四川大學
SICHUAN UNIVERSITY

机器学习-第七章 深度神经网络简介

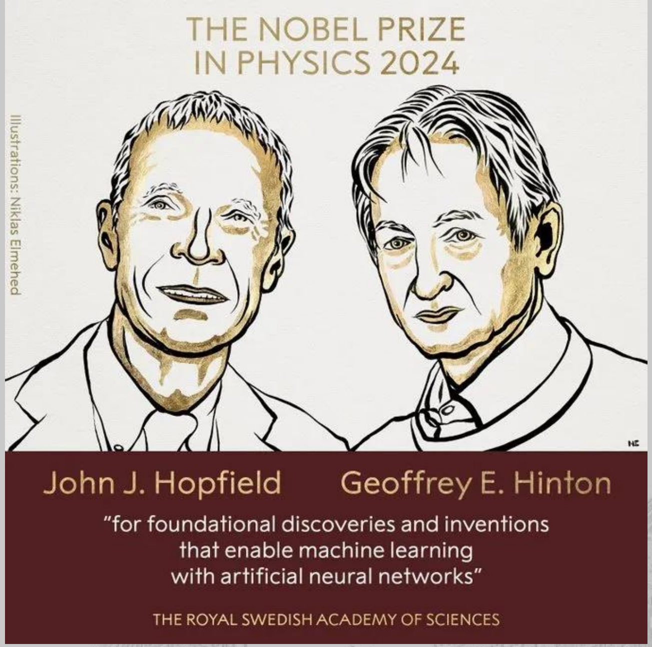
教师：胡俊杰 副教授

邮箱：hujunjie@scu.edu.cn

人工智能

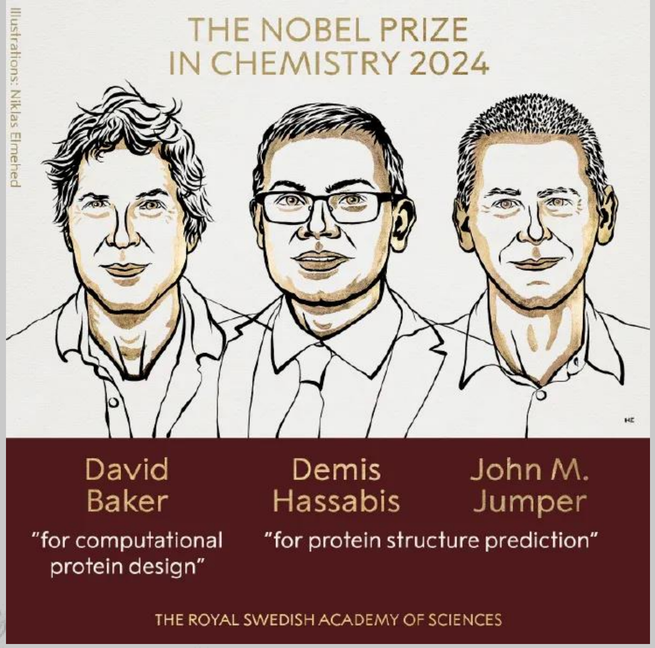
2024年，人工智能占据诺贝尔奖半壁江山

诺贝尔物理学奖



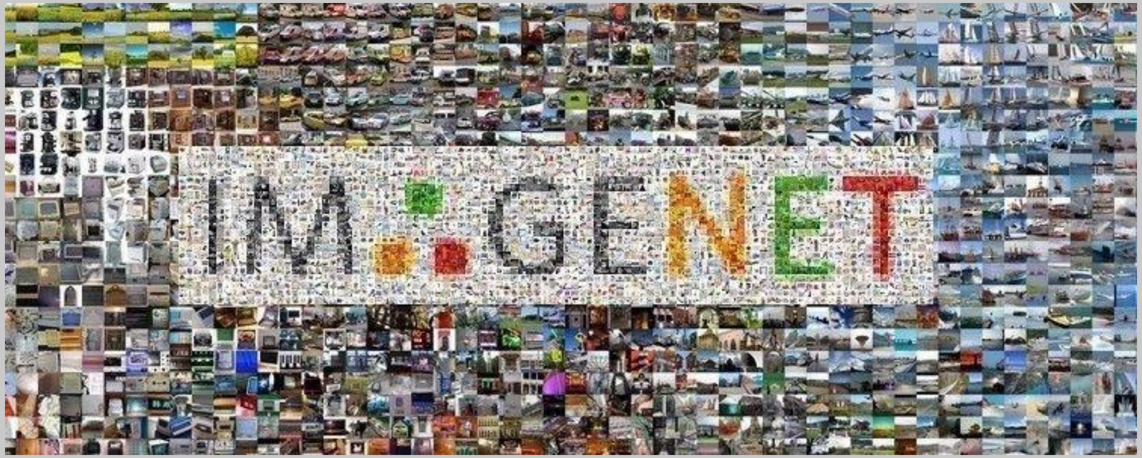
"人工神经网络机器学习的基础发现和发明"

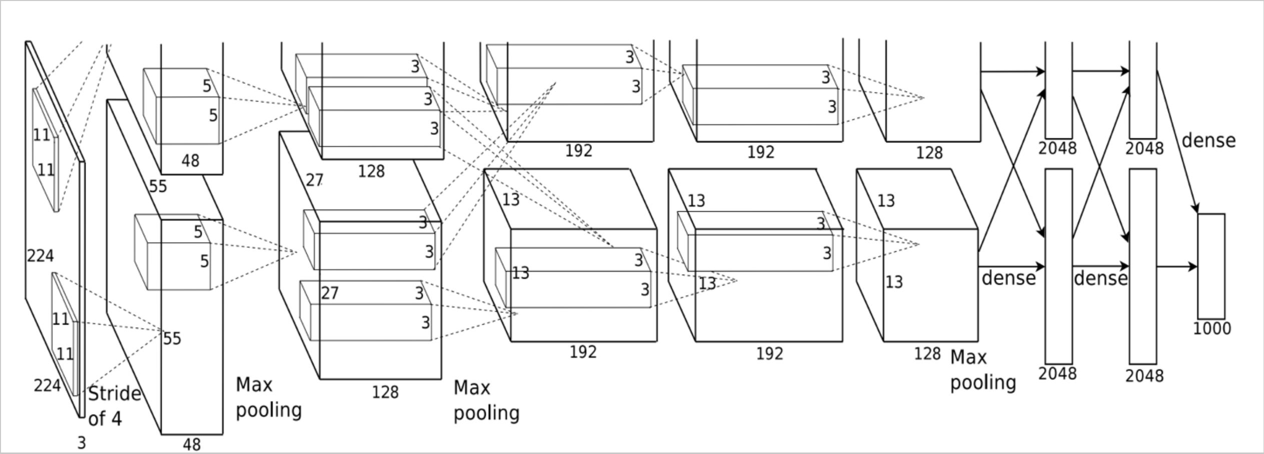
诺贝尔化学奖



"蛋白质结构预测"

