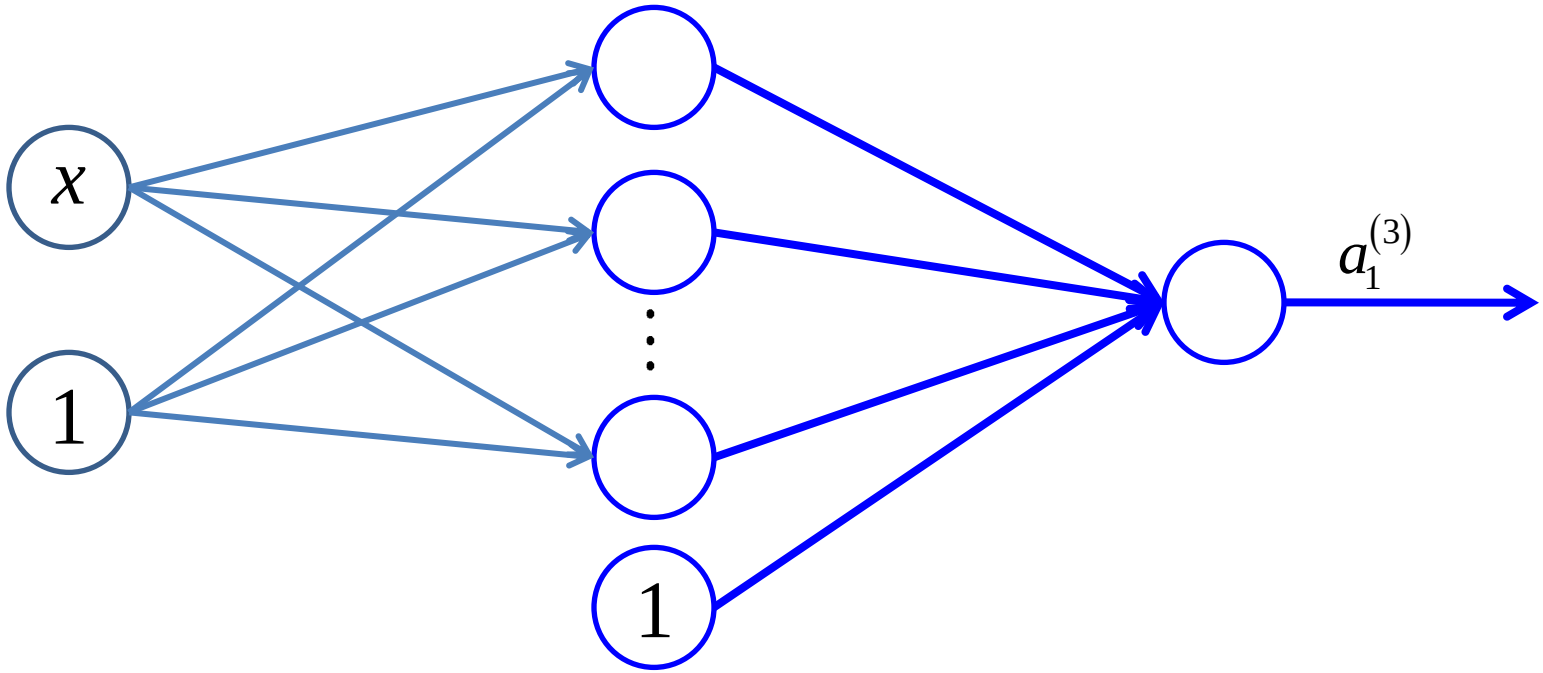
例题2：曲线拟合



$$y = x + 0.3\sin(2\pi x) + \varepsilon \quad x \in [0 \quad 1] \quad \varepsilon \in [-0.1 \quad 0.1]$$

