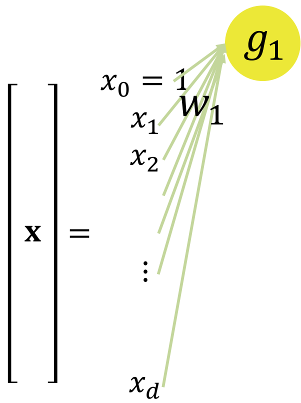
人工智能与自动化学院

模式识别

第十讲 神经网络到深度学习(*From Neural Networks to Deep Learning*)

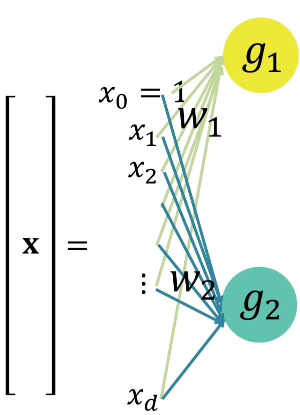
10.1 神经网络动机

多个感知器的线性集成(*Linear Aggregation of Perceptrons*)



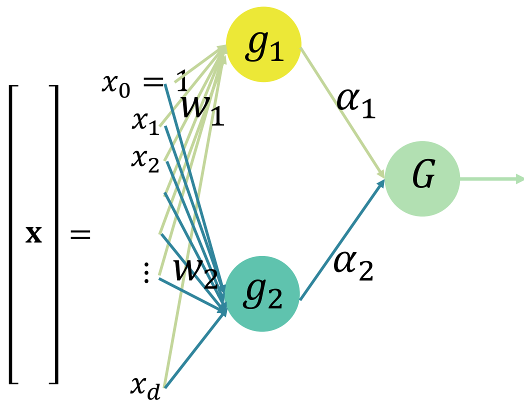
10.1 神经网络动机

多个感知器的线性集成(*Linear Aggregation of Perceptrons*)



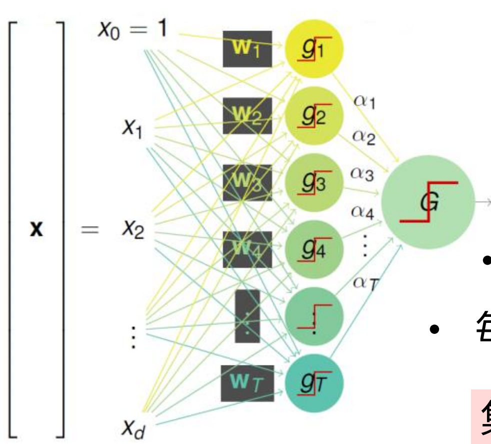
10.1 神经网络动机

多个感知器的线性集成(*Linear Aggregation of Perceptrons*)



10.1 神经网络动机

多个感知器的线性集成(*Linear Aggregation of Perceptrons*)



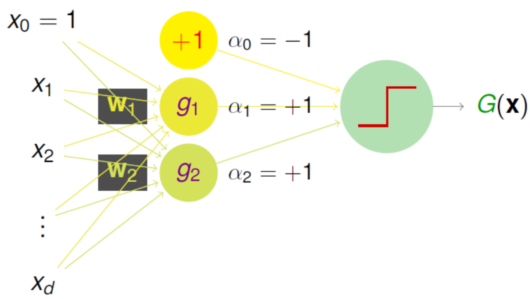
$$G(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t \underbrace{sign(\mathbf{w}_t^T \mathbf{x})}_{g_t(\mathbf{x})}\right)$$

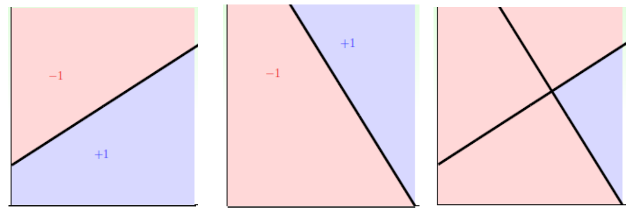
- 一共两层，每层都有权向量： $\mathbf{w}_t$和$\boldsymbol{\alpha}$
- 每层用符号函数*sign*()分别得到： g_t和G

集成后得到分类面G的边界是怎样的?

10.1 神经网络动机

用集成方式实现逻辑运算





g_1 g_2 $\text{AND}(g_1, g_2)$

$$G(\mathbf{x}) = sign(-1 + g_1(\mathbf{x}) + g_2(\mathbf{x}))$$

$if \;\; g_1(\mathbf{x}) = g_2(\mathbf{x}) = +1: \quad G(\mathbf{x}) = +1$

$otherwise: \quad G(\mathbf{x}) = -1$

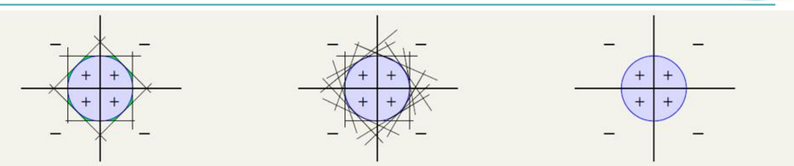
集成也能实现逻辑运算*OR*、*NOT*

$$G(\mathbf{x}) \equiv AND(g_1(\mathbf{x}), g_2(\mathbf{x}))$$

10.1 神经网络动机

两层集成的强大能力：

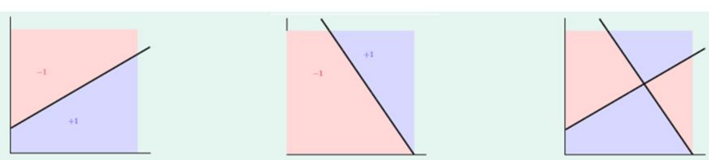
足够多的感知器可得到光滑的非线性分类面



8 perceptrons 16 perceptrons target boundary

两层集成的不足：

在两层结构下，做不到“线性可分”，无法实现“异或(*XOR*)”逻辑



g_1 g_2 $\text{XOR}(g_1, g_2)$

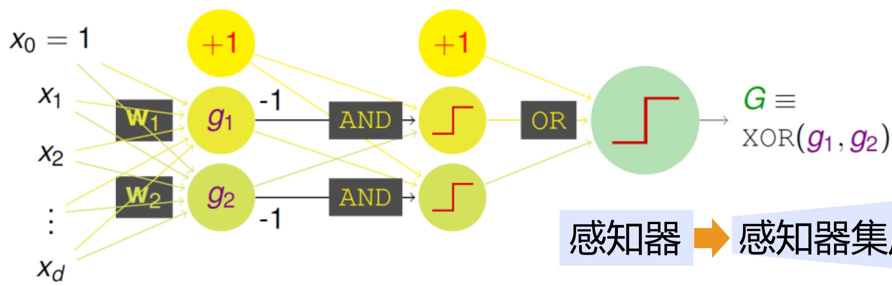
10.1 神经网络动机

如何实现“异或”逻辑?

- 途径1：非线性变换
- 途径2：在两层实现简单逻辑基础上，再增加层数

$$XOR(g_1, g_2) = OR(AND(-g_1, g_2) + AND(g_1, -g_2))$$

多层感知机—基本的神经网络



感知器 ➡ 感知器集成 ➡ 多层感知机

第十讲 神经网络到深度学习(*From Neural Networks to Deep Learning*)

10.2 神经网络模型参数空间

感知器算法再回顾

输入图像

特征表达

分类器模型

g

预测结果$\hat{y}_n$

损失函数

误差度量

真实结果y_n

f

分类器模型参数空间$\mathcal{H}(\cdot)$

$\mathcal{A}$

参数优化

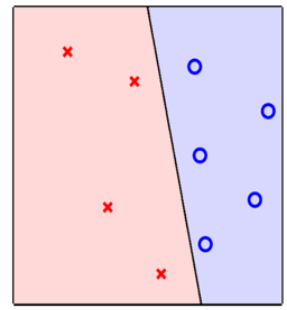
$$\hat{y}_{n(t)} = \mathrm{sign}(\boldsymbol{w}_t^T \mathbf{x}_{n(t)})$$

算法收敛：

$$L_{in} = \sum_{n=1}^{N} [\![y_n \neq \hat{y}_n]\!] = 0$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_n \mathbf{x}_{n(t)}$$

线性可分



- 设置初始分类面(权重)$\boldsymbol{w}_0$
- 如果有样本分错，就修正权重

10.2 神经网络模型参数空间

神经网络模型

输入图像

特征表达

分类器模型

g

预测结果$\hat{y}_n$

损失函数

误差度量

真实结果y_n

f

分类器模型参数空间$\mathcal{H}(\cdot)$

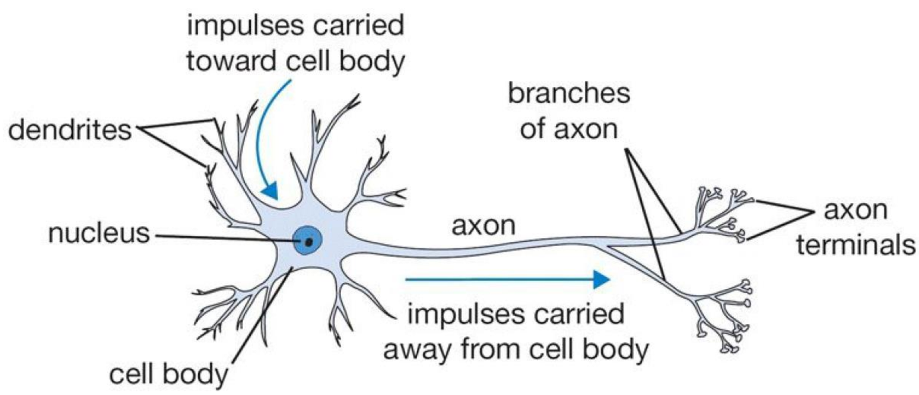
$\mathcal{A}$

参数优化

神经网络模型参数空间?

