深度学习引论

章 毅，张 蕾，郭 泉

四川大学·计算机学院·人工智能系

机器智能实验室

http://www.machineilab.org/

作业回顾

■程序实现神经网络的前向计算

- 标量形式
- 向量形式

- 提供MATLAB模版
- 可以使用MATLAB或Python

标量形式:

$Input\ W^l, a^1$
$for\ l = 1{:}L$
$\qquad a^{l+1} = fc\_c(W^l, a^l)$
$return$

$\text{Function}\ fc\_c(W^l, a^l)$
$for\ i = 1{:}n_{l+1}$

$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l$$

$$a_i^{l+1} = f(z_i^{l+1})$$

end

向量形式:

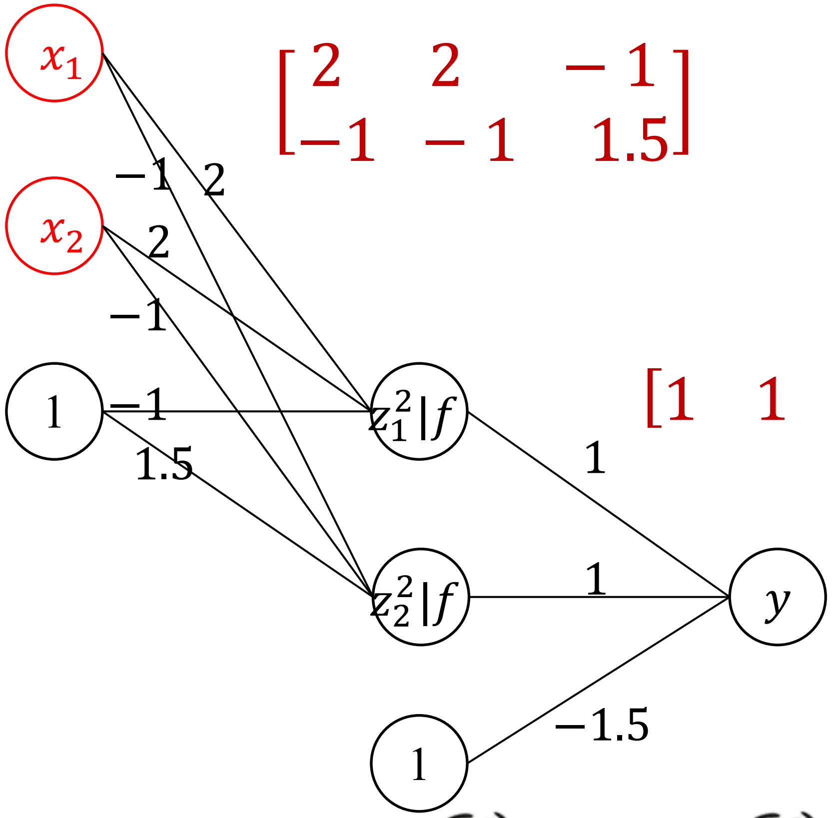
$Input\ W^l, a^1$
$for\ l = 1{:}L$
$\qquad a^{l+1} = fc\_v(W^l, a^l)$
$return$