人工智能





- ImageNet包含超过100万张图像数据
- 划分为1000个类别

2012年AlexNet深度神经网络模型夺得ImageNet分类数据集冠军

人工智能

AlphaGo：2016年3月战胜世界围棋冠军李世石

AlphaGo Zero:2017年12月推出，无需人类棋谱，完成自我博弈

人工智能

2022年11月ChatGPT上线，引发大语言模型人工智能全球热潮

■ OpenAI，闭源

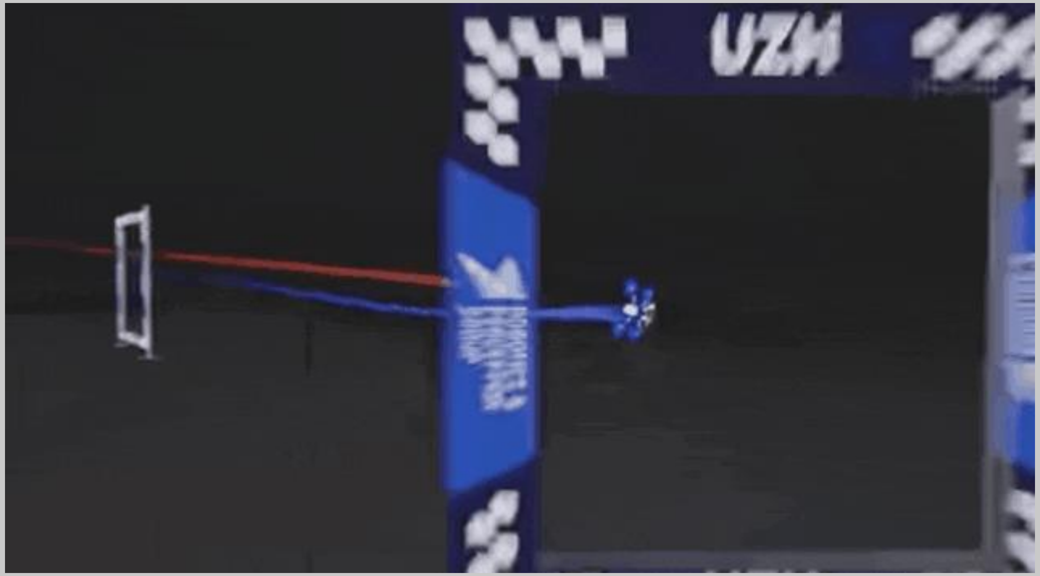
2024年12月DeepSeek V3上线，为全球人工智能贡献中国力量

■ 幻化量方，开源

人工智能

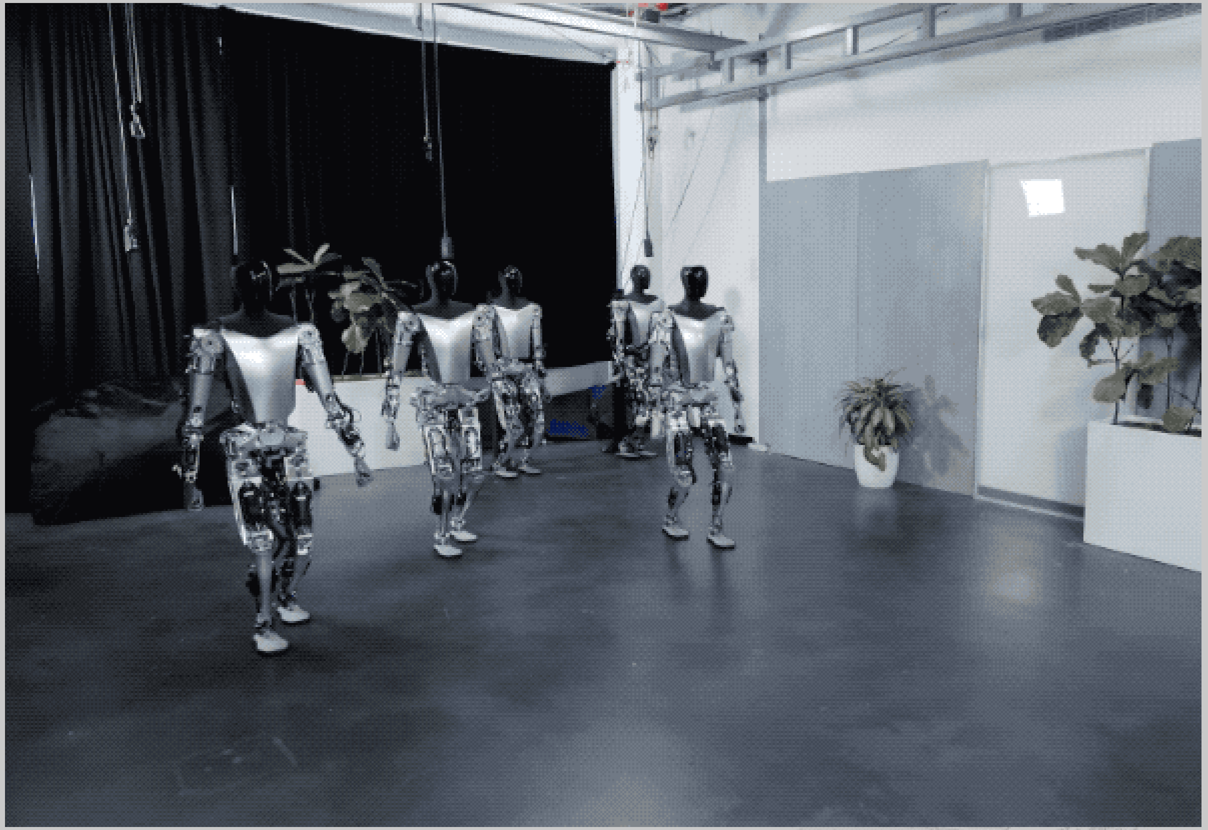
自动驾驶
人工智能端到端
自动驾驶决策

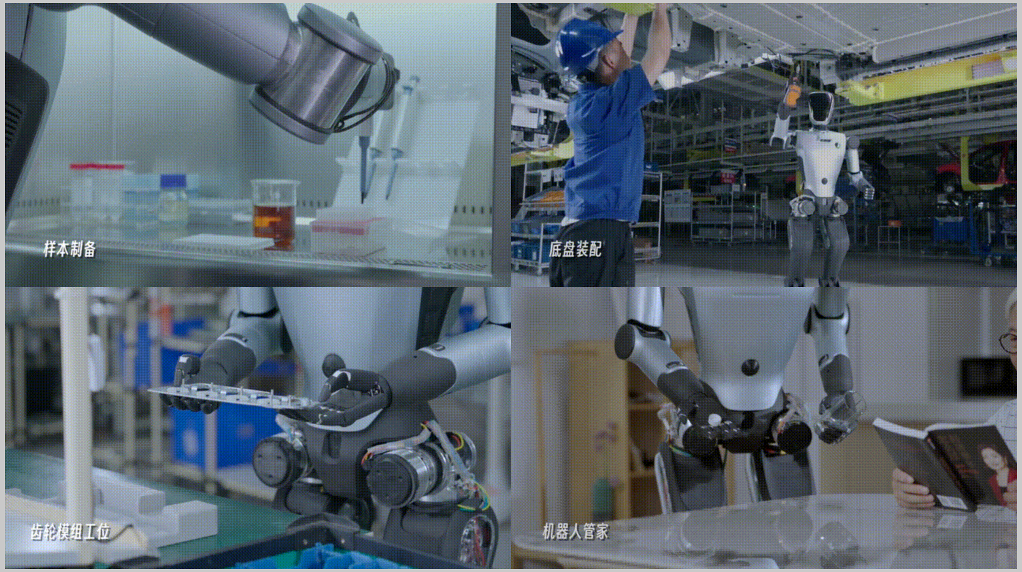
人工智能



无人机极限竞速
人工智能击败人类顶尖选手

人工智能





智能机器人
代替人类完成一系列工作

人工智能属于国家战略

中华人民共和国中央人民政府
www.gov.cn

国务院关于印发
新一代人工智能发展规划的通知
国发〔2017〕35号

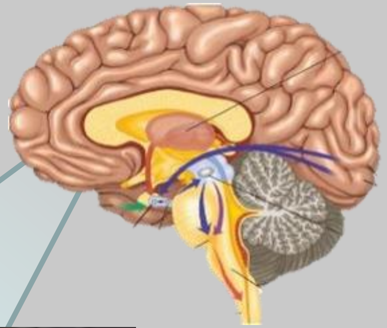
2017年10月，党的十九大报告中指出要加快人工智能发展

2017年7月，国务院印发《新一代人工智能发展规划》，从国家战略层面推动我国人工智能发展，抢占人工智能全球制高点

本章目录

1. 深度神经网络概述

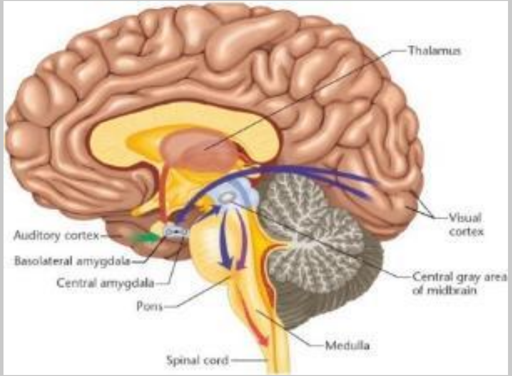
深度神经网络
是一种模拟大脑生物神经网络信息处理机制的
计算模型



1. 深度神经网络概述

生物神经网络

- 由神经元连接构成的网络
- 神经元是一种特殊的细胞，主要包含三个部分：树突、胞体和轴突
- 神经科学研究表明：神经元是信息的载体，是智能产生的基础



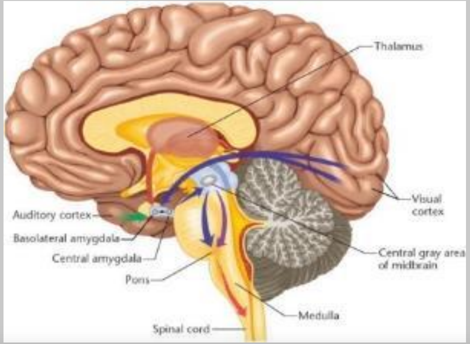
人脑约含有10^{11}（一千亿）个神经元，相当于银河系里恒星的数量。

神经元

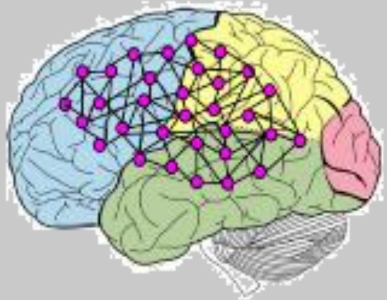
1. 深度神经网络概述

所有的智能
行为都产生
于神经网络

生物神经网络



神经网络=神经元+连接



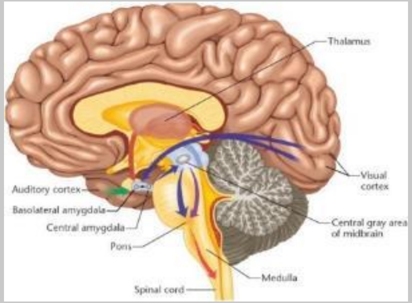
类比计算机与互联网

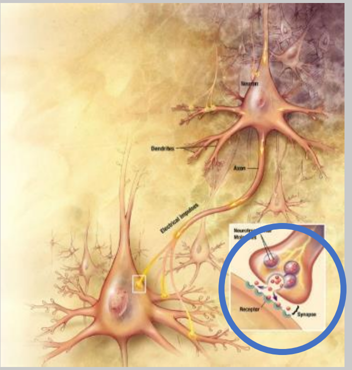
1. 深度神经网络概述

生物神经网络

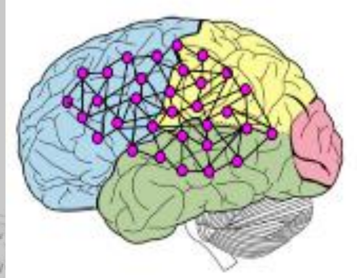
主要概念

- **神经元**：神经元是信息的载体，是智能产生的基础
- **突触连接**：知识存储在突触连接上
- **神经网络**：神经网络=神经元+连接；所有的智能行为都产生于神经网络
- **学习**：知识是通过学习获取的。学习表现为新连接的建立和已有连接的修改