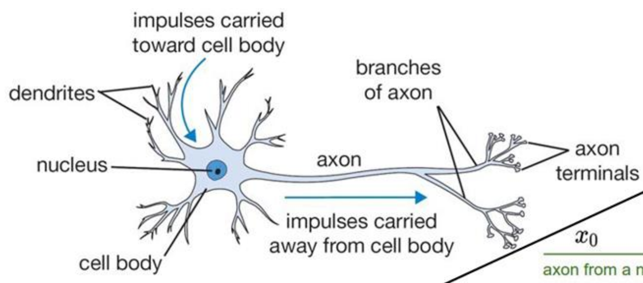
10.2 神经网络模型参数空间

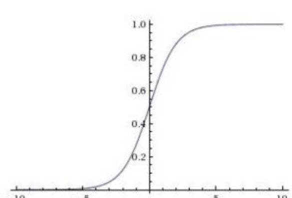
神经网络(*Neural Networks*):



10.2 神经网络模型参数空间

神经网络(*Neural Networks*):

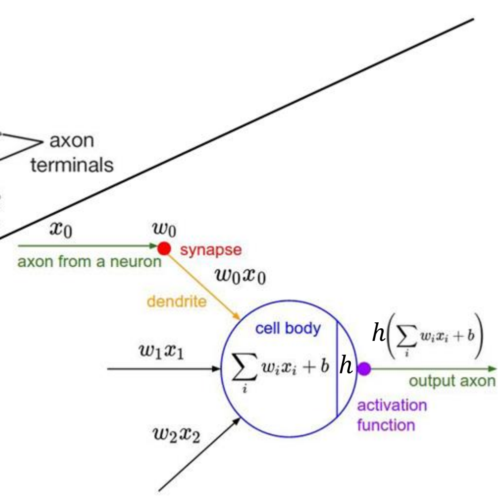




Sigmoid 激活函数

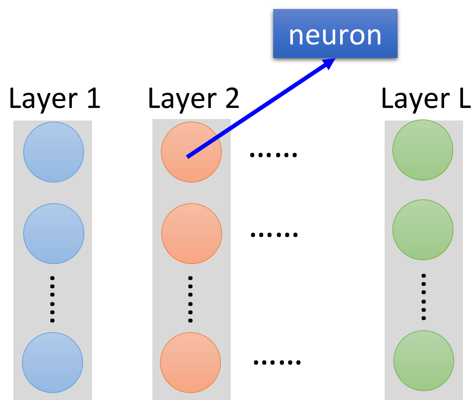
(*Sigmoid activation function*)

$$\frac{1}{1+e^{-x}}$$



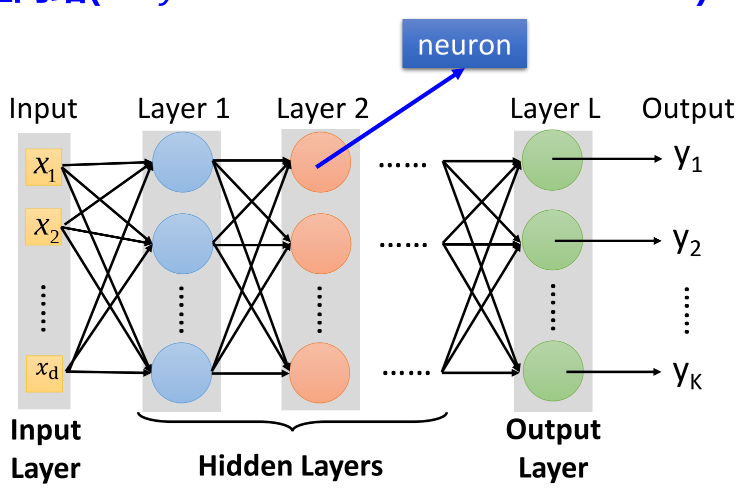
10.2 神经网络模型参数空间

神经网络(*Neural Networks*):



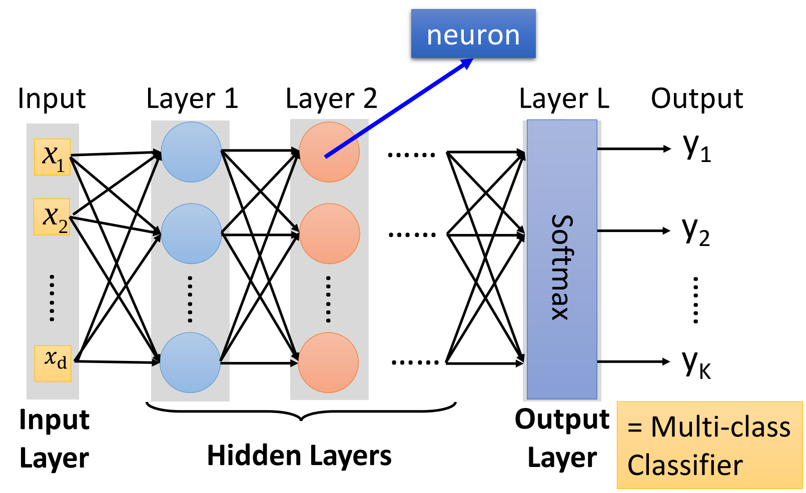
10.2 神经网络模型参数空间

全链接前馈神经网络(*Fully Connect Feedforward Network*):



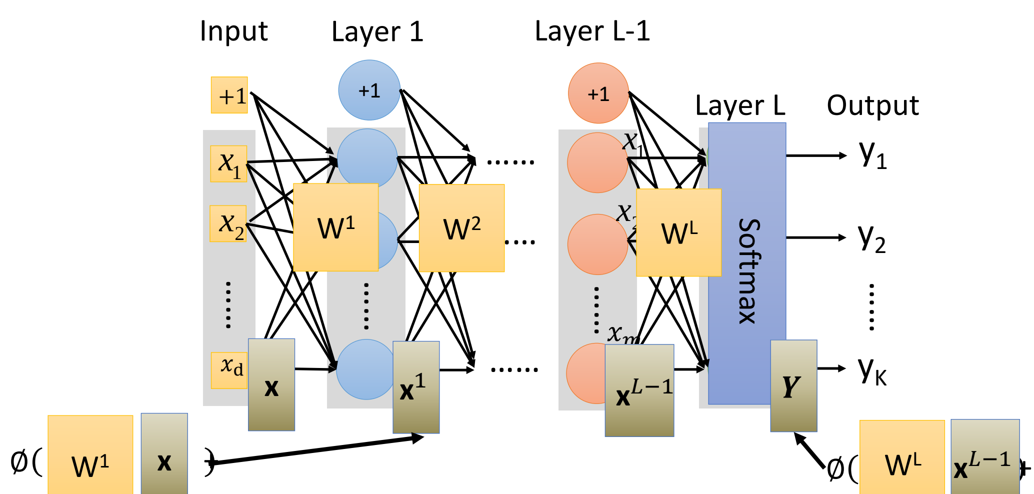
10.2 神经网络模型参数空间

全链接前馈神经网络(*Fully Connect Feedforward Network*):



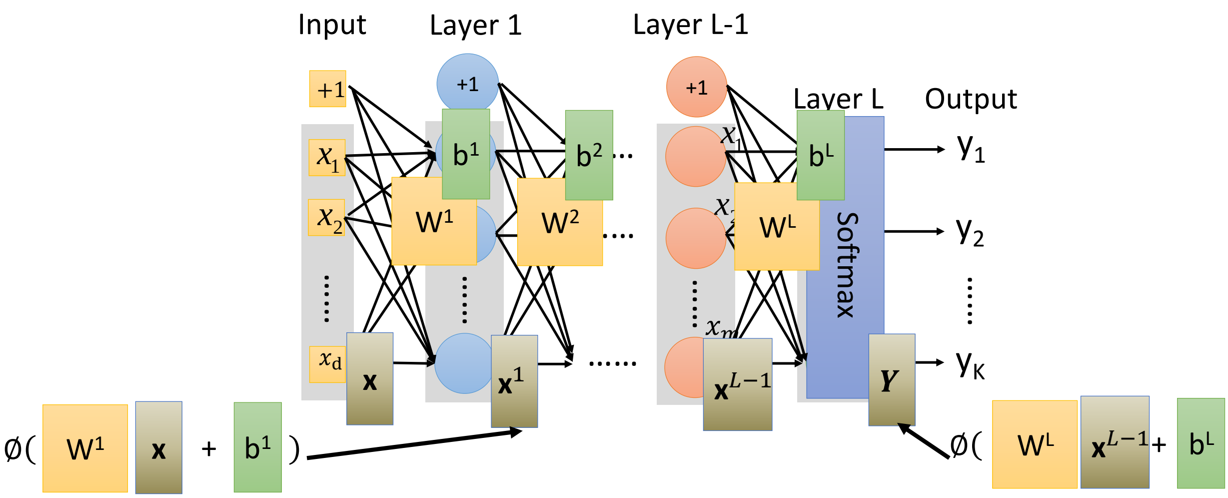
10.2 神经网络模型参数空间

全链接前馈神经网络(*Fully Connect Feedforward Network*):