反向传播算法

例题2： 曲线拟合



Layer L1
$N_1=1$

Layer L2：
$N_2=7$

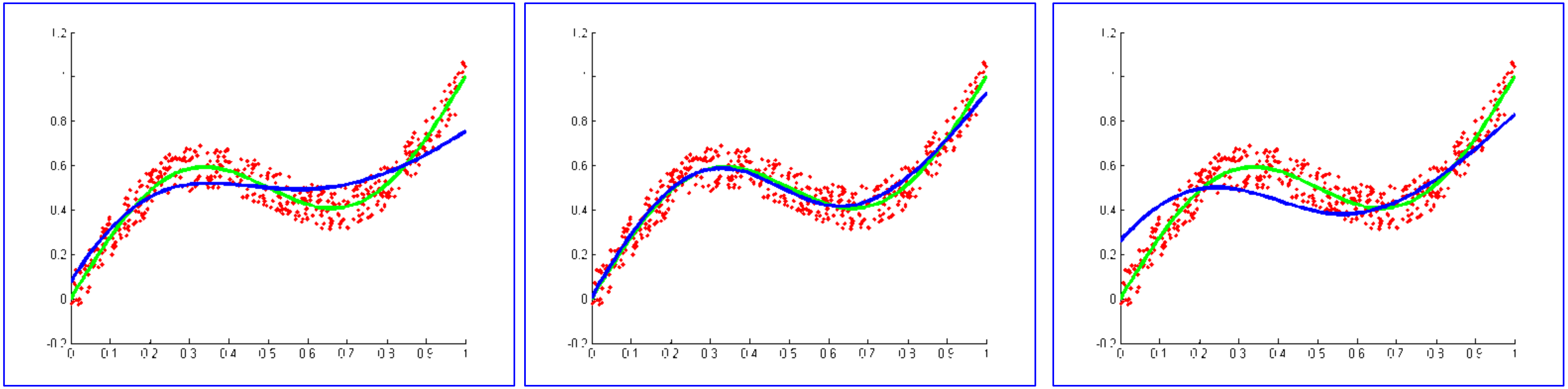
Layer L3
$N_3=1$

$$a_1^{(3)}(x) = z_1^{(3)}(x)$$

$$J(W,b) = -\frac{1}{2m}\sum_{i=1}^{m}\left\{\left(a_1^{(3)}\left(x^{(i)}\right) - y^{(i)}\right)^2\right\} + \frac{\lambda}{2m}\sum_{j=1}^{N_2}\left[\left(w_{1j}^{(1)}\right)^2 + \left(w_{j1}^{(2)}\right)^2\right]$$