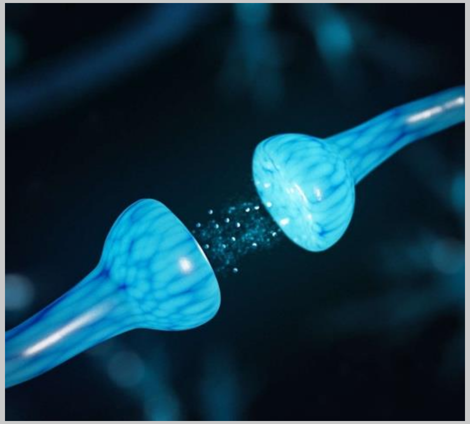
突触



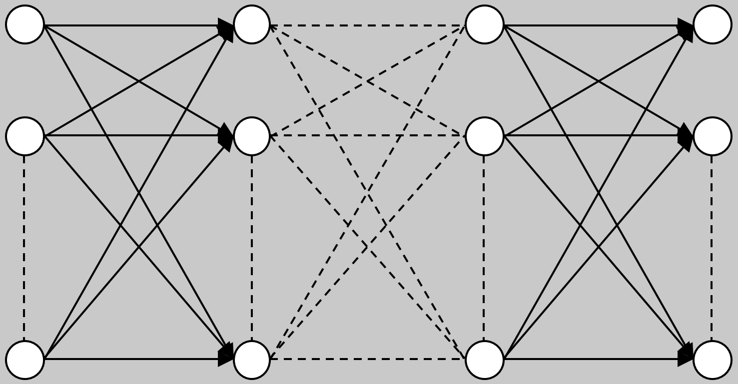
1. 深度神经网络概述

生物神经网络		人工神经网络
神经元		神经元模型
突触连接	抽象 → 建立可计算的数学模型	连接权
神经网络		神经网络模型
学习		学习算法

1. 深度神经网络概述



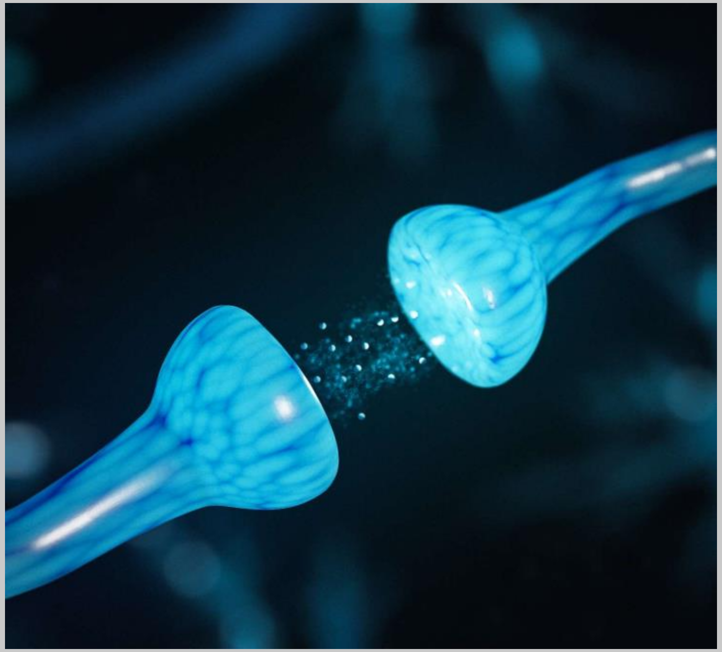
抽象

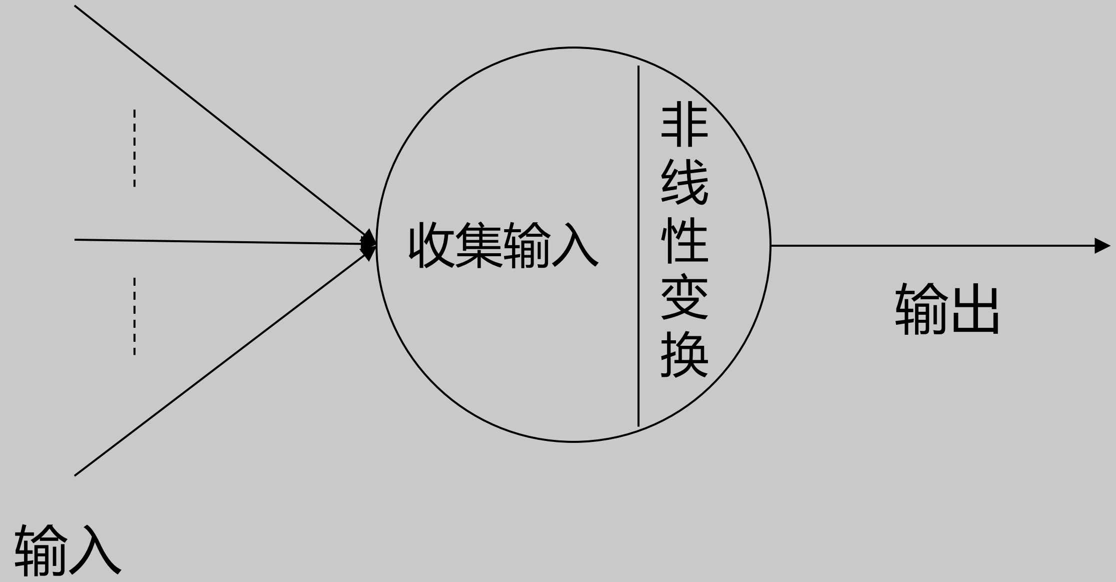


- 人脑中包含上百亿个神经元
- 每个神经元包含上万个连接

- 可计算的神经网络模型

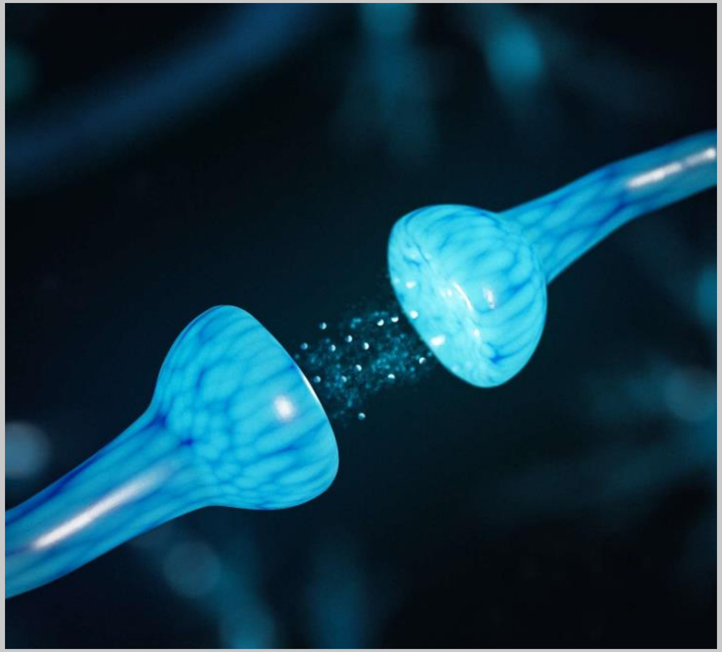
1. 深度神经网络概述



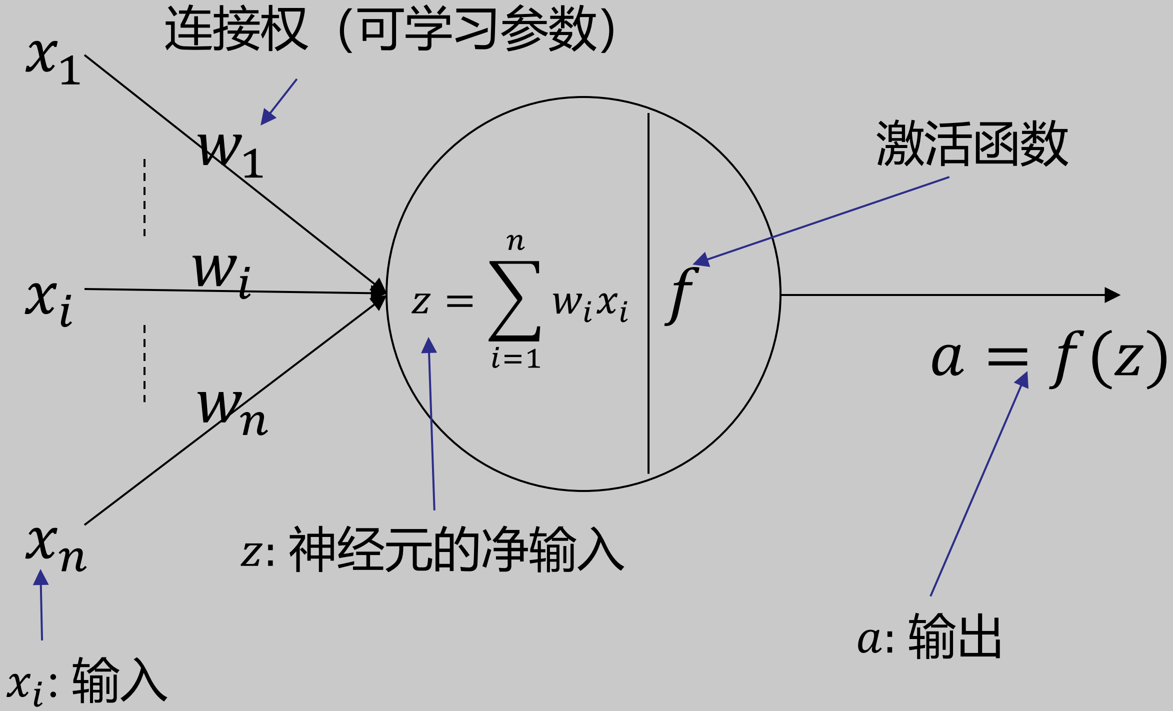


- 生物学神经元
- 可计算的神经元模型

1. 深度神经网络概述

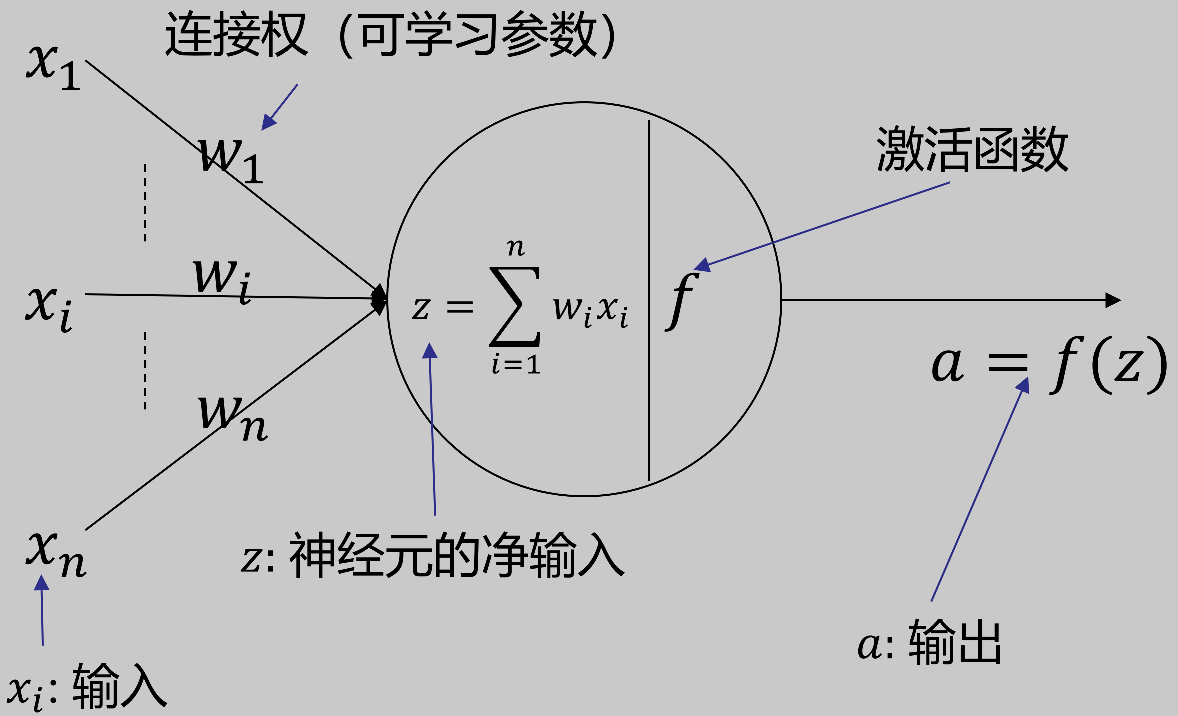


■ 生物学神经元



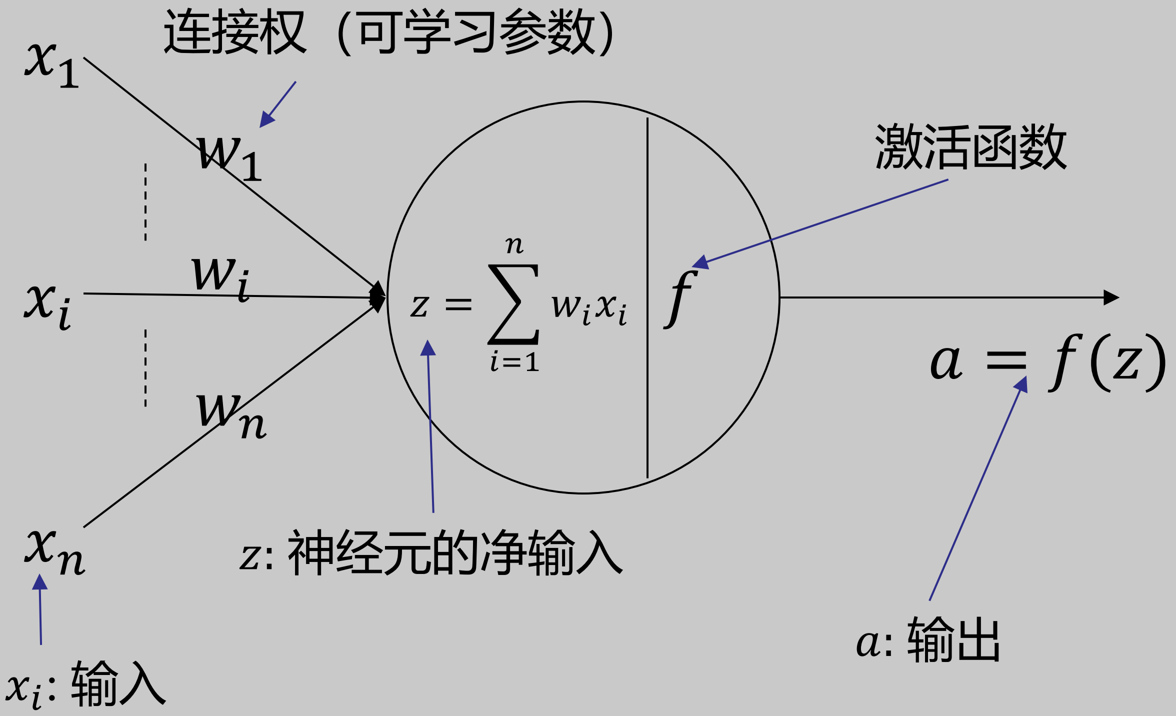
- x_i: 输入