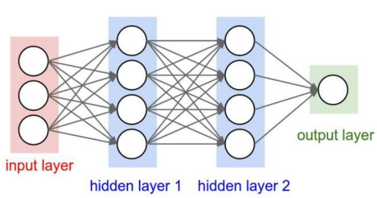
10.2 神经网络模型参数空间

全链接前馈神经网络(*Fully Connect Feedforward Network*):



10.2 神经网络模型参数空间

神经网络模型:



为简单起见，输出端考虑二分类问题

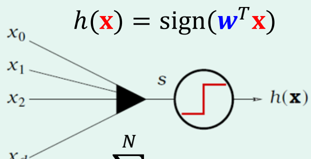
为简单起见，输出层选择线性回归模型

Output: $S = \mathbf{w}^T\phi^{(2)}(\phi^{(1)}(\mathbf{x}))$

$$L_{in} = \frac{1}{N}\sum_{n=1}^{N}(S - y_n)^2$$

线性分类(感知器):

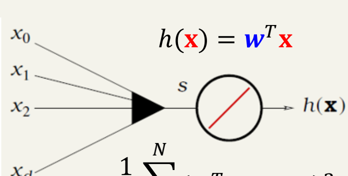
$h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})$



$$L_{in} = \sum_{n=1}^{N}[\![y_n \neq \hat{y}_n]\!]$$

线性回归:

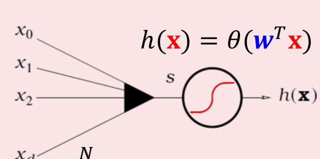
$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$



$$L_{in} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2$$

逻辑斯蒂回归:

$h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$

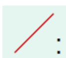


$$L_{in} = \frac{1}{N}\sum_{n=1}^{N}\ln(1 + exp(-y_n\mathbf{w}^T\mathbf{x}_n))$$

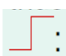
10. 2 神经网络模型参数空间

激活(activation)/变换(transformation)函数:

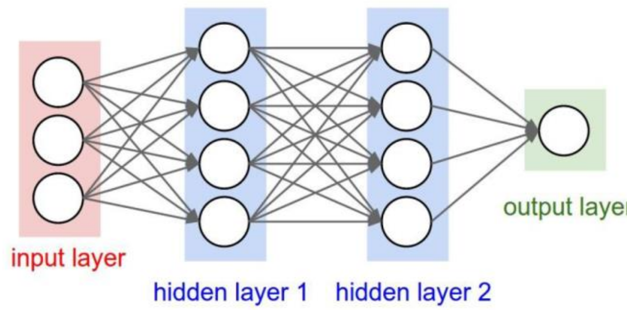
激活函数可以是任意形式吗?

 : 线性网络，作用不大

$$s = \boldsymbol{w}^{(3)}\left(\boldsymbol{w}^{(2)}\left(\boldsymbol{w}^{(1)}\mathbf{x} + b^{(1)}\right) + b^{(2)}\right) + b^{(3)}$$
$$= \boldsymbol{W}\mathbf{x} + b$$

 : NP难问题，难以获得最佳的$\boldsymbol{w}$

 : 非线性变换，增强网络的性能



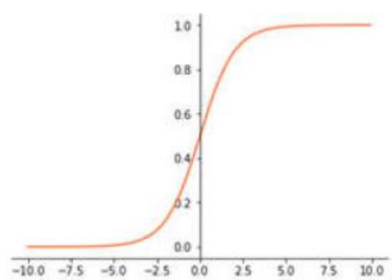
$$s = \boldsymbol{w}^{(3)}\phi\left(\boldsymbol{w}^{(2)}\phi\left(\boldsymbol{w}^{(1)}\mathbf{x} + b^{(1)}\right) + b^{(2)}\right) + b^{(3)}$$

10. 2 神经网络模型参数空间

激活(activation)/变换(transformation)函数:

Sigmoid

$1/(1+e^{-x})$

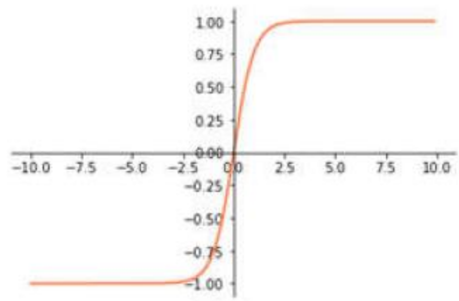


10.2 神经网络模型参数空间

激活(activation)/变换(transformation)函数:

tanh

$$\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

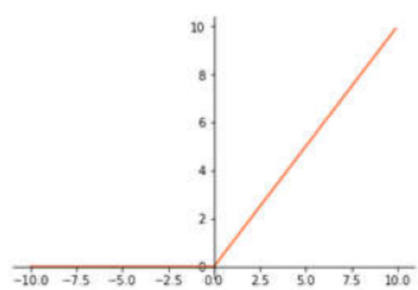


10.2 神经网络模型参数空间

激活(activation)/变换(transformation)函数:

ReLU

$$\max(0, x)$$

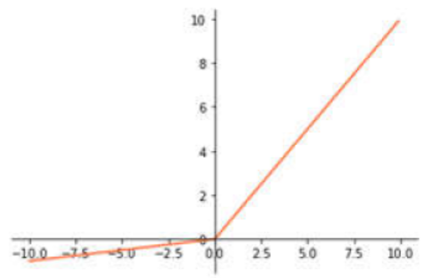


10. 2 神经网络模型参数空间

激活(activation)/变换(transformation)函数：

Leaky ReLU

$\max(0.1x, x)$

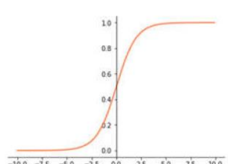


10. 2 神经网络模型参数空间

激活(activation)/变换(transformation)函数：

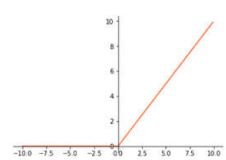
Sigmoid

$1/(1+e^{-x})$



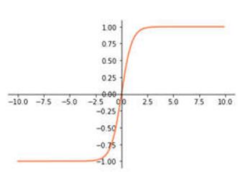
ReLU

$\max(0, x)$



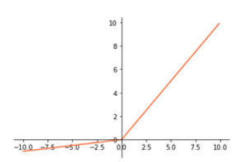
tanh

$\dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$



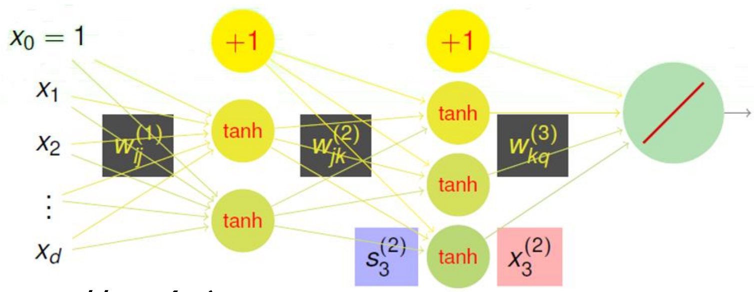
Leaky ReLU

$\max(0.1x, x)$



- 每一个感知器单元都应具有非线性特性
- 非线性函数应该具有可导性质便于训练

10.2 神经网络模型参数空间



符号含义:

$d^{(l)}$: 网络的第l层

$\mathbf{w}_{ij}^{(l)}$: 网络的第$l-1$层到第l层的权系数矩阵

$s_j^{(l)}$: 网络第l层第j个神经元的输入

$x_j^{(l)}$: 网络第l层第j个神经元的输出

$$\mathbf{w}_{ij}^{(l)}: \begin{cases} 1 \le l \le L & \text{layers} \\ 0 \le i \le d^{(l-1)} & \text{inputs} \\ 1 \le j \le d^{(l)} & \text{outputs} \end{cases}$$

$$s_j^{(l)} = \sum_{i=0}^{d^{(l-1)}} \mathbf{w}_{ij}^{(l)} x_i^{(l-1)}$$

$$x_j^{(l)} = \begin{cases} \tanh(s_j^{(l)}) & if\ l < L \\ s_j^{(l)} & if\ l = L \end{cases}$$

将输入样本$\mathbf{x}$作为input layer，经过hidden layers得到$\mathbf{x}^{(l)}$，在output layer去预测$x_1^{(L)}$

第十讲 神经网络到深度学习(*From Neural Networks to Deep Learning*)