反向传播算法

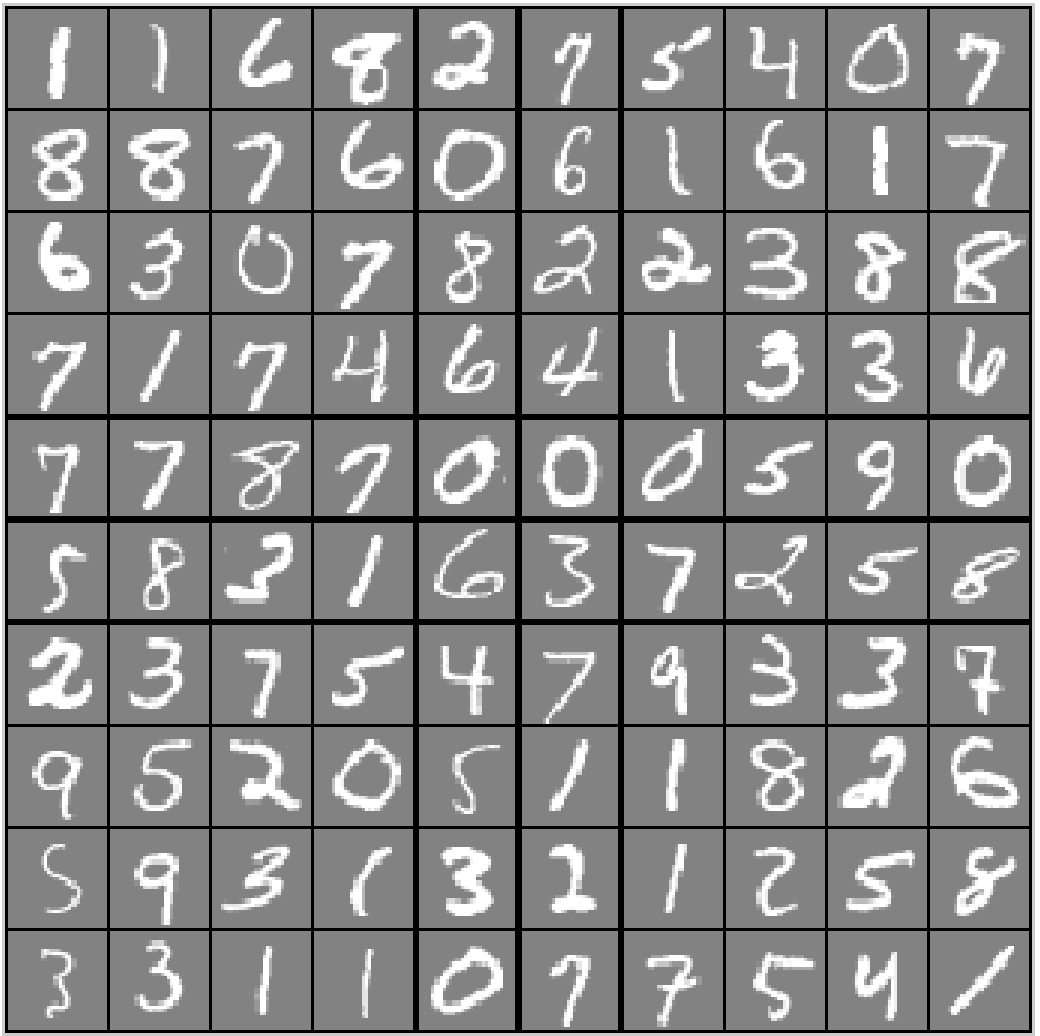
例题2： 曲线拟合



根据不同初始值优化的结果

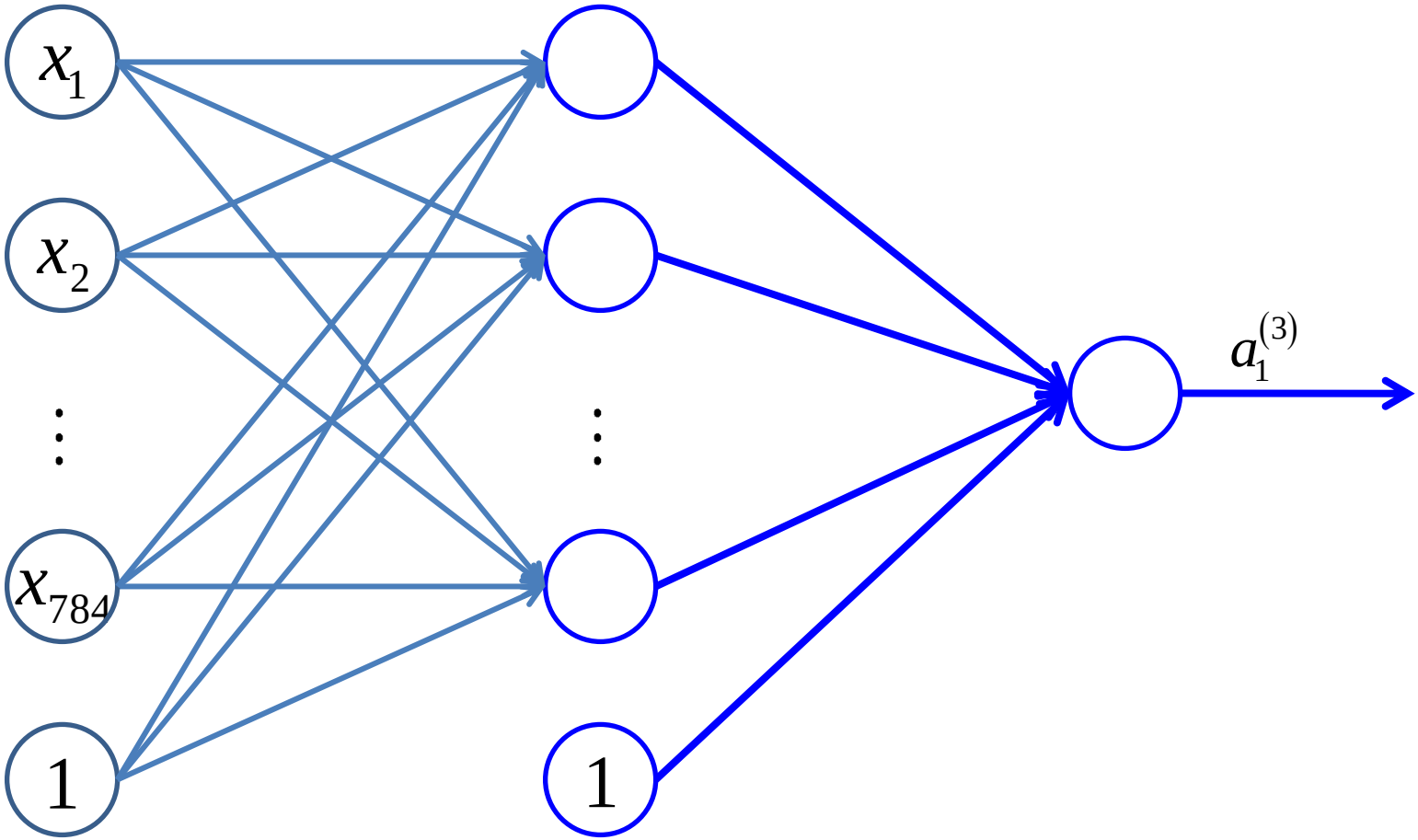
反向传播算法

例题3： 手写体数字识别-识别数字2



反向传播算法

例题3：手写体数字识别-识别数字2



Layer L1
N_1=784

Layer L2：
N_2=25

Layer L3
N_3=1

反向传播算法

例题3： 手写体数字识别-识别数字2

$$P(y=1|x)=a_1^{(3)}(x)=\frac{1}{1+\exp\left(-z_1^{(3)}(x)\right)}$$ Sigmoid函数

$$P(y=0|x)=1-a_1^{(3)}(x)$$

$$P(y|x)=\left[P(y=1|x)\right]^{y}\left[P(y=0|x)\right]^{1-y}$$

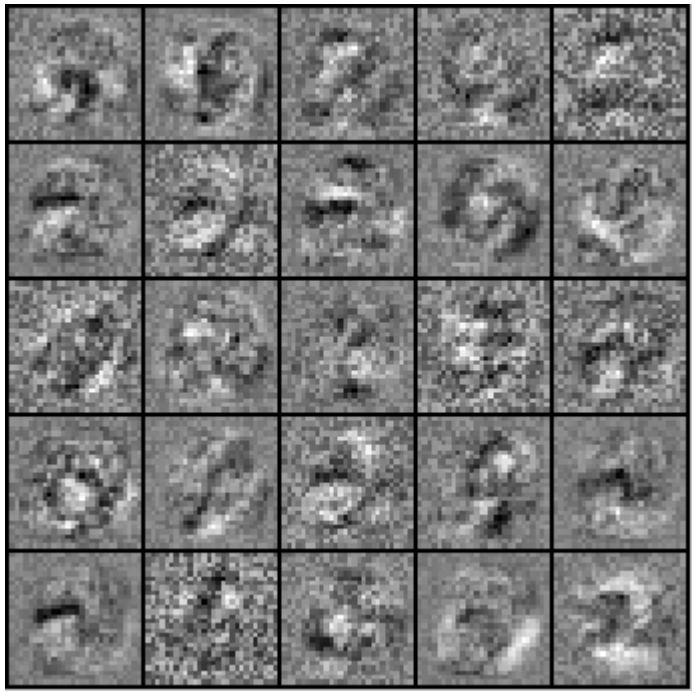
$$J(W,b;x,y)=-\log P(y|x)=-y\log\left[a_1^{(3)}(x)\right]-(1-y)\log\left[1-a_1^{(3)}(x)\right]$$

$$J(W,b)=-\frac{1}{m}\sum_{i=1}^{m}\left\{y^{(i)}\log\left[a_1^{(3)}\left(x^{(i)}\right)\right]+\left(1-y^{(i)}\right)\log\left[1-a_1^{(3)}\left(x^{(i)}\right)\right]\right\}$$

$$+\frac{\lambda}{2m}\left\{\sum_{i=1}^{784}\sum_{j=1}^{25}\left(w_{ij}^{(1)}\right)^2+\sum_{i=1}^{25}\left(w_{i1}^{(2)}\right)^2\right\}$$ 权值衰减(Weight Decay)