- w_i: 连接权，可学习参数
- z: 净输入，输入的加权和
- f: 激活函数，如sigmoid
- a: 输出

1. 深度神经网络概述

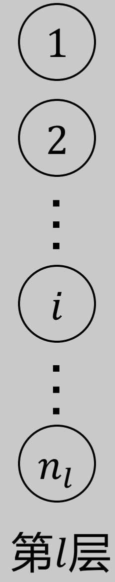


- 通过加权求和收集输入 $z = \sum_{i=1}^{n} w_i x_i$
- 作用激活函数给出输出 $a = f(z)$

1. 深度神经网络概述

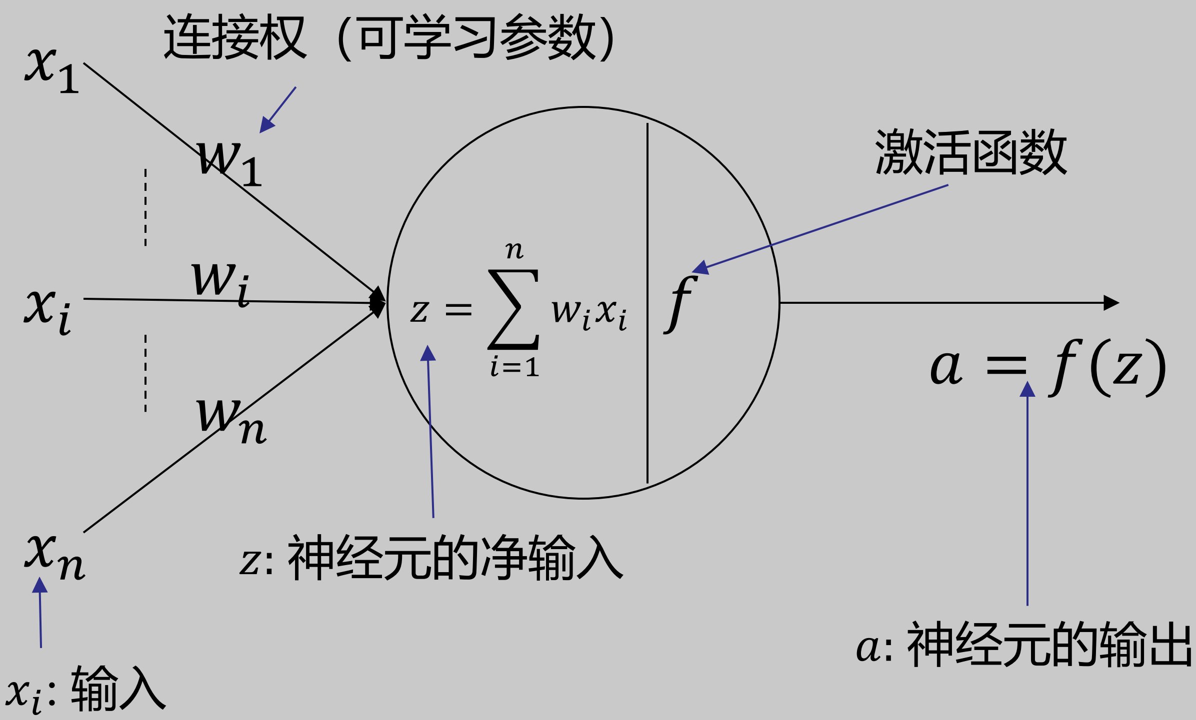


■ 多个神经元组成一层

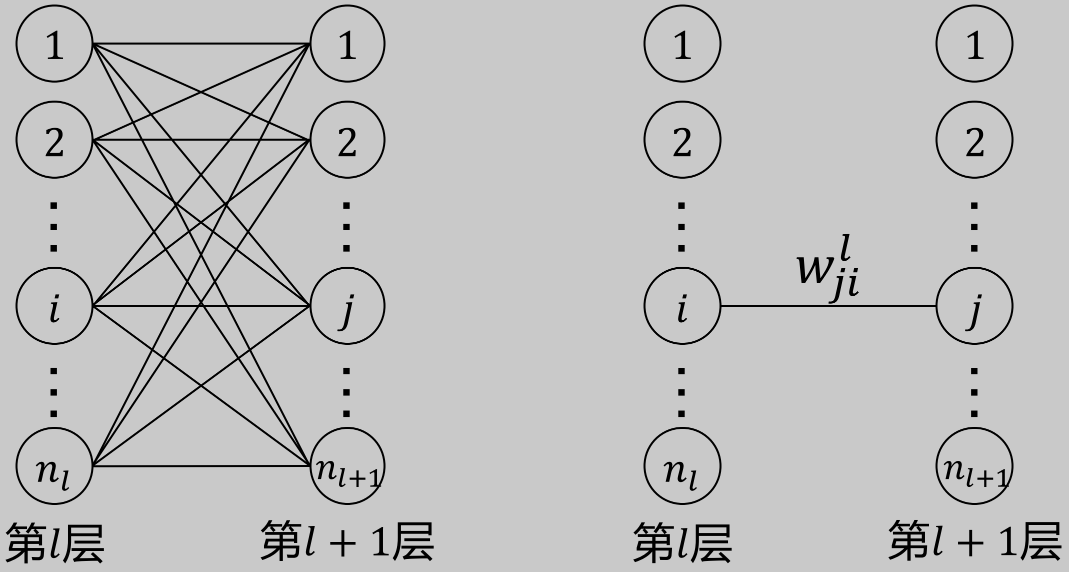


- 假设第l层
- 每层包含n_l个神经元
- 第i个神经元的净输入记为z_i^l
- 第i个神经元的输出记为a_i^l

1. 深度神经网络概述

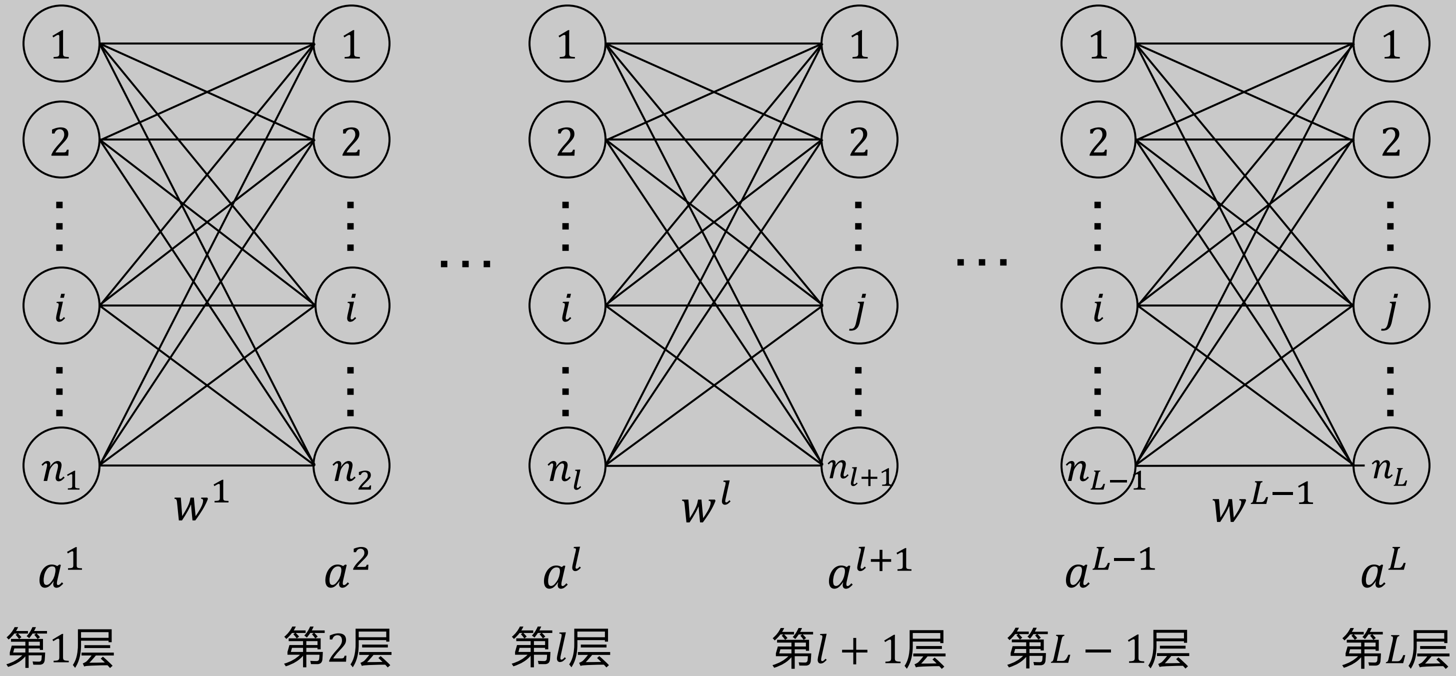


■ 两层之间构成连接权矩阵W^l



- 全连接（fully-connected）网络：两层之间的神经元均通过连接权相连
- 同层的神经元无连接
- w_{ji}^l: 第$l+1$层j神经元与第l层i神经元之间的连接权
- $W^l \in R^{n_{l+1} \times n_l}$