10.1 神经网络动机（*Motivation of Neural Networks*）

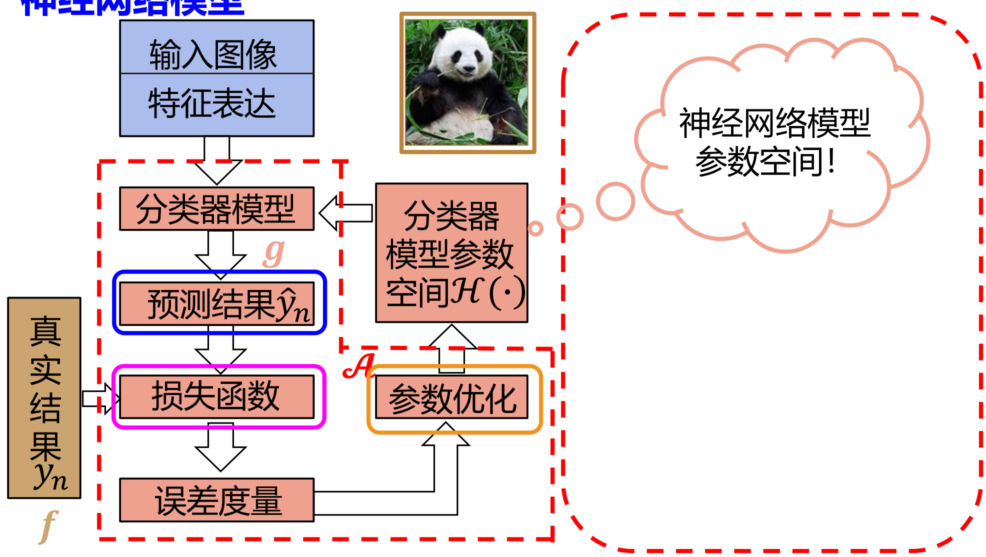
10.2 神经网络模型(*Neural Network Hypothesis*)

10.3 神经网络学习（*Neural Network Learning*）

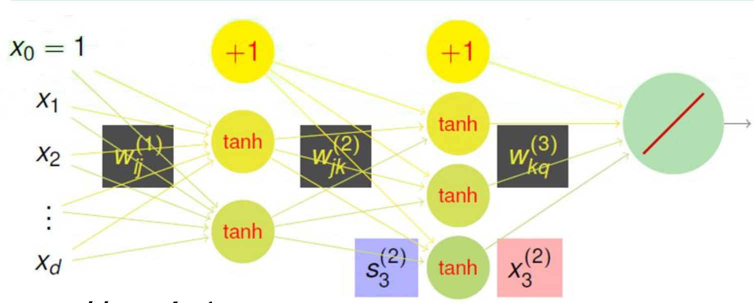
10.4 深度神经网络（*Deep Neural Networks*）

10.3 神经网络学习

神经网络模型



10.3 神经网络学习



符号含义：

$d^{(l)}$：网络的第l层

$\boldsymbol{w}_{ij}^{(l)}$：网络的第$l-1$层到第$l$层的权系数矩阵

$S_j^{(l)}$：网络第l层第j个神经元的输入

$x_j^{(l)}$：网络第l层第j个神经元的输出

$$\boldsymbol{w}_{ij}^{(l)}: \begin{cases} 1 \le l \le L & \text{layers} \\ 0 \le i \le d^{(l-1)} & \text{inputs} \\ 1 \le j \le d^{(l)} & \text{outputs} \end{cases}$$

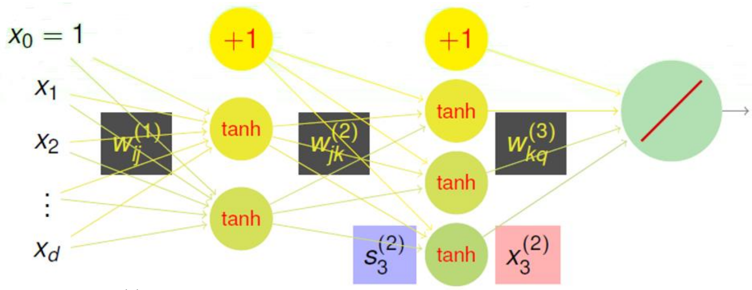
$$S_j^{(l)} = \sum_{i=0}^{d^{(l-1)}} \boldsymbol{w}_{ij}^{(l)} x_i^{(l-1)}$$

$$x_j^{(l)} = \begin{cases} \tanh(S_j^{(l)}) & if\ l < L \\ S_j^{(l)} & if\ l = L \end{cases}$$

将输入样本$\mathbf{x}$作为input layer，经过hidden layers得到$\mathbf{x}^{(l)}$，在output layer去预测$x_1^{(L)}$

10.3 神经网络学习

如何学习网络权重 $w_{ij}^{(l)}$ ？



$d^{(l)}$：网络的第l层

$\mathbf{w}_{ij}^{(l)}$：网络的第$l-1$层到第l层的权系数矩阵

$s_j^{(l)}$：网络第l层第j个神经元的输入

$x_j^{(l)}$：网络第l层第j个神经元的输出

$$\mathbf{w}_{ij}^{(l)}: \begin{cases} 1 \le l \le L & \text{layers} \\ 0 \le i \le d^{(l-1)} & \text{inputs} \\ 1 \le j \le d^{(l)} & \text{outputs} \end{cases}$$

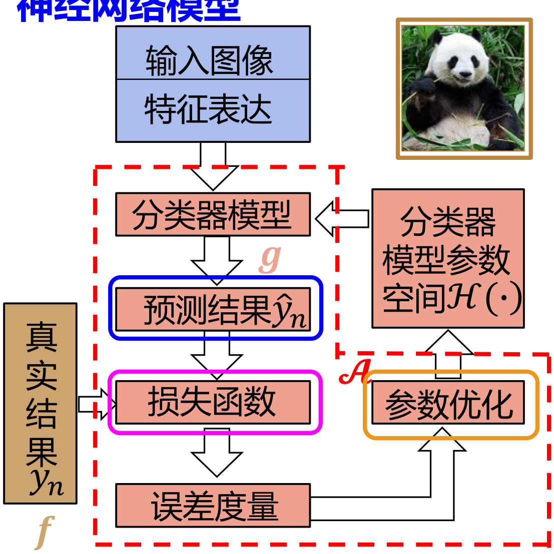
$$s_j^{(l)} = \sum_{i=0}^{d^{(l-1)}} \mathbf{w}_{ij}^{(l)} x_i^{(l-1)}$$

$$x_j^{(l)} = \begin{cases} \tanh(s_j^{(l)}) & if\ l < L \\ s_j^{(l)} & if\ l = L \end{cases}$$

将输入样本**x**作为input layer，经过hidden layers得到$\mathbf{x}^{(l)}$，在output layer去预测$x_1^{(L)}$

10.3 神经网络学习

神经网络模型

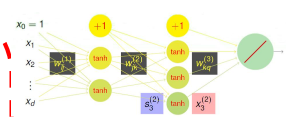


$$\hat{y}_n = NNet(\mathbf{x}_n)$$

$$L_n = (y_n - NNet(\mathbf{x}_n))^2$$

参数优化? 随机梯度下降法

求解网络中所有的$\mathbf{w}_{ij}^{(l)}$，使L_n最小化，即为最佳解



$$\mathbf{w}_{ij}^{(l)}: \begin{cases} 1 \le l \le L \\ 0 \le i \le d^{(l-1)} \\ 1 \le j \le d^{(l)} \end{cases}$$

$$s_j^{(l)} = \sum_{i=0}^{d^{(l-1)}} \mathbf{w}_{ij}^{(l)} x_i^{(l-1)}$$

$$x_j^{(l)} = \begin{cases} \tanh(s_j^{(l)}) & if\ l < L \\ s_j^{(l)} & if\ l = L \end{cases}$$

10.3 神经网络学习

神经网络中如何求解梯度： $\dfrac{\partial L_n}{\partial \mathbf{w}_{ij}^{(l)}}$

$$L_n = (y_n - NNet(\mathbf{x}_n))^2 = (y_n - S_1^{(L)})^2 = (y_n - \sum_{i=0}^{d^{(L-1)}} \mathbf{w}_{i1}^{(L)} x_i^{(L-1)})^2$$

网络输出层：第L层

$(0 \le i \le d^{(L-1)}, j = 1, l = L)$

$$\frac{\partial L_n}{\partial \mathbf{w}_{i1}^{(L)}} = \frac{\partial L_n}{\partial S_1^{(L)}} \cdot \frac{\partial S_1^{(L)}}{\partial \mathbf{w}_{i1}^{(L)}}$$

$$= -2(y_n - S_1^{(L)}) \cdot (x_i^{(L-1)})$$

$$\delta_j^{(L)} = -2(y_n - S_1^{(L)})$$

网络其他层：第l层

$(0 \le i \le d^{(l-1)}, 1 \le j \le d^{(l)})$

$$\frac{\partial L_n}{\partial \mathbf{w}_{ij}^{(l)}} = \frac{\partial L_n}{\partial S_j^{(l)}} \cdot \frac{\partial S_j^{(l)}}{\partial \mathbf{w}_{ij}^{(l)}}$$

$$= \delta_j^{(l)} \cdot (x_i^{(l-1)})$$

$$\delta_j^{(l)} = ?$$

$$\mathbf{w}_{ij}^{(l)} : \begin{cases} 1 \le l \le L \\ 0 \le i \le d^{(l-1)} \\ 1 \le j \le d^{(l)} \end{cases}$$

$$S_j^{(l)} = \sum_{i=0}^{d^{(l-1)}} \mathbf{w}_{ij}^{(l)} x_i^{(l-1)}$$

$$x_j^{(l)} = \begin{cases} \tanh(S_j^{(l)}) & if\ l < L \\ S_j^{(l)} & if\ l = L \end{cases}$$