$\text{Function}\ fc\_v(W^l, a^l)$

$$z^{l+1} = W^l a^l$$

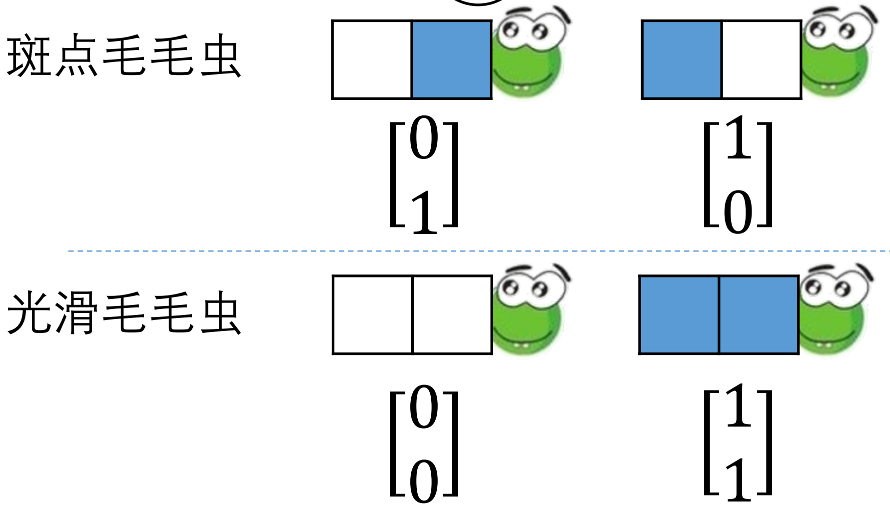
$$a^{l+1} = f(z^{l+1})$$

end

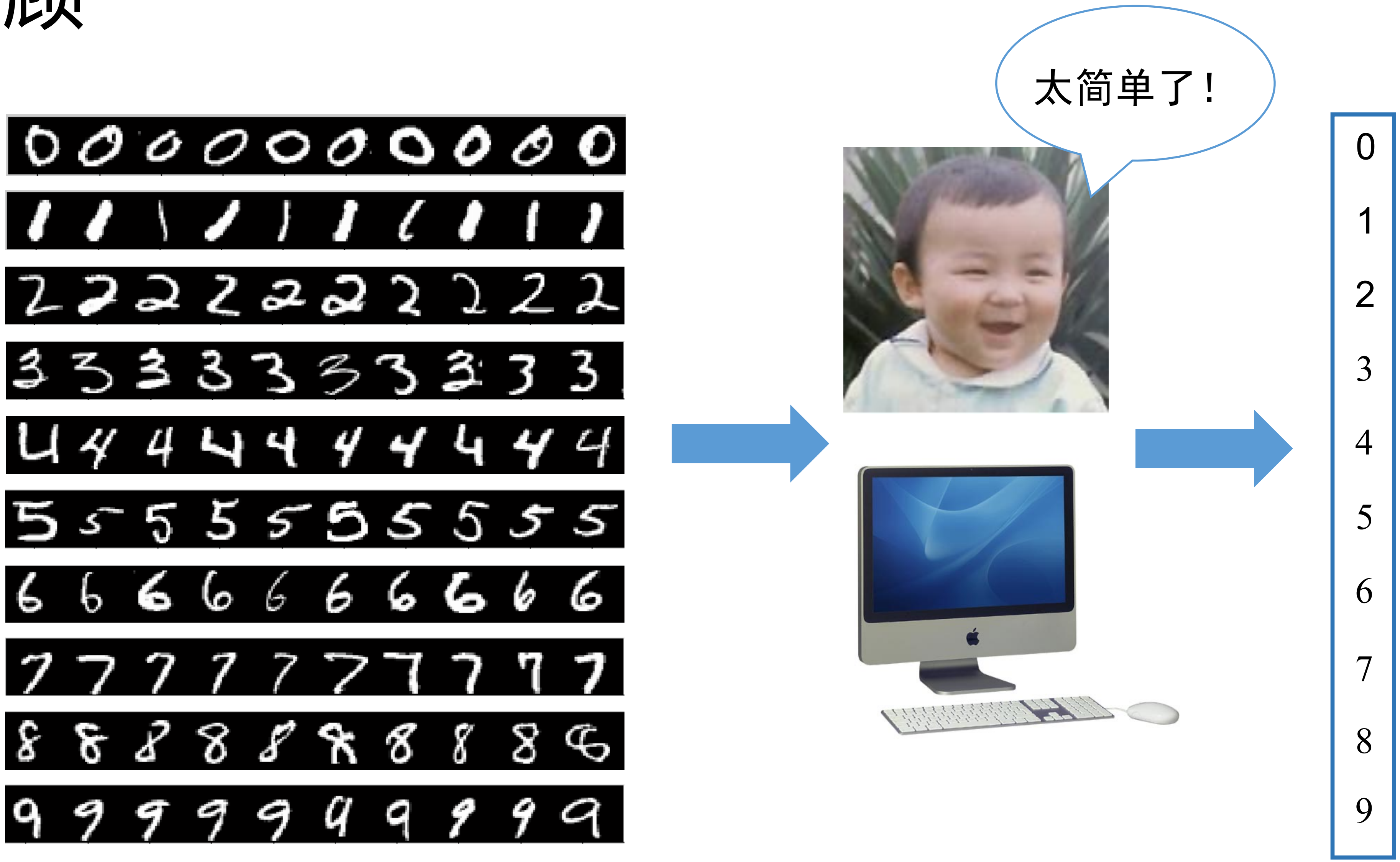