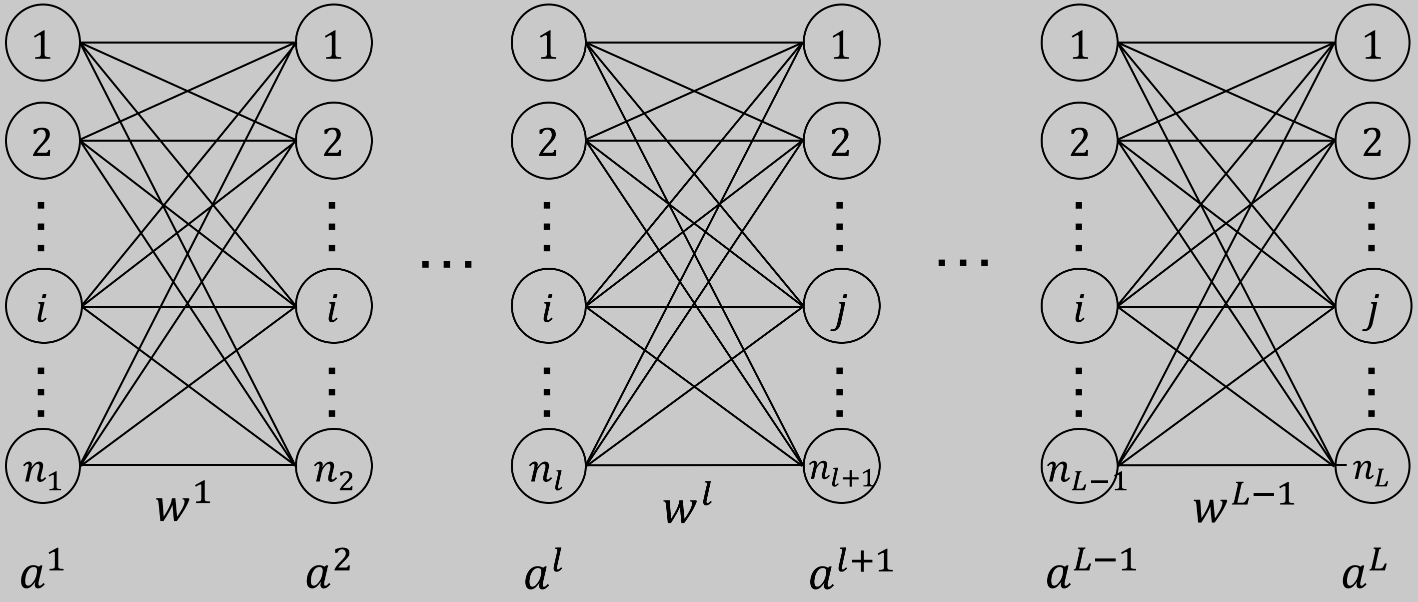
1. 深度神经网络概述



1
2
i
n_1
w^1
1
2
i
n_2
1
2
i
n_l
w^l
1
2
j
n_{l+1}
1
2
i
n_{L-1}
w^{L-1}
1
2
j
n_L
a^1
a^2
a^l
a^{l+1}
a^{L-1}
a^L
第1层
第2层
第l层
第$l+1$层
第$L-1$层
第L层

本章目录

2. 前向计算



前向计算

- 前向计算-标量形式

$$
\begin{cases}
z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l \\
a_i^{l+1} = f\left(z_i^{l+1}\right)
\end{cases}
$$

$$a_i^{l+1} = f\left(\sum_{j=1}^{n_l} w_{ij}^l a_j^l\right)$$

- 前向计算-向量形式

$$
\begin{cases}
z^{l+1} = w^l a^l \\
a^{l+1} = f\left(z^{l+1}\right)
\end{cases}
$$

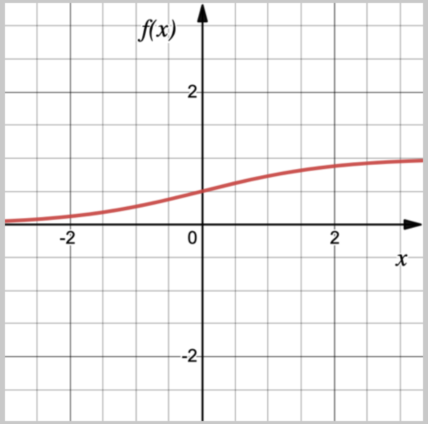
$$a^{l+1} = f\left(w^l a^l\right)$$

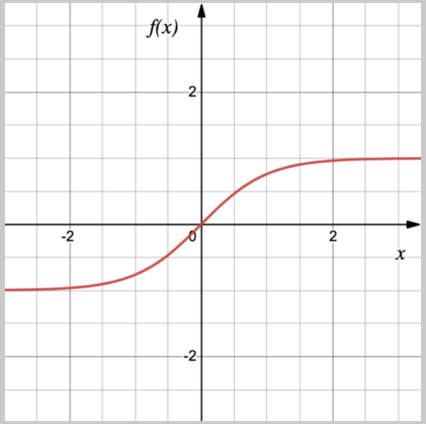
2. 前向计算

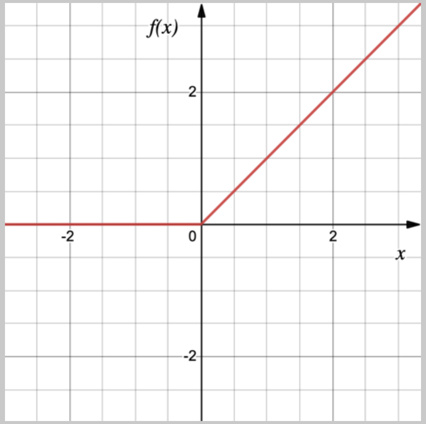
- 激活函数f

$$a_i^{l+1} = f(\sum_{j=1}^{n_l} w_{ij}^l a_j^l)$$

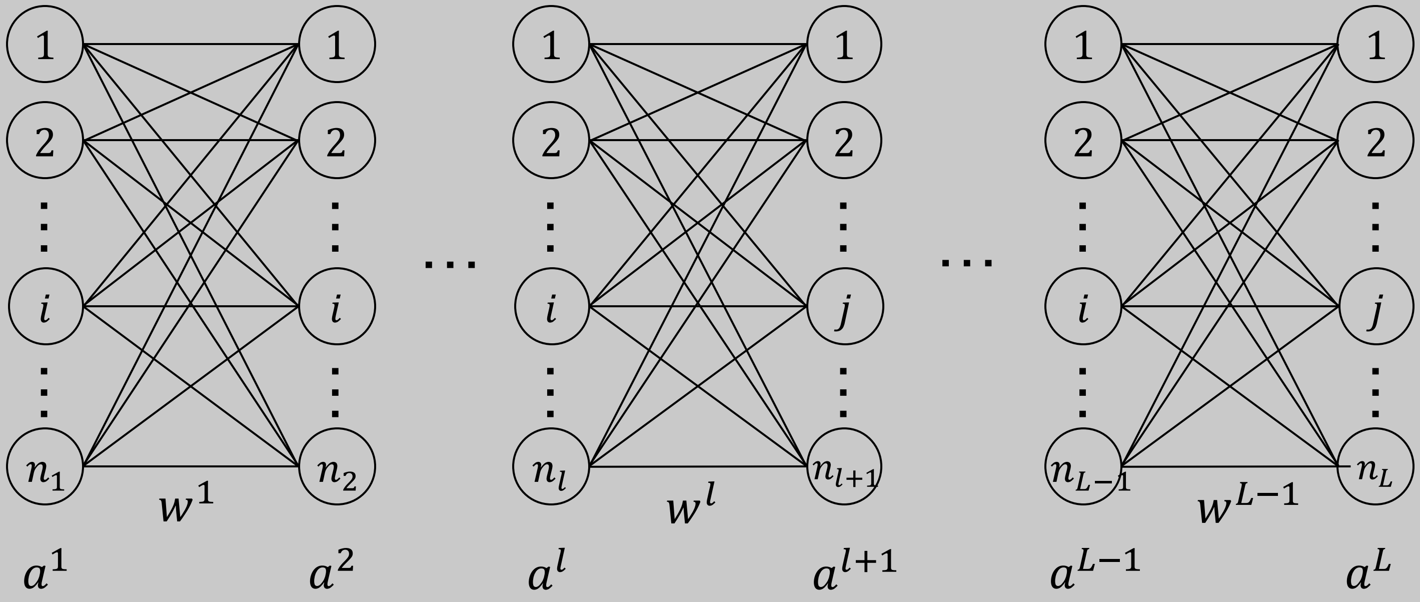
Sigmoid	Tanh	ReLU
$f(x) = \dfrac{1}{1+e^{-x}}$	$f(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$	$f(x) = \max(0, x)$







2. 前向计算



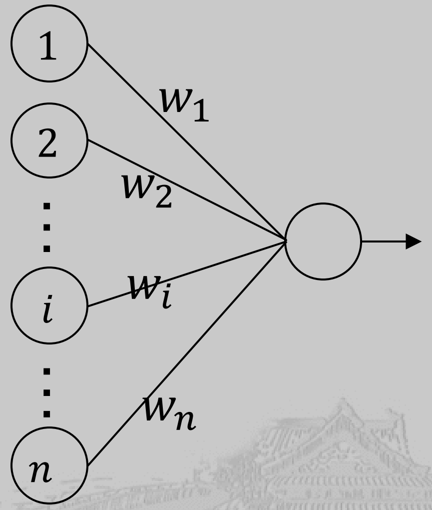
- 两层之间的计算形式： $a^{l+1} = f\left(w^l a^l\right)$
- 前向计算的一般表达形式： $a^L = f\left(w^{L-1} f\left(w^{L-2} f\left(w^{L-3} \cdots f\left(w^1 a^1\right)\right)\right)\right)$
- 其中a^1代表输入，a^L代表输出