10.3 神经网络学习

神经网络中如何求解梯度： $\delta_j^{(l)} = \dfrac{\partial L_n}{\partial S_j^{(l)}}$

$$S_j^{(l)} \xrightarrow{\tanh} x_j^{(l)} \xrightarrow{\mathbf{w}_{jk}^{(l+1)}} \begin{pmatrix} S_1^{(l+1)} \\ \vdots \\ S_k^{(l+1)} \\ \vdots \end{pmatrix} \rightarrow \cdots \rightarrow L_n$$

$$\delta_j^{(l)} = \frac{\partial L_n}{\partial S_j^{(l)}} = \sum_{k=1}^{d^{(l+1)}} \frac{\partial L_n}{\partial S_k^{(l+1)}} \cdot \frac{\partial S_k^{(l+1)}}{\partial x_j^{(l)}} \cdot \frac{\partial x_j^{(l)}}{\partial S_j^{(l)}}$$

$$= \sum_{k=1}^{d^{(l+1)}} (\delta_k^{(l+1)}) \cdot (\mathbf{w}_{jk}^{(l+1)}) \cdot (\tanh(S_j^{(l)}))$$

$\delta_j^{(l)}$能够通过反传$\delta_k^{(l+1)}$求得

$$\mathbf{w}_{ij}^{(l)} : \begin{cases} 1 \le l \le L \\ 0 \le i \le d^{(l-1)} \\ 1 \le j \le d^{(l)} \end{cases}$$

$$S_j^{(l)} = \sum_{i=0}^{d^{(l-1)}} \mathbf{w}_{ij}^{(l)} x_i^{(l-1)}$$

$$x_j^{(l)} = \begin{cases} \tanh(S_j^{(l)}) & if\ l < L \\ S_j^{(l)} & if\ l = L \end{cases}$$

10.3 神经网络学习

利用反向传播算法实现神经网络学习(*Back propagation (Backprop) Algorithm*)

初始化所有的权系数$w_{ij}^{(l)}$

$for\ t = 0,1,\dots T$

① 随机性(*stochastic*)：从训练样本集中任选一个$\mathbf{x}_n$，$n \in \{1,2,\dots N\}$

② 前向传播(*forward*)：利用$\mathbf{x}^{(0)} = \mathbf{x}_n$计算所有的$x_i^{(l)}$

③ 后向传播(*backward*)：在$\mathbf{x}^{(0)} = \mathbf{x}_n$基础上，计算所有的$\delta_j^{(l)}$

④ 梯度下降(*gradient descent*)：$\mathbf{w}_{ij}^{(l)} \leftarrow \mathbf{w}_{ij}^{(l)} - \eta x_i^{(l-1)} \delta_j^{(l)}$

返回结果：$g_{NNET}(\mathbf{x}) = \left(\cdots \tanh\left(\sum_j \mathbf{w}_{jk}^{(2)} \cdot \tanh\left(\sum_i \mathbf{w}_{ij}^{(1)} x_i\right)\right)\right)$

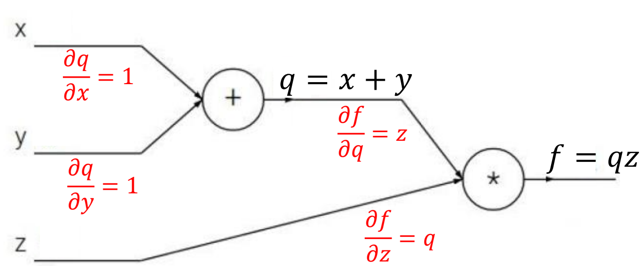
- 所有梯度下降法的技巧都可使用
- 步骤①②③可并行处理，提高效率

反向传播算法是当前神经网络学习的有效技术

10.3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*)：

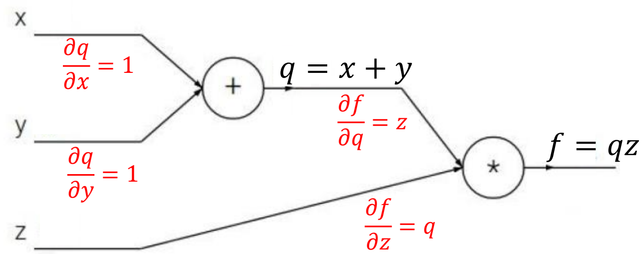
$f = (x + y)z$



10.3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*):

$$f = (x + y)z$$



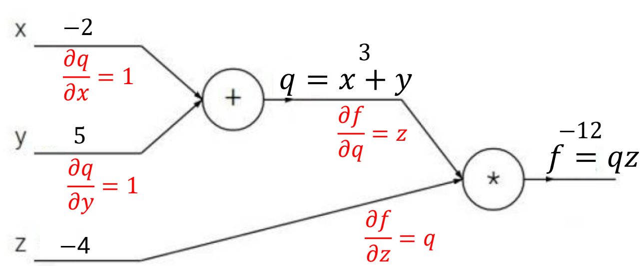
$if: \quad x = -2, y = 5, z = -4$

$want: \quad \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

10.3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*):

$$f = (x + y)z$$



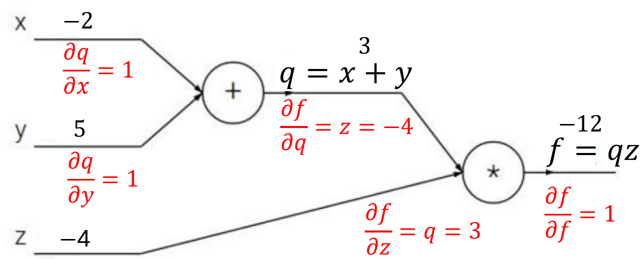
$if: \quad x = -2, y = 5, z = -4$

$want: \quad \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

10.3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*):

$f = (x+y)z$



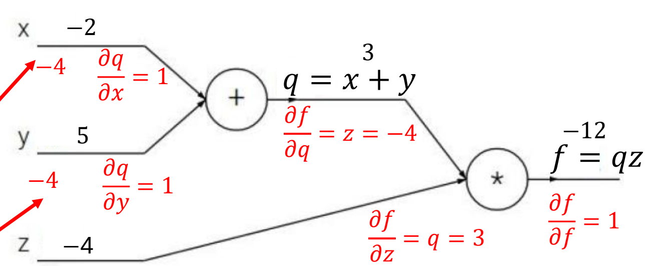
$if: \quad x = -2, y = 5, z = -4$

$want: \quad \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

10.3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*):

$f = (x+y)z$



$if: \quad x = -2, y = 5, z = -4$

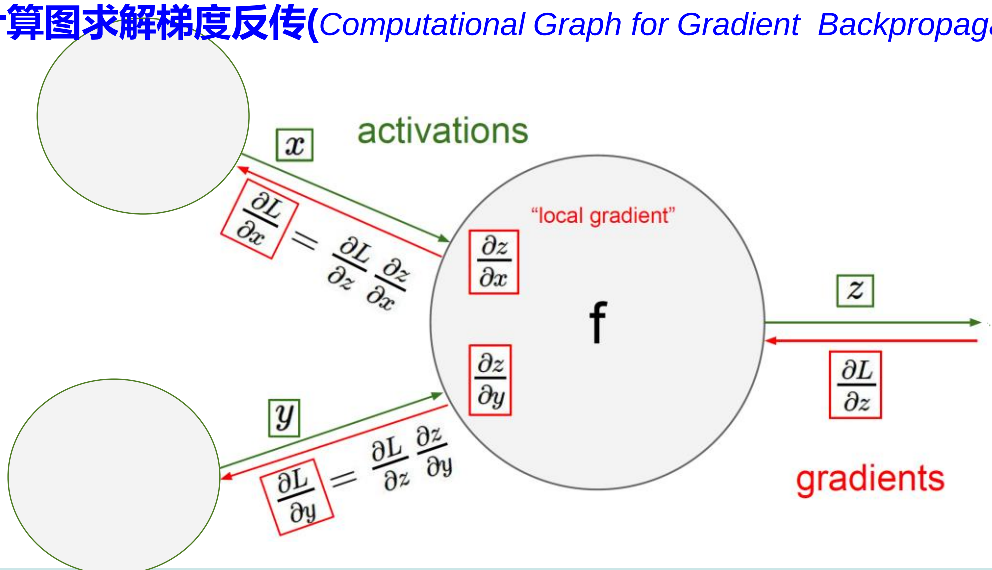
$want: \quad \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Chain Rule: $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$