反向传播算法

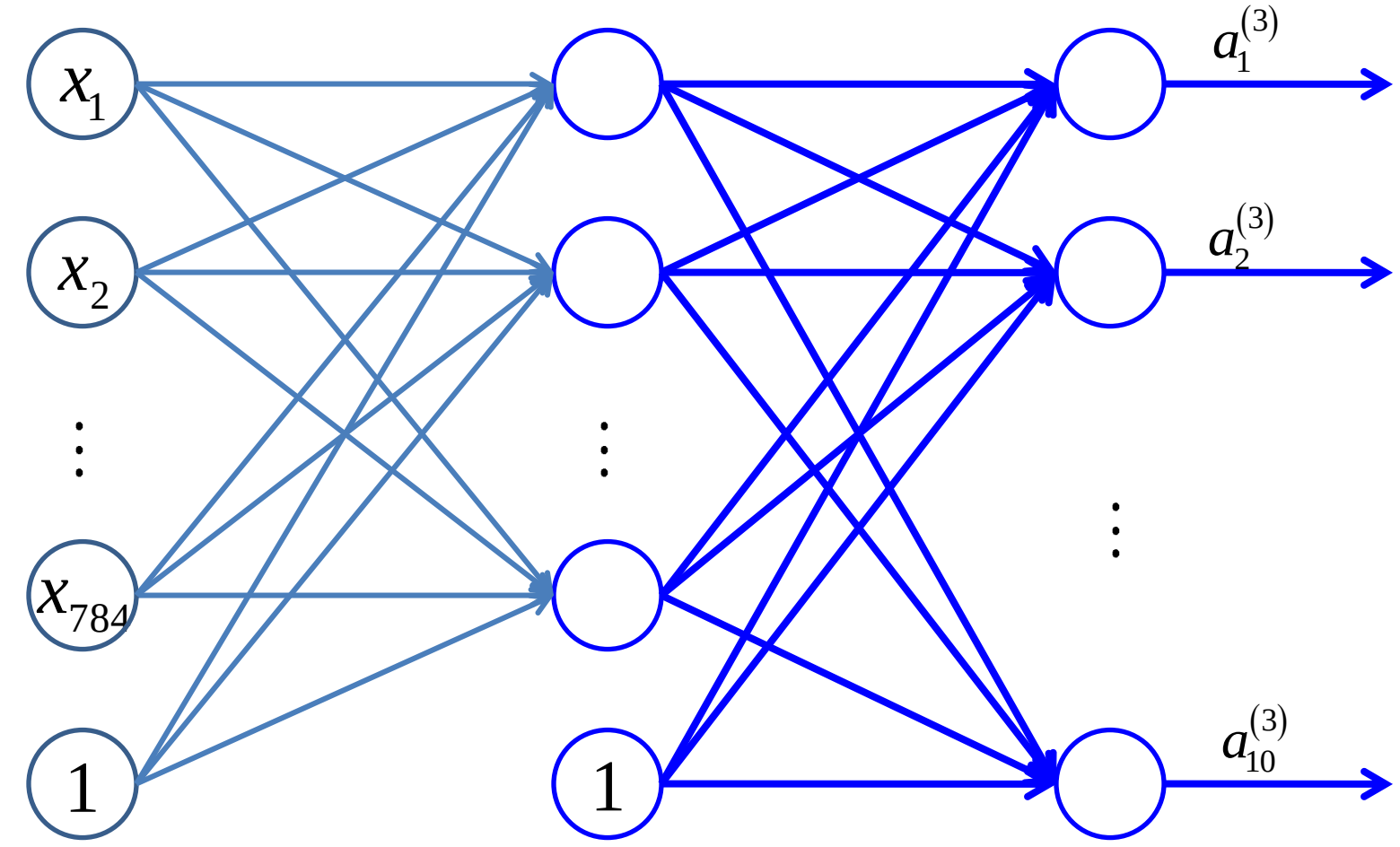
例题3：手写体数字识别-识别数字2



准确率=98.7%

反向传播算法

例题4：手写体数字识别-识别10个数字



反向传播算法

例题4：手写体数字识别-识别10个数字

$$P(y=k\mid x)=a_k^{(3)}(x)=\frac{\exp\left(z_k^{(3)}(x)\right)}{\sum_{i=1}^{10}\exp\left(z_i^{(3)}(x)\right)}$$

$$y=\begin{pmatrix}y_1\\y_2\\\vdots\\y_{10}\end{pmatrix};\ y_k=\begin{cases}1 & x\in k\\0 & x\notin k\end{cases}$$

$$P(y\mid x)=\prod_{j=1}^{10}\left[P(y=j\mid x)\right]^{y=j}$$

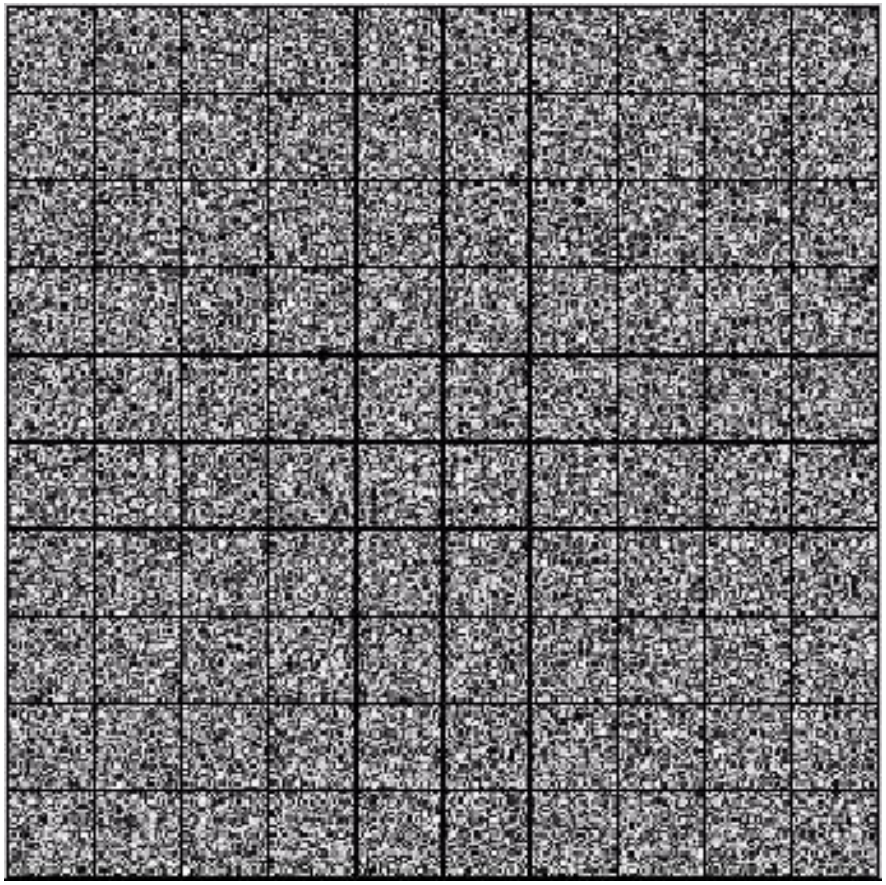
$$J(W,b;x,y)=-\log P(y\mid x)=-\sum_{j=1}^{10}\left\{y_j\log\left[a_j^{(3)}(x)\right]\right\}$$