2. 前向计算

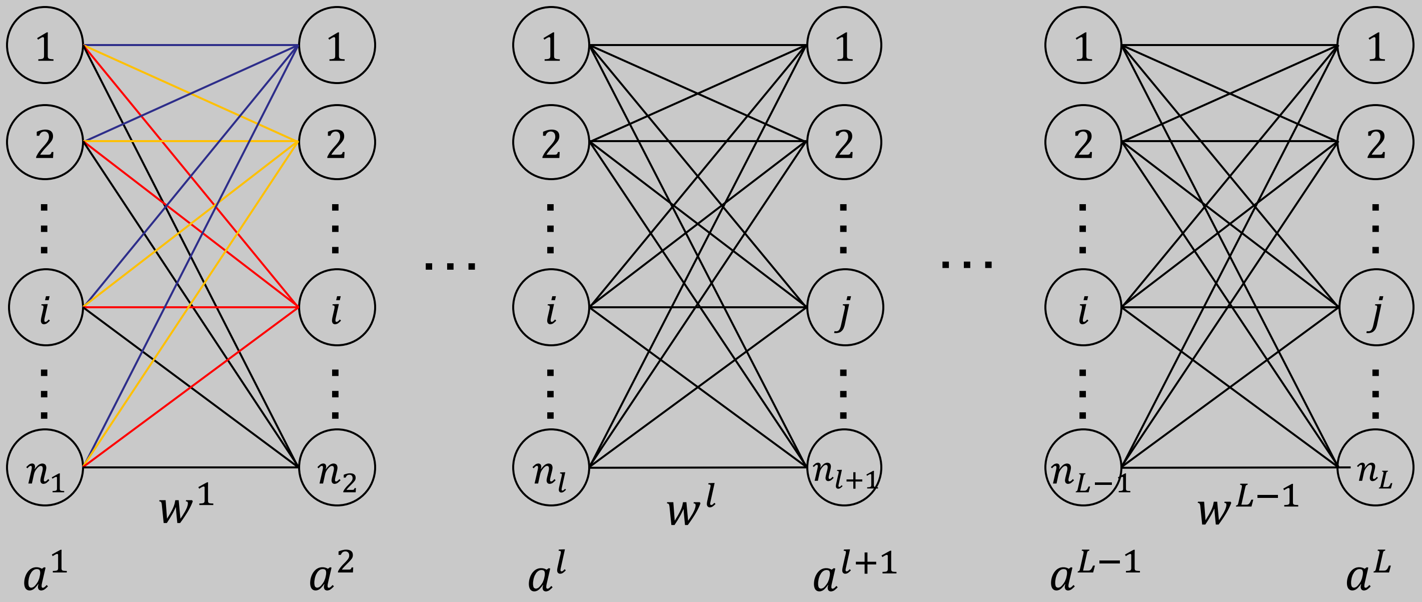
- 逻辑回归模型

$$\begin{cases} z = w^T x \\ g(z) = \dfrac{1}{1+e^{-z}} \end{cases}$$

- 神经网络模型



- 深度神经网络模型

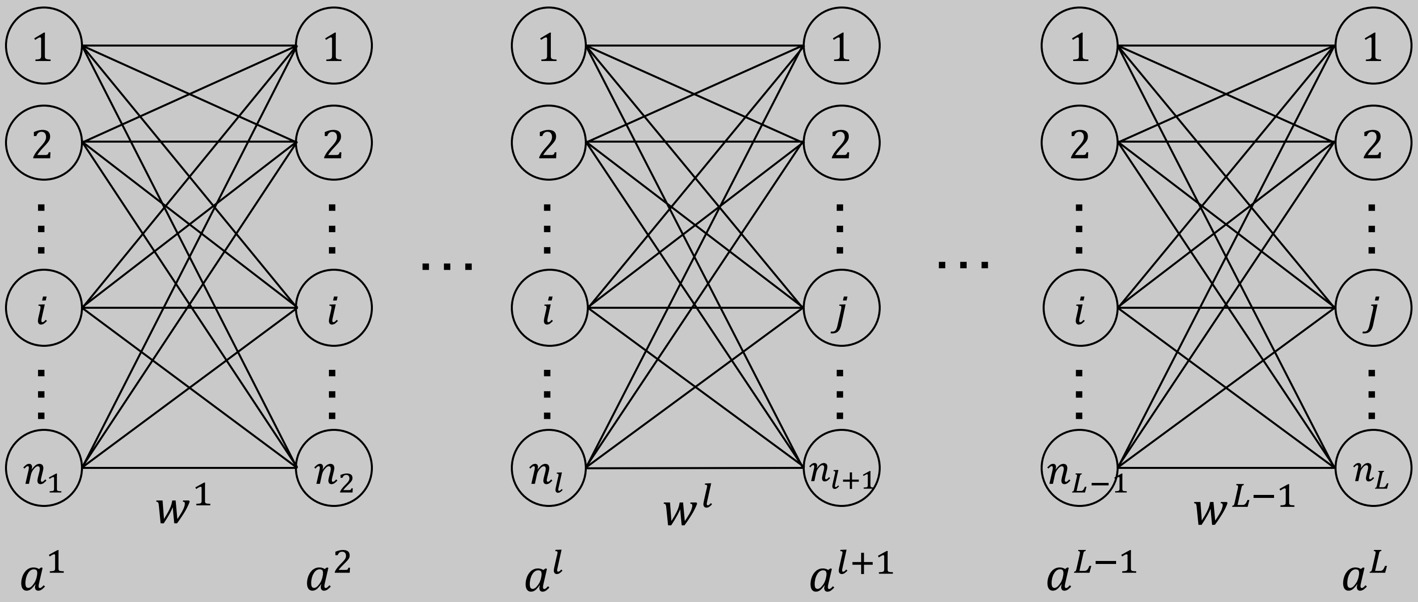


$$a^L = f\left(w^{L-1} f\left(w^{L-2} f\left(w^{L-3} \cdots f\left(w^1 a^1\right)\right)\right)\right)$$

- 更强的非线性拟合能力

本章目录

3. 反向传播



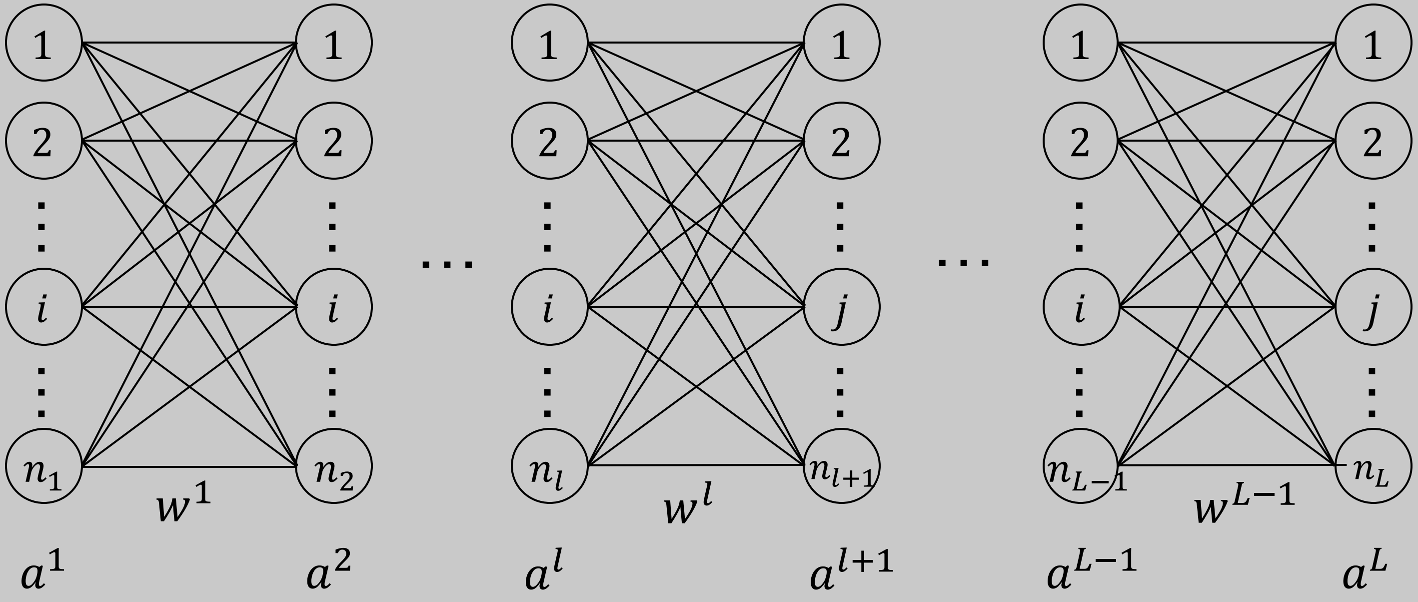
- a^1: 输入，a^L: 输出

- 考虑训练数据样本(x, y)，其中a^1即对应x，二者维度相等
- 标签y采用one-hot编码，假设有y共包含10个类别，则各类别的标签表达形式如下

$$
\begin{array}{cccccccccc}
\begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\1\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\1\\0\\0\\0\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\1\\0\\0\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\\1\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\\0\\1\end{bmatrix} \\
\text{类1} & \text{类2} & \text{类3} & \text{类4} & \text{类5} & \text{类6} & \text{类7} & \text{类8} & \text{类9} & \text{类10}
\end{array}
$$

- a^L的维度与y的维度相等

3. 反向传播



$\boxed{\text{输出：} a^L} \xrightarrow{\text{距离}} \boxed{\text{标签：} y}$

$$a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ \vdots \\ y_{n_L} \end{bmatrix}$$

- 损失函数（代价函数/性能函数）J
- J度量输出a^L与标签y的距离
- J是关于$(w^1,\cdots,w^{L-1})$的函数，$J=J(w^1,\cdots,w^{L-1})$

- a^1: 输入，a^L: 输出

均方误差

（Mean Squared Error ,MSE）

$$J(w^1,\cdots,w^{L-1})=\frac{1}{2}\sum_{j=1}^{n_L}\left(a_j^L-y_j\right)^2$$

目标：最小化J

3. 反向传播

■ 随机梯度下降算法

第1步. 准备训练数据集 $D=\{(x,y)\}$

第2步. 随机初始化神经网络各层参数$(w^1,w^2,\ldots,w^{L-1})$, 设置学习率 α.