$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$

10.3 神经网络学习

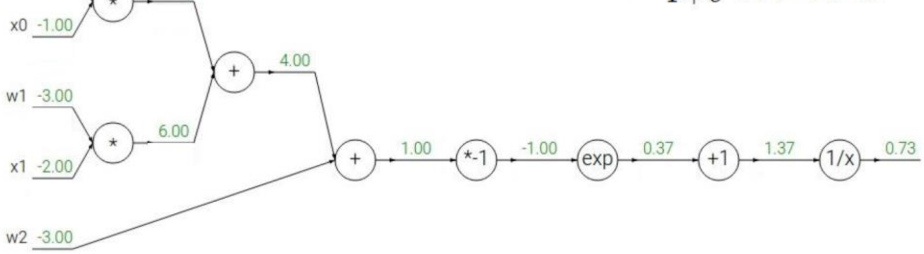
用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*)：



10.3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*)：

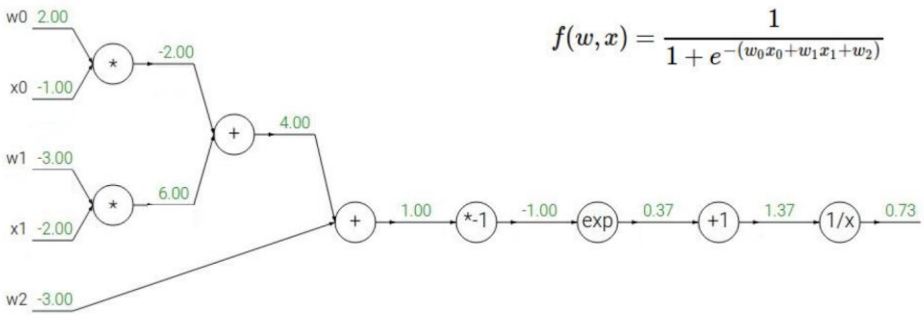
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



10. 3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*)：

$$f(w,x)=\frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



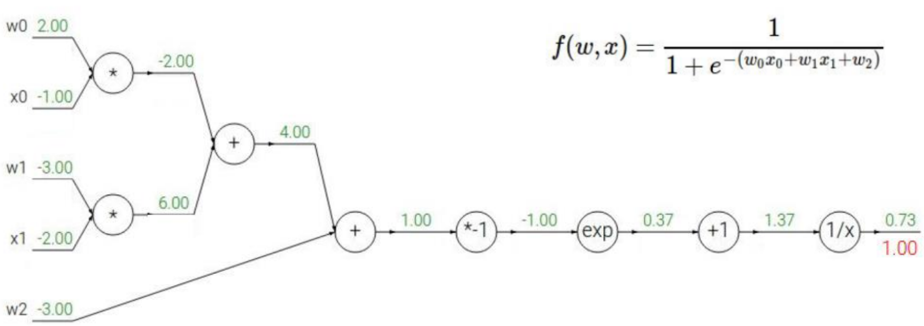
$f(x)=e^x \quad \rightarrow \quad \frac{df}{dx}=e^x$ | $f(x)=\frac{1}{x} \quad \rightarrow \quad \frac{df}{dx}=-1/x^2$

$f_a(x)=ax \quad \rightarrow \quad \frac{df}{dx}=a$ | $f_c(x)=c+x \quad \rightarrow \quad \frac{df}{dx}=1$

10. 3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*)：

$$f(w,x)=\frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



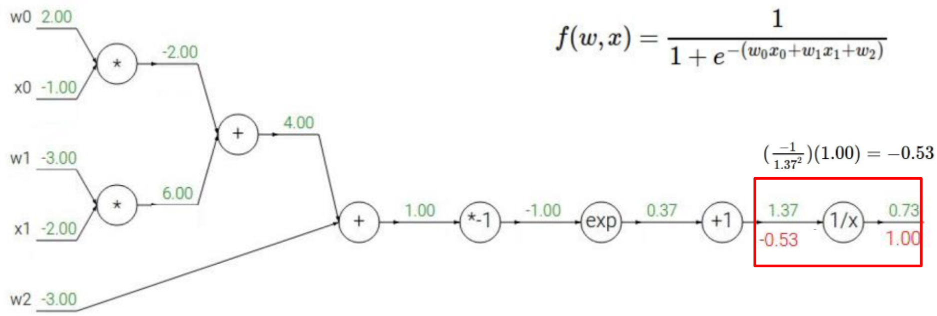
$f(x)=e^x \quad \rightarrow \quad \frac{df}{dx}=e^x$ | $f(x)=\frac{1}{x} \quad \rightarrow \quad \frac{df}{dx}=-1/x^2$

$f_a(x)=ax \quad \rightarrow \quad \frac{df}{dx}=a$ | $f_c(x)=c+x \quad \rightarrow \quad \frac{df}{dx}=1$

10.3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*)：

$$f(w,x)=\frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



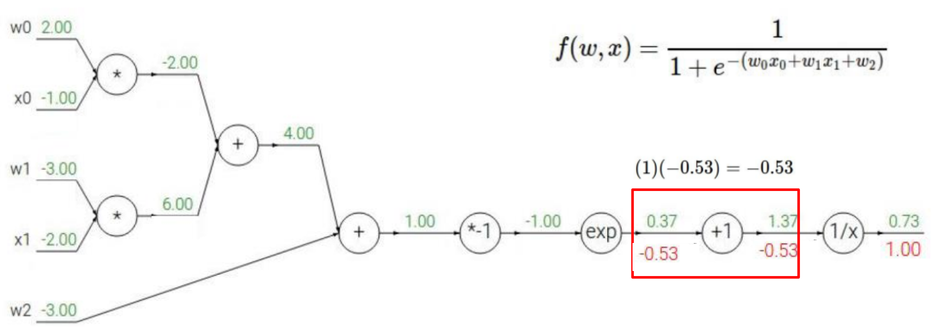
$f(x)=e^x \quad\rightarrow\quad \frac{df}{dx}=e^x$ | $\boxed{f(x)=\frac{1}{x} \quad\rightarrow\quad \frac{df}{dx}=-1/x^2}$

$f_a(x)=ax \quad\rightarrow\quad \frac{df}{dx}=a$ | $f_c(x)=c+x \quad\rightarrow\quad \frac{df}{dx}=1$

10.3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*)：

$$f(w,x)=\frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



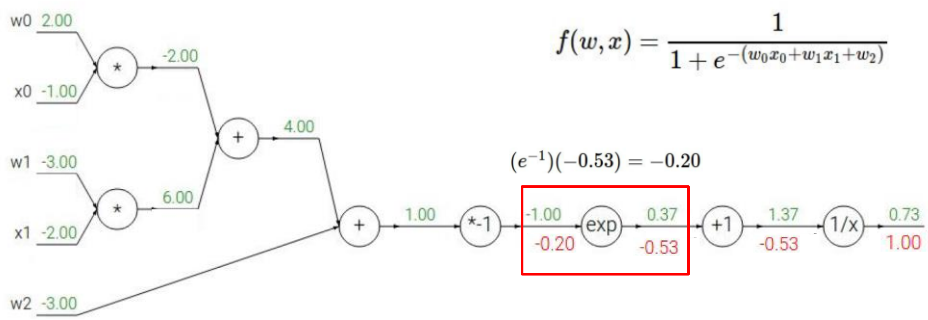
$f(x)=e^x \quad\rightarrow\quad \frac{df}{dx}=e^x$ | $f(x)=\frac{1}{x} \quad\rightarrow\quad \frac{df}{dx}=-1/x^2$

$f_a(x)=ax \quad\rightarrow\quad \frac{df}{dx}=a$ | $\boxed{f_c(x)=c+x \quad\rightarrow\quad \frac{df}{dx}=1}$

10.3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*)：

$$f(w,x)=\frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

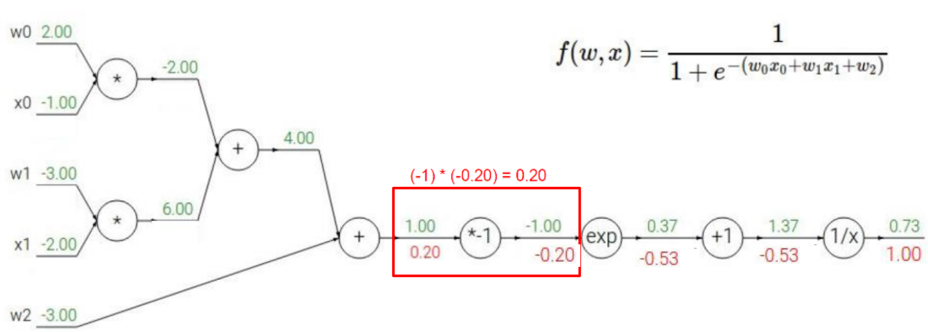


$\boxed{f(x)=e^x \quad \rightarrow \quad \frac{df}{dx}=e^x}$		$f(x)=\frac{1}{x}$ $\rightarrow$	$\frac{df}{dx}=-1/x^2$
$f_a(x)=ax$ $\rightarrow$	$\frac{df}{dx}=a$	$f_c(x)=c+x$ $\rightarrow$	$\frac{df}{dx}=1$