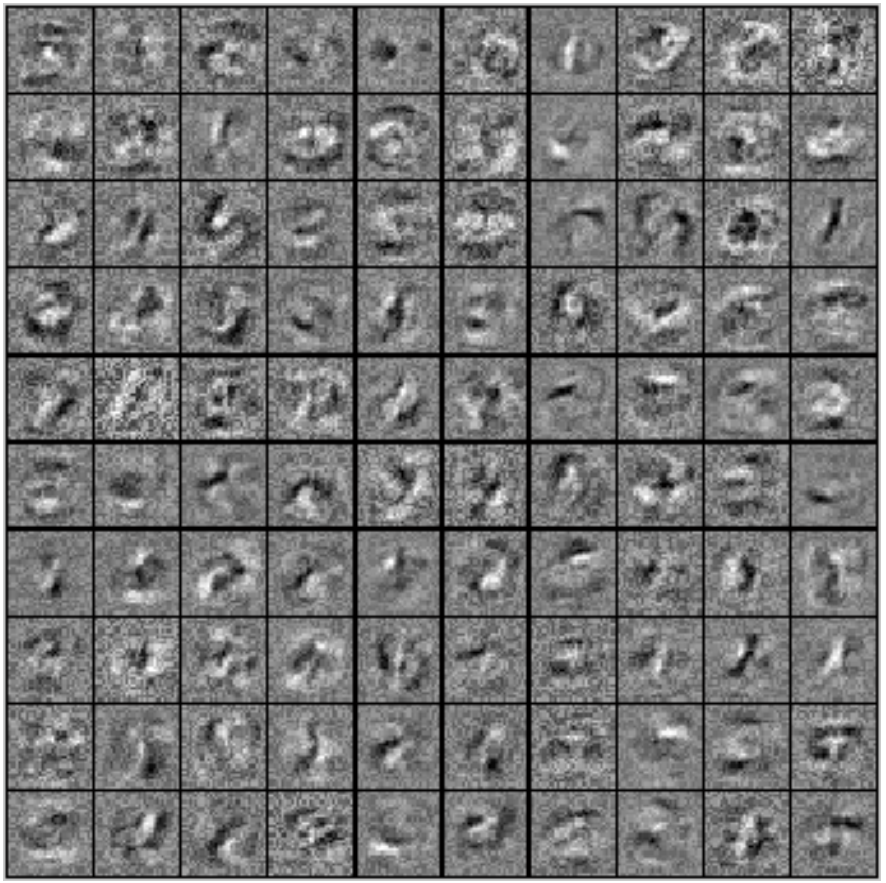
$$J(W,b)=-\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{10}\left\{y_j^{(i)}\log\left[a_j^{(3)}\left(x^{(i)}\right)\right]\right\}$$

$$+\frac{\lambda}{2m}\left\{\sum_{i=1}^{784}\sum_{j=1}^{100}\left(w_{ij}^{(1)}\right)^2+\sum_{i=1}^{100}\sum_{j=1}^{10}\left(w_{ij}^{(2)}\right)^2\right\}$$