第3步. 随机选择b个样本（一个batch），计算并累积各样本对各层参数的梯度$\frac{\partial J}{\partial w_{ji}^l}$

第4步. 更新参数

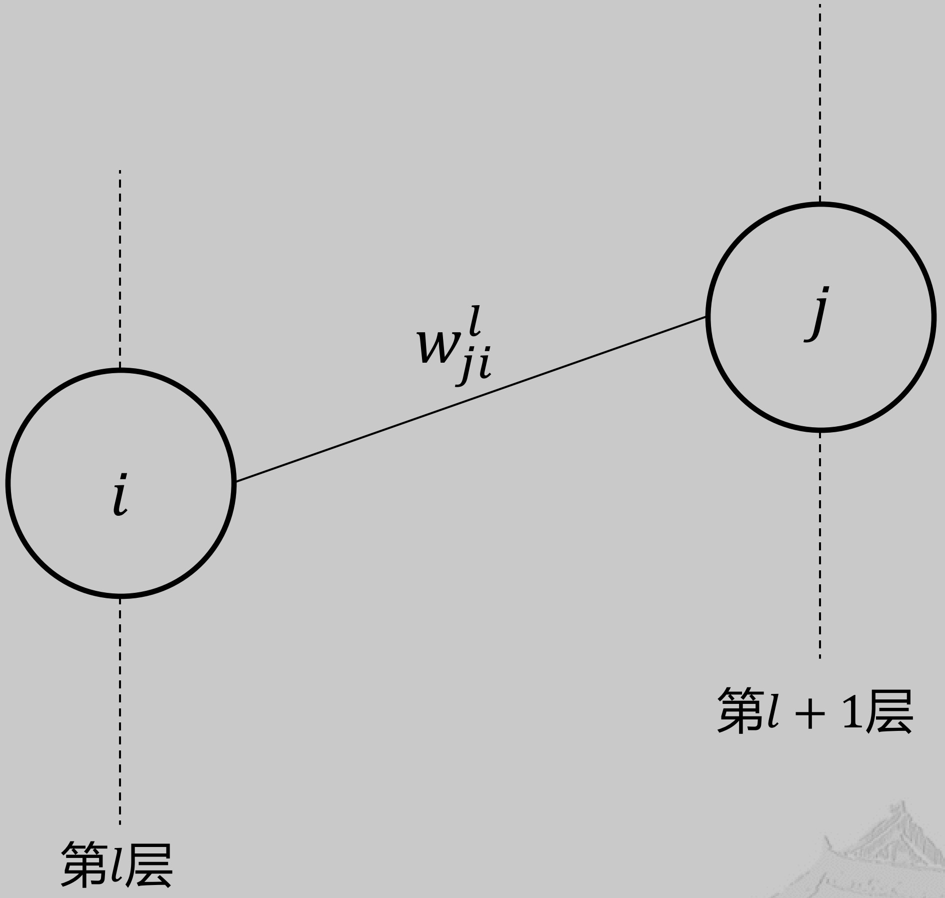
$$w_{ji}^l := w_{ji}^l - \alpha\frac{1}{b}\frac{\partial J}{\partial w_{ji}^l}$$

第5步. 继续第3步，直到模型收敛.

$$J(w^1,\cdots,w^{L-1}) = \frac{1}{2}\sum_{j=1}^{n_L}\left(a_j^L - y_j\right)^2$$

反向传播（backpropagation）算法求$\frac{\partial J}{\partial w_{ji}^l}$

3. 反向传播



前向计算 $\begin{cases} z_j^{l+1} = \sum\limits_{i=1}^{n_l} w_{ji}^{l} a_i^{l} \\ a_j^{l+1} = f\left(z_j^{l+1}\right) \end{cases}$

- 定义敏感系数$\delta_i^l = \dfrac{\partial J}{\partial z_i^l}$
- 链式法则:

$$\frac{\partial J}{\partial w_{ji}^{l}} = \frac{\partial J}{\partial z_j^{l+1}} \frac{\partial z_j^{l+1}}{\partial w_{ji}^{l}} = \delta_j^{l+1} a_i^{l}$$

- a_i^l由前向计算可求得，如何求δ_j^{l+1}，即各层神经元的敏感系数

3. 反向传播

- 最后一层神经元的敏感系数δ_i^L

$$J = \frac{1}{2}\sum_{j=1}^{n_L}\left(a_j^L - y_j\right)^2 \qquad a_i^l = f\left(z_i^l\right)$$

$$\delta_i^L = \frac{\partial J}{z_i^L} = ?$$

3. 反向传播

- 最后一层神经元的敏感系数δ_i^L

$$J=\frac{1}{2}\sum_{j=1}^{n_L}\left(a_j^L-y_j\right)^2 \qquad a_i^l=f\left(z_i^l\right)$$

$$\delta_i^L=\frac{\partial J}{z_i^L}=\left(a_i^L-y_i\right)\frac{\partial a_i^L}{\partial z_i^L}=\left(a_i^L-y_i^L\right)\dot{f}\left(z_i^L\right)$$

3. 反向传播

- 隐藏层神经元的敏感系数δ_i^l

$$z_j^{l+1}=\sum_{i=1}^{n_l} w_{ji}^l a_i^l=\sum_{i=1}^{n_l} w_{ji}^l f\left(z_i^l\right)$$

$$\delta_i^l=\frac{\partial J}{\partial z_i^l}=\sum_{j=1}^{n_{l+1}} \frac{\partial J}{\partial z_j^{l+1}} \frac{\partial z_j^{l+1}}{\partial z_i^l}=?$$

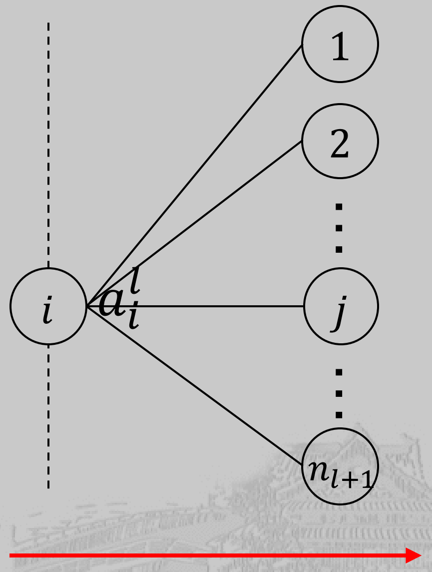
链式法则

3. 反向传播

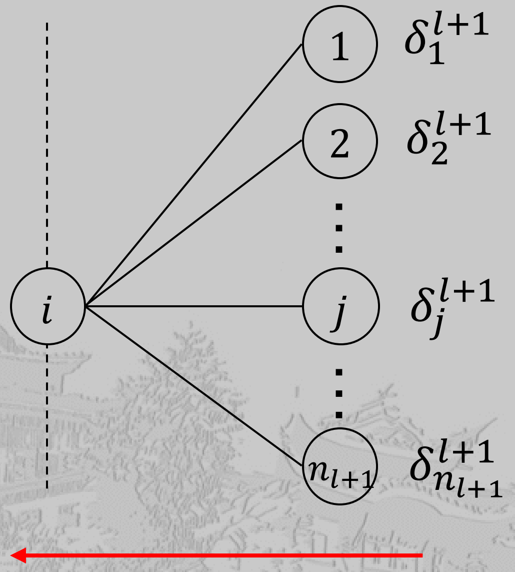
- 隐藏层神经元的敏感系数δ_i^l　　$z_j^{l+1}=\sum_{i=1}^{n_l} w_{ji}^l a_i^l=\sum_{i=1}^{n_l} w_{ji}^l f(z_i^l)$

$$\delta_i^l=\frac{\partial J}{\partial z_i^l}=\sum_{j=1}^{n_{l+1}}\frac{\partial J}{\partial z_j^{l+1}}\frac{\partial z_j^{l+1}}{\partial z_i^l}=\sum_{j=1}^{n_{l+1}}\delta_j^{l+1}\cdot\frac{\partial z_j^{l+1}}{\partial z_i^l}=\sum_{j=1}^{n_{l+1}}\delta_j^{l+1}w_{ji}^l\,\dot f(z_i^l)=\dot f(z_i^l)\left(\sum_{j=1}^{n_{l+1}}\delta_j^{l+1}w_{ji}^l\right)$$

链式法则



前向计算



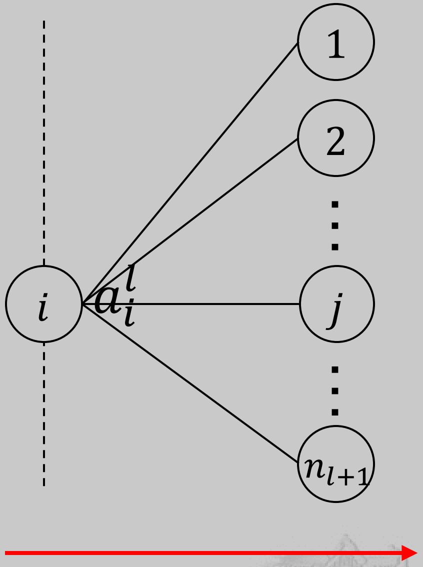
反向传播

3. 反向传播

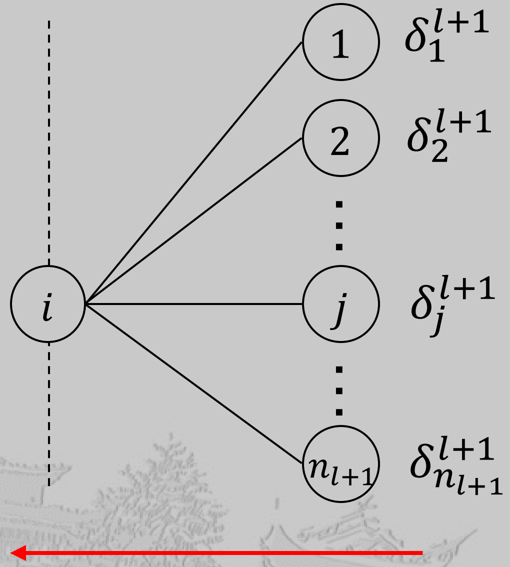
■ 前向计算与反向传播对比

$$a_j^{l+1} = f\left(\sum_{i=1}^{n_l} w_{ji}^l a_i^l\right)$$

$$\delta_i^l = \dot{f}(z_i^l)\left(\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} w_{ji}^l\right)$$



前向计算



反向传播

3. 反向传播

总结

- 反向传播是一种用于计算$\frac{\partial J}{\partial w_{ji}^{l}}$的算法

- δ_j^{l+1}与$\frac{\partial J}{\partial w_{ji}^{l}}$的关系：$\frac{\partial J}{\partial w_{ji}^{l}} = \frac{\partial J}{\partial z_j^{l+1}} \frac{\partial z_j^{l+1}}{\partial w_{ji}^{l}} = \delta_j^{l+1} a_i^{l}$

通过计算δ_j^{l+1}从而得到$\frac{\partial J}{\partial w_{ji}^{l}}$

- δ_j^{l+1}与δ_i^{l}的关系：$\delta_i^{l} = \dot{f}(z_i^{l}) \left(\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} w_{ji}^{l} \right)$

δ_j^{l+1}的反向传播

3. 反向传播

■ 标量形式

$$\frac{\partial J}{\partial w_{ji}^{l}} = \frac{\partial J}{\partial z_j^{l+1}} \frac{\partial z_j^{l+1}}{\partial w_{ji}^{l}} = \delta_j^{l+1} a_i^l$$

$$\delta_i^l = \dot{f}(z_i^l)\left(\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} w_{ji}^l\right)$$

■ 向量形式

- $\frac{\partial J}{\partial w^l} \in R^{n_{l+1} \times n_l}, a^l \in R^{n_l \times 1}, \delta^l \in R^{n_l \times 1}$

请写出左侧两个公式的向量形式

3. 反向传播

标量形式

$$\frac{\partial J}{\partial w_{ji}^{l}} = \frac{\partial J}{\partial z_j^{l+1}} \frac{\partial z_j^{l+1}}{\partial w_{ji}^{l}} = \delta_j^{l+1} a_i^l$$

$$\delta_i^l = \dot{f}(z_i^l) \left(\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} w_{ji}^l \right)$$