$$\begin{bmatrix} 2 & 2 & -1 \\ -1 & -1 & 1.5 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & -1.5 \end{bmatrix}$$

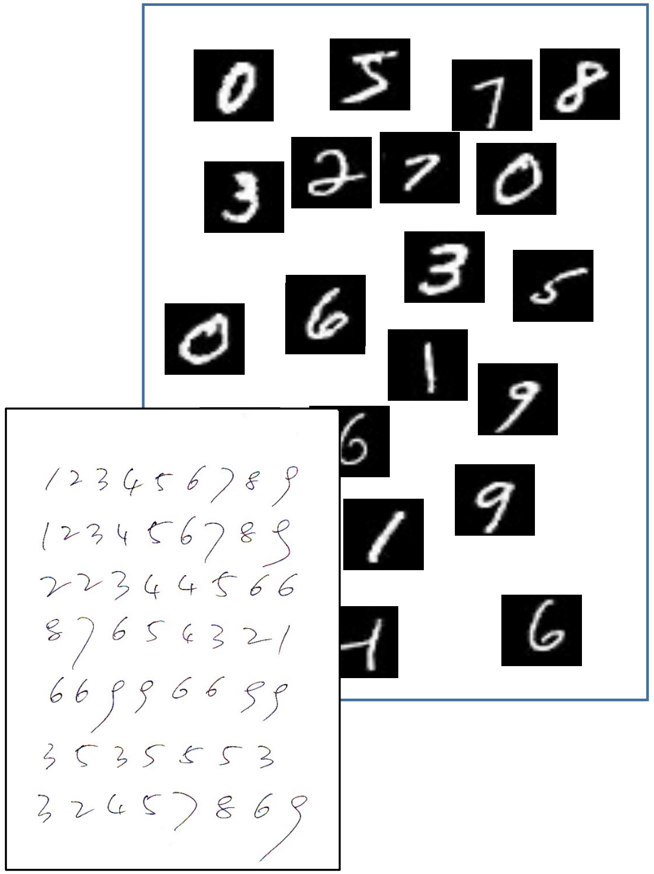


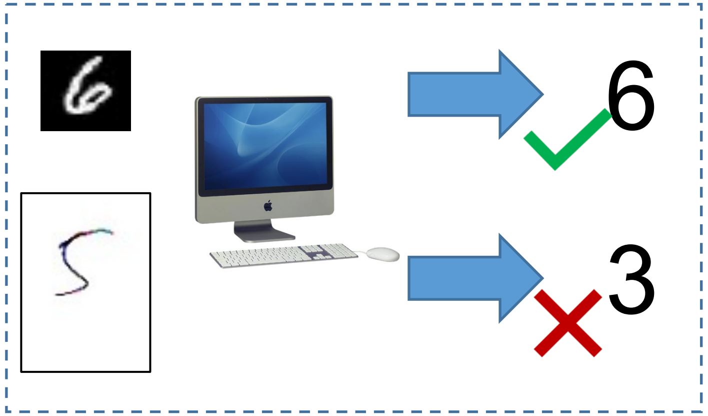