反向传播算法

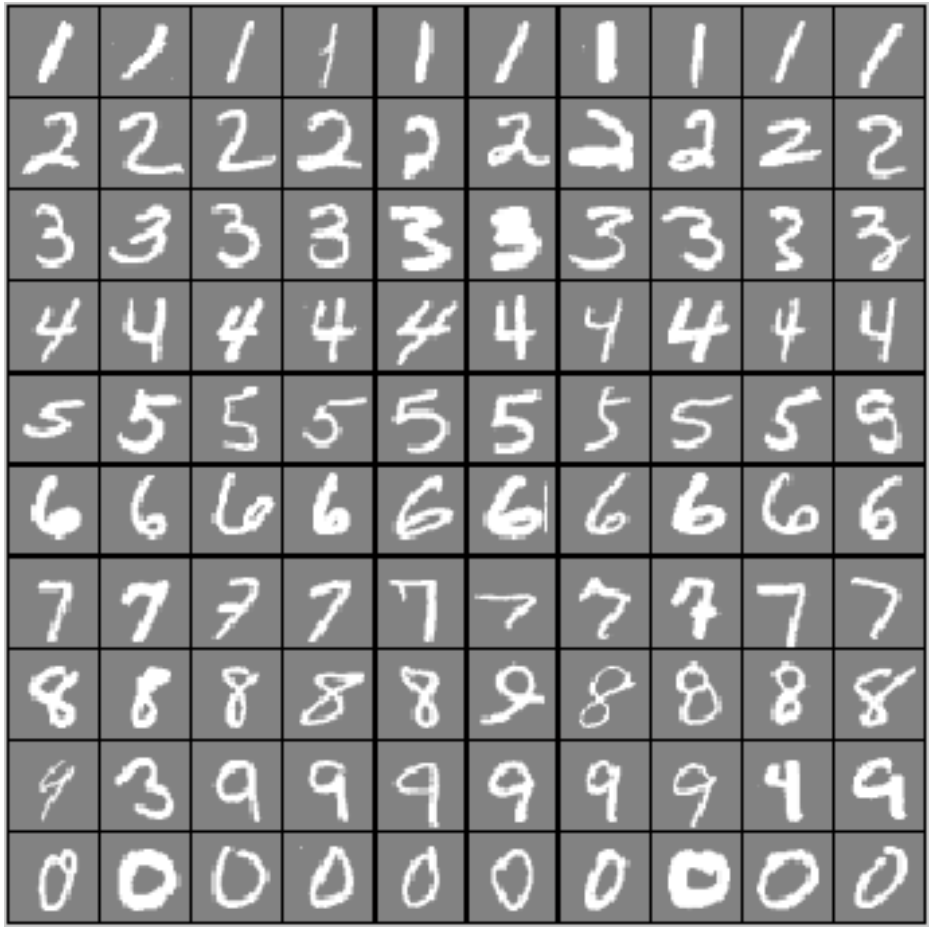
例题4：手写体数字识别-识别10个数字





反向传播算法

例题4：手写体数字识别-识别10个数字



准确率=97.3%

本节目录

问题分析

- **梯度消失问题**
  - 深度网络模型通常包含多层数据处理神经元，故此类模型的容量要比浅层学习模型大得多。但随着网络层数加深，深度网络模型易出现网络性能的退化、容易陷入局部最优等问题
  - 所谓退化，是指神经网络模型在训练集与测试集上所表现出的性能随着网络层数加深而降低的现象。理论上讲，网络层数越深则模型容量越大，网络模型应更易出现过拟合现象，但退化网络模型会出现严重欠拟合现象

问题分析

- **梯度消失问题**
  - 导致模型退化的直接原因是模型训练过程中产生的梯度消失现象，即深度模型前几层参数的梯度接近于0
  - 如果模型训练出现梯度消失现象，则每次参数更新均无法有效改变模型前几层的参数，故无论是训练过程还是测试过程，网络前几层的参数均接近于初始状态
  - 训练过程中一旦出现梯度消失现象，则整个网络模型在训练和测试过程中都无法取得良好的性能
  - 之所以会出现梯度消失现象，是因为传统神经网络模型多采用Sigmoid激活函数或tanh激活函数。由于这些激活函数的梯度任何情况下的取值均小于1，故梯度值将会在深度网络模型的训练过程中逐层衰减并最终接近于0

问题分析

- **梯度消失问题**
  - 现具体考察下图所示简单多层神经网络输入层与第一个隐含层之间连接权重w^1的梯度取值情况。假设该模型中各神经元均采用Sigmoid函数作为激活函数σ，第l层与第$l+1$层之间的连接权重为w^l，第l层神经元的偏执为b^l，则对于输入样本数据X，该模型的最终输出为：

$$f(X)=\sigma(w^3\sigma(w^2\sigma(w^1X+b^2)+b^3)+b^4)$$

问题分析

- **梯度消失问题**
  - 记模型第j层结点的输出为h^j，则目标函数为：

$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} L(f(X_i), y_i)$$

其中$\boldsymbol{W}$参数向量。此时$J(\boldsymbol{W})$对于w^1的梯度为：

$$\frac{\partial J(\boldsymbol{W})}{\partial w^1} = \frac{\partial J(\boldsymbol{W})}{\partial f(X_i)} \frac{\partial f(X_i)}{\partial h^3} \frac{\partial h^3}{\partial h^2} \frac{\partial h^2}{\partial w^1}$$

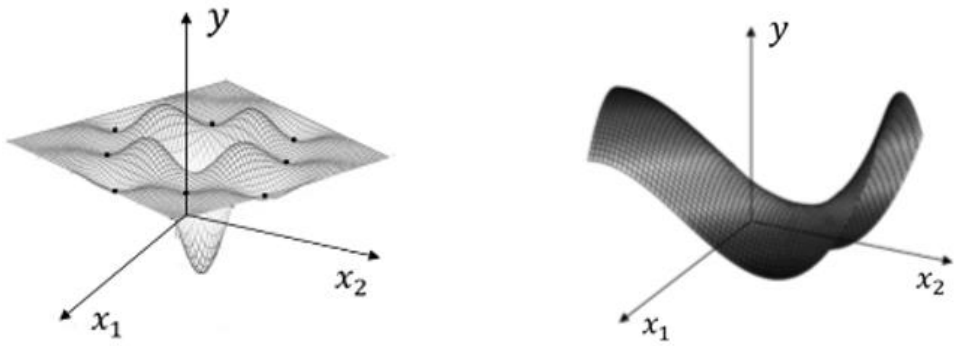
  - 上式中$\partial f(X_i)/\partial \mathrm{h}^3$和$\partial \mathrm{h}^3/\partial \mathrm{h}^2$均为Sigmoid函数的梯度，而Sigmoid函数的梯度取值均小于1。由于$\partial J(\mathrm{W})/\partial w^1$为连乘形式，故神经网络模型前一层参数$w^l$所对应的梯度取值必然比$w^{l+1}$小，当神经网络模型足够深时，该网络前几层的参数均会接近于0，从而出现梯度消失现象