10.3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*)：

$$f(w,x)=\frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

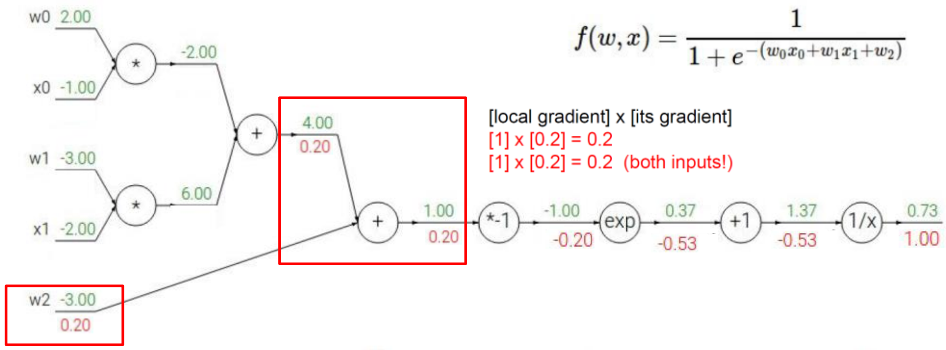


$f(x)=e^x$ $\rightarrow$	$\frac{df}{dx}=e^x$	$f(x)=\frac{1}{x}$ $\rightarrow$	$\frac{df}{dx}=-1/x^2$
$\boxed{f_a(x)=ax \quad \rightarrow \quad \frac{df}{dx}=a}$		$f_c(x)=c+x$ $\rightarrow$	$\frac{df}{dx}=1$

10. 3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*)：

$$f(w,x)=\frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



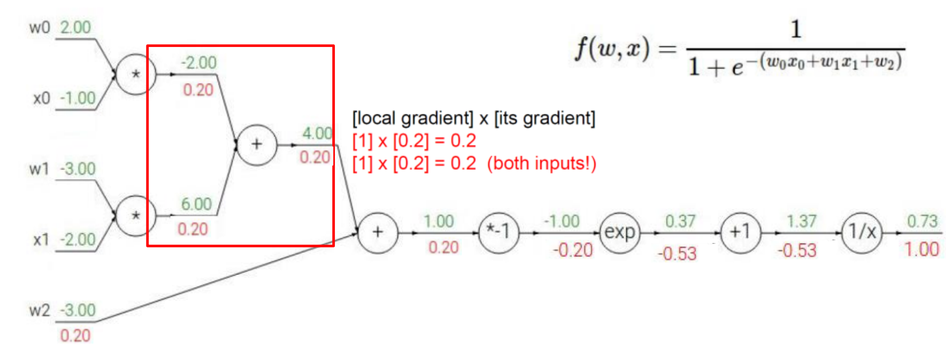
$f(x)=e^x \quad \rightarrow \quad \frac{df}{dx}=e^x$ | $f(x)=\frac{1}{x} \quad \rightarrow \quad \frac{df}{dx}=-1/x^2$

$f_a(x)=ax \quad \rightarrow \quad \frac{df}{dx}=a$ | $f_c(x)=c+x \quad \rightarrow \quad \frac{df}{dx}=1$

10. 3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*)：

$$f(w,x)=\frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

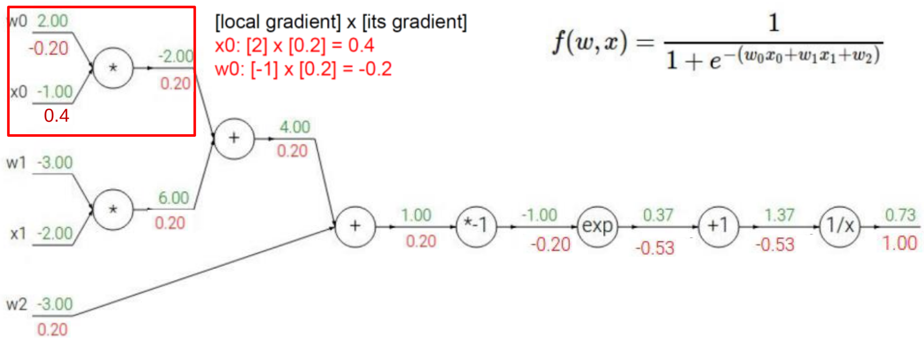


$f(x)=e^x \quad \rightarrow \quad \frac{df}{dx}=e^x$ | $f(x)=\frac{1}{x} \quad \rightarrow \quad \frac{df}{dx}=-1/x^2$

$f_a(x)=ax \quad \rightarrow \quad \frac{df}{dx}=a$ | $f_c(x)=c+x \quad \rightarrow \quad \frac{df}{dx}=1$

10. 3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*)：



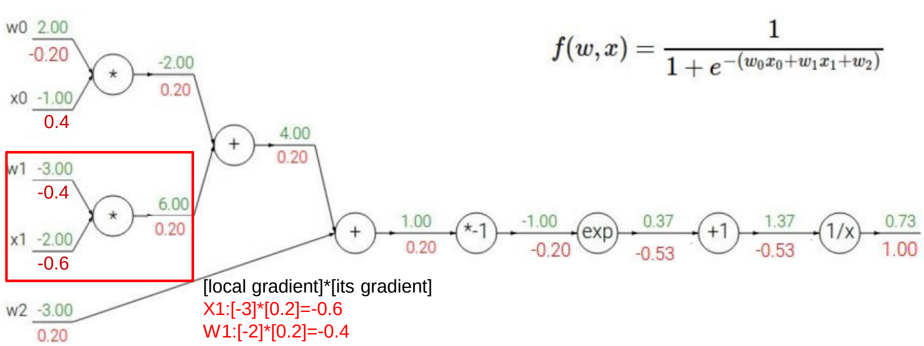
$$f(w,x)=\frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

$f(x)=e^x \quad \rightarrow \quad \frac{df}{dx}=e^x$ | $f(x)=\frac{1}{x} \quad \rightarrow \quad \frac{df}{dx}=-1/x^2$

$f_a(x)=ax \quad \rightarrow \quad \frac{df}{dx}=a$ | $f_c(x)=c+x \quad \rightarrow \quad \frac{df}{dx}=1$

10. 3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*)：



$$f(w,x)=\frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

$f(x)=e^x \quad \rightarrow \quad \frac{df}{dx}=e^x$ | $f(x)=\frac{1}{x} \quad \rightarrow \quad \frac{df}{dx}=-1/x^2$

$f_a(x)=ax \quad \rightarrow \quad \frac{df}{dx}=a$ | $f_c(x)=c+x \quad \rightarrow \quad \frac{df}{dx}=1$

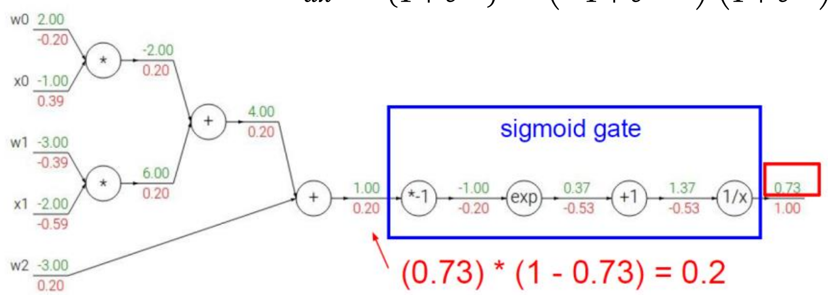
10.3 神经网络学习

用计算图求解梯度反传(*Computational Graph for Gradient Backpropagation*):

$$f(w,x)=\frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

$$\theta(x)=\frac{1}{1+e^{-x}}$$ sigmoid function

$$\frac{d\theta(x)}{dx}=\frac{e^{-x}}{(1+e^{-x})^2}=\left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right)=(1-\theta(x))\theta(x)$$



第十讲 神经网络到深度学习(*From Neural Networks to Deep Learning*)

10.1 神经网络动机（*Motivation of Neural Networks*）

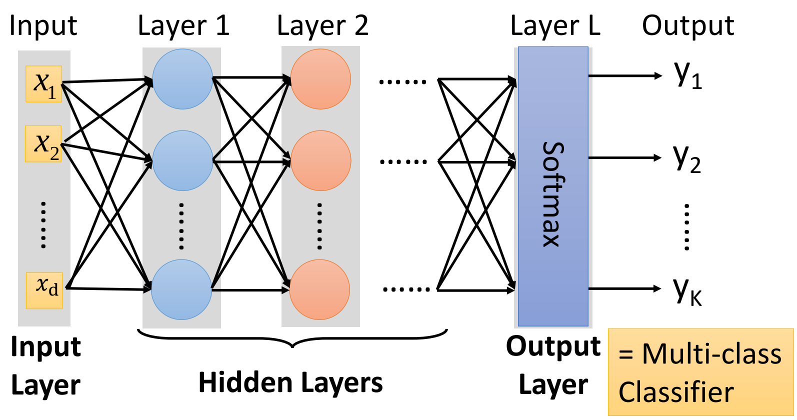
10.2 神经网络模型(*Neural Network Hypothesis*)