向量形式

- $\frac{\partial J}{\partial w^l} \in R^{n_{l+1} \times n_l}, a^l \in R^{n_l \times 1}, \delta^l \in R^{n_l \times 1}$

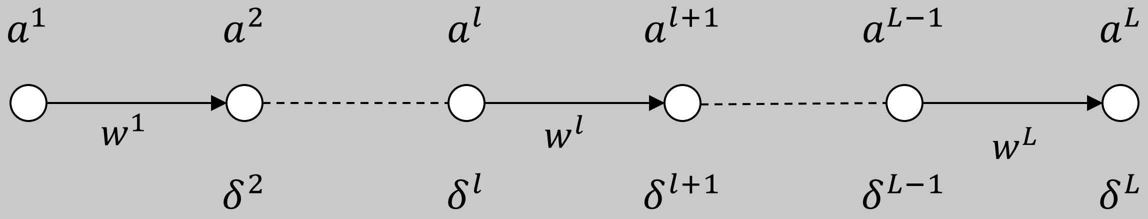
$$\frac{\partial J}{\partial w^l} = \delta^{l+1} (a^l)^T$$

$$\delta^l = \dot{f}(z^l) \circ (w^l)^T \delta^{l+1}$$

$\circ$: element-wise product

本章目录

4. 梯度消失问题

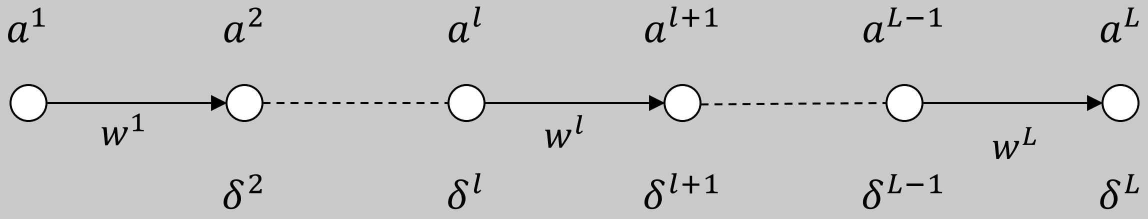


考虑每层仅有单个神经元

- 前向计算： $a^L = f\left(W^{L-1} f\left(W^{L-2} f(W^{L-3} \cdots f(W^1 a^1))\right)\right)$

- 反向传播：
$$\begin{aligned}\delta^l &= \dot{f}(z^l) w^l \delta^{l+1} \\ &= \dot{f}(z^l) w^l \dot{f}(z^{l+1}) w^{l+1} \delta^{l+2} \\ &= \dot{f}(z^l) w^l \dot{f}(z^{l+1}) w^{l+1} \cdots \dot{f}(z^{L-1}) w^{L-1} \delta^L\end{aligned}$$

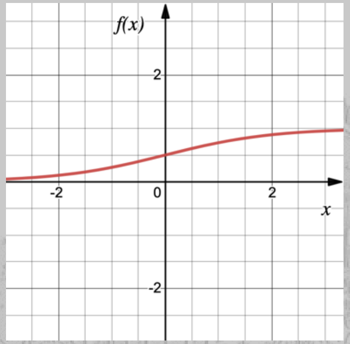
4. 梯度消失问题



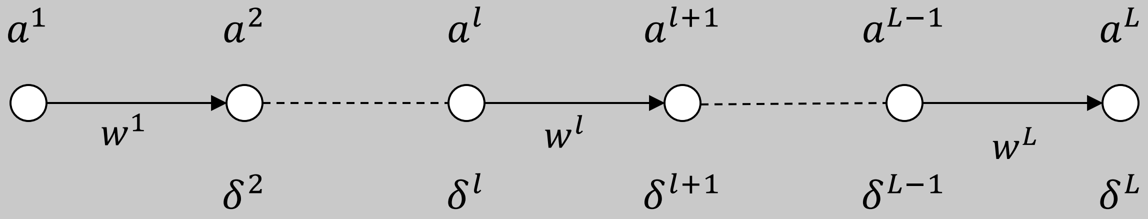
考虑每层单个神经元

- 反向传播： $\delta^l = \dot{f}(z^l)w^l\dot{f}(z^{l+1})w^{l+1}\cdots\dot{f}(z^{L-1})w^{L-1}\delta^L$ 梯度消失

- 当激活函数选择sigmoid $f(x) = \dfrac{1}{1+e^{-x}}$ $0 < \dot{f}(x) < 1$



4. 梯度消失问题



考虑每层单个神经元

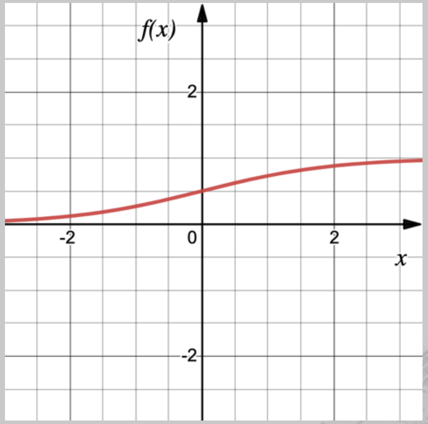
$$\delta^l = \dot{f}(z^l)w^l\dot{f}(z^{l+1})w^{l+1}\cdots\dot{f}(z^{L-1})w^{L-1}\delta^L$$

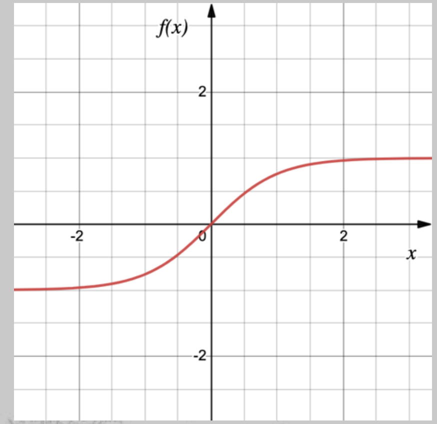
- $\dot{f}(x) = 1$，线性激活函数，得到的是线性模型，模型拟合能力弱
- $\dot{f}(x) \neq 1$，非线性激活函数，模型拟合能力强，但梯度消失/爆炸，训练困难

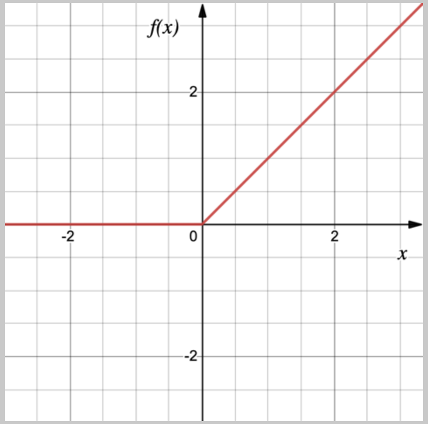
4. 梯度消失问题

■ 激活函数的角度

Sigmoid	Tanh	ReLU
$f(x) = \dfrac{1}{1+e^{-x}}$	$f(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$	$f(x) = \max(0, x)$

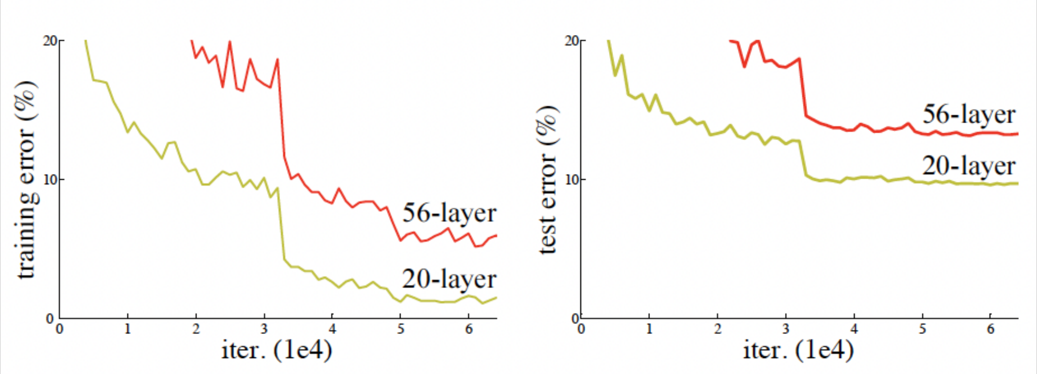




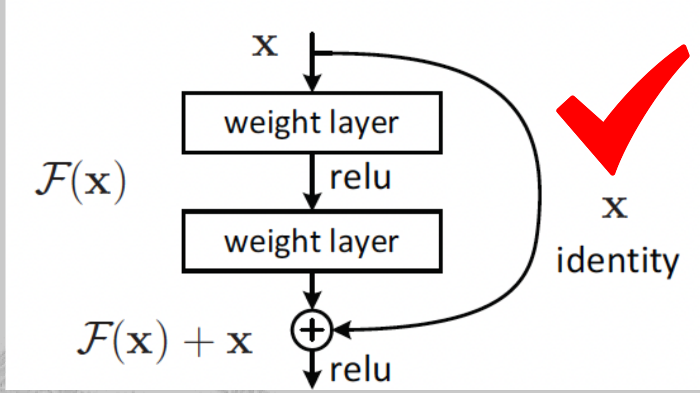


4. 梯度消失问题

- 模型结构的角度



- 模型深度并非越大越好

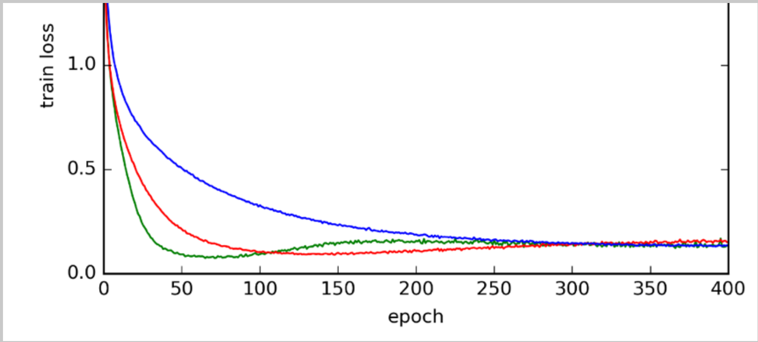


- 传统深度神经网络模型：$F(x)$
- 深度残差神经网络模型：$F(x) + x$

课后作业

- 安装pytorch
- 针对CIFAR-10数据集，训练一个5-8层的全连接神经网络
  - 从训练集中划分5000个样本作为验证集
  - 汇报训练集和验证集的loss曲线
  - 汇报测试集的最高accuracy

每个
epoch的
平均loss



epoch

- epoch: 所有训练样本迭代完一次称为一个epoch
- 选择验证集上准确率最高的模型，汇报其在测试集上的准确率

谢 谢！