问题分析

- **梯度消失问题**
  - 梯度消失问题的应对策略
    - 改进激活函数，如ReLU、Leaky ReLU
    - 中间层引入损失函数，如GoogLeNet
    - 残差连接，如ResNet

问题分析

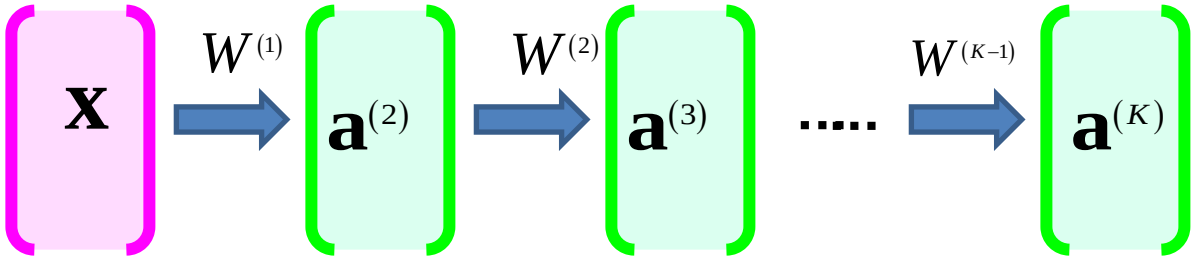
- **局部最优问题**
  - 深度学习面临的另外一个挑战是模型易陷入局部最优。这是由于深度网络模型目标函数过于复杂而导致
  - 对于过于复杂的函数，难免会存在多个方向上梯度取值均为0的非最优点，如局部最小值点和鞍点等
    - 下图（左、右）分别展示了三维空间中函数局部最小值点和鞍点，由于函数在这些点处对于任意参数的梯度取值均为0，故在模型优化过程中一旦陷入局部最小值点或鞍点，则参数取值很难再发生变化。此时模型虽并未达到最优状态，但模型参数已收敛，故难以保证所求模型满足实际需求



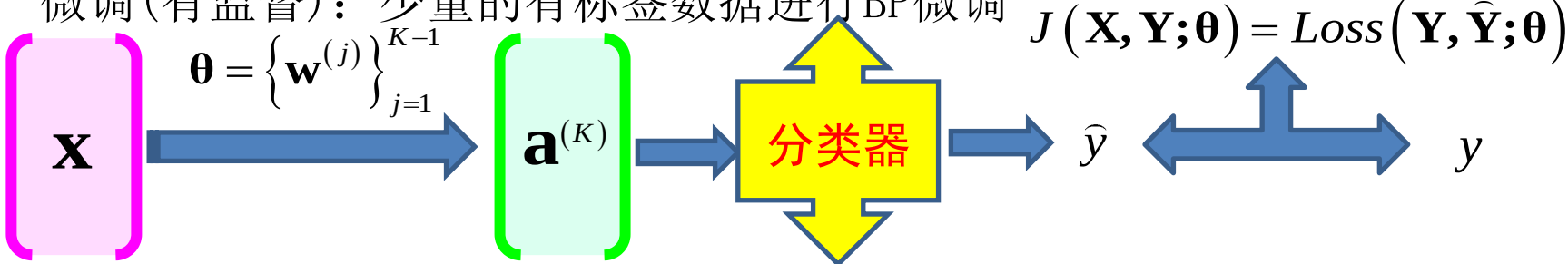
问题分析

- **局部最优问题**
  - 局部最优问题的应对策略
    - 改进初始化方式，如DAE、DBN
    - 随机梯度下降，跳出局部极值
    - 设置冲量，加速优化

1. 预训练(无监督):大量的无标签数据集进行逐层预训练



2. 微调(有监督)：少量的有标签数据进行BP微调



本节目录

思考题

- 神经网络模型优化过程中为什么一般要对数据进行归一化
- BP神经网络的训练次数是否越多越好
- 激活函数主要有哪几种，各有什么优缺点

练习题

- 针对二分类任务，试将反向传播算法部分的例题1的输出改为单个神经元，并设计一些简单训练样本，完成一次连接权重更新