10.3 神经网络学习（*Neural Network Learning*）

10.4 深度神经网络（*Deep Neural Networks*）

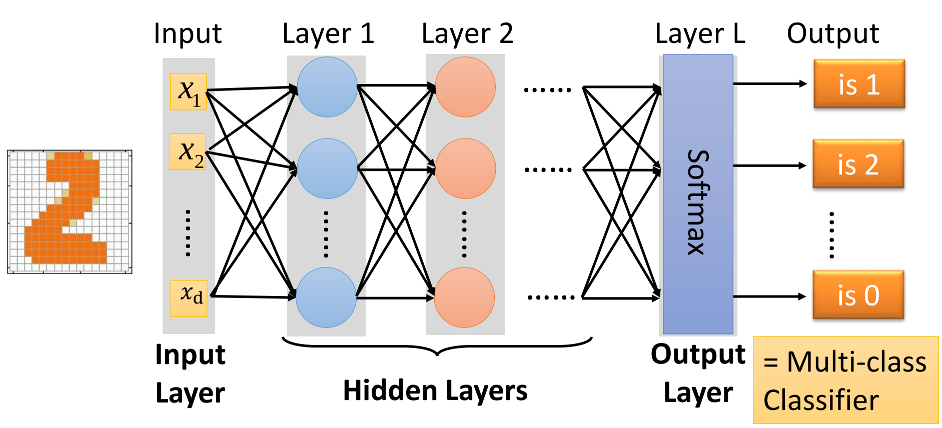
10.4 深度神经网络

神经网络(*Neural Networks*):



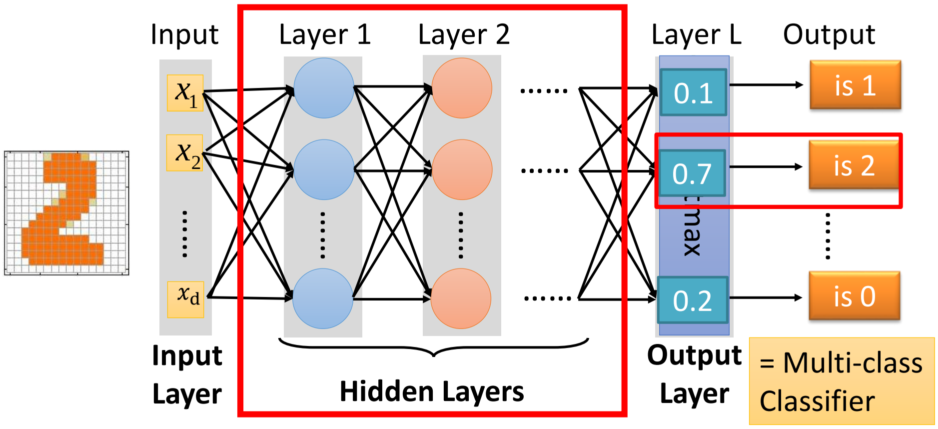
10.4 深度神经网络

神经网络(*Neural Networks*):



10. 4 深度神经网络

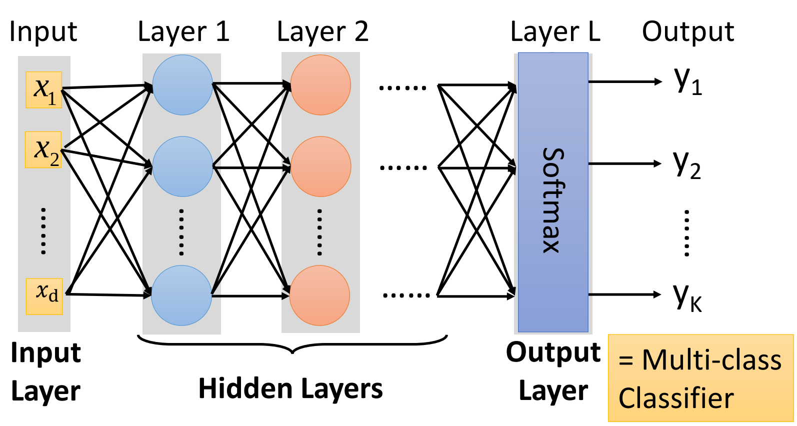
神经网络(*Neural Networks*):



10. 4 深度神经网络

神经网络(*Neural Networks*):

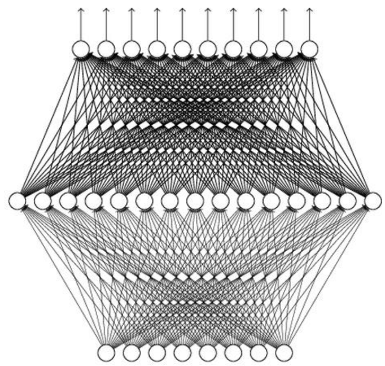
Deep = Many hidden layers



10. 4 深度神经网络

为什么要用深度网络?

Layer X Size	Word Error Rate (%)
1 X 2k	24.2
2 X 2k	20.4
3 X 2k	18.4
4 X 2k	17.8
5 X 2k	17.2
7 X 2k	17.1



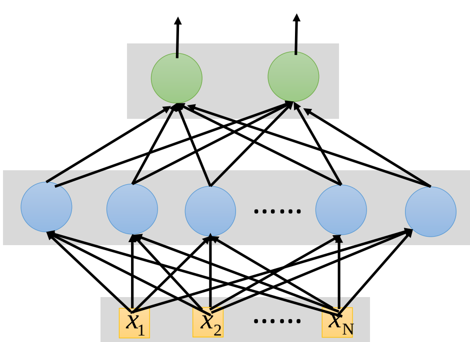
$$f : R^N \rightarrow R^M$$

只要有足够多神经元，一个隐含层的网络结构也能拟合出任意复杂的连续曲面

按照学习理论，参数多、性能好

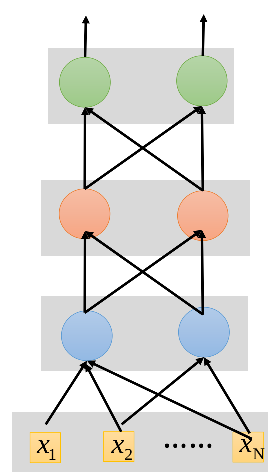
10. 4 深度神经网络

浅层网络好还是深层网络好?



Shallow

同等参数量条件下比较性能



Deep

10.4 深度神经网络

为什么要用深度网络?

Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

10.4 深度神经网络

Neural Networks practitioner

10.4 深度神经网络

深度学习存在的挑战和关键技术

- 如何设计或确定网络结构

10.4 深度神经网络

深度学习存在的挑战和关键技术

- 如何设计或确定网络结构

10.4 深度神经网络

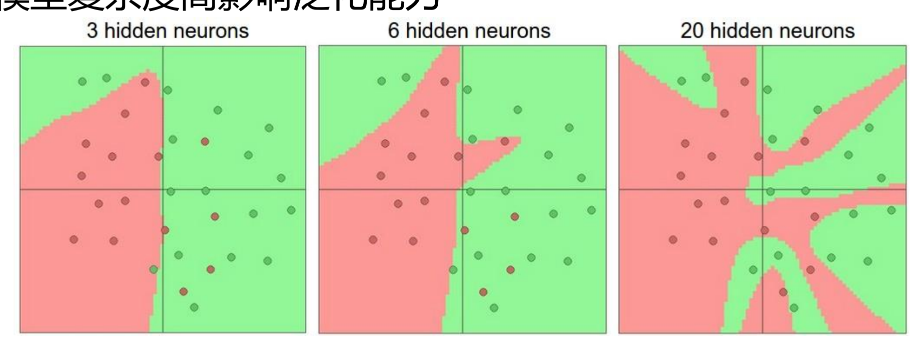
深度学习存在的挑战和关键技术

- 如何设计或确定网络结构
  - “领域知识”有助于选择网络结构，如计算机视觉中常常用卷积神经网络

10.4 深度神经网络

深度学习存在的挑战和关键技术

- 如何设计或确定网络结构
  - “领域知识”有助于选择网络结构，如计算机视觉中常常用卷积神经网络
- 模型复杂度高影响泛化能力



10.4 深度神经网络

深度学习存在的挑战和关键技术

- 如何设计或确定网络结构
  - "领域知识"有助于选择网络结构，如计算机视觉中常常用卷积神经网络
- 模型复杂度高导致过拟合
  - "大数据"有助于增加训练集，提升泛化能力
  - "正则化"—如Dropout等手段，增强对噪声的容忍度
- 模型复杂难以找到最优解
  - 仔细选择初始值，避免落入局部极值，包括预训练等
  - 防止梯度消失，采用合适的激活函数、设计更好的网络结构
- 计算复杂度高
  - 平行化、批量化、GPU等

第十讲 神经网络到深度学习(*From Neural Networks to Deep Learning*)

10.1 神经网络动机 (Motivation of Neural Networks)

多层网络结构具有更强的分类能力

10.2 神经网络模型参数空间(Neural Network Hypothesis)

基于线性模型构建多层结构

10.3 神经网络学习 (Neural Network Learning)

通过反向传播法有效实现梯度计算

10.4 深度神经网络 (Deep Neural Networks)

介绍了其优越性和面临的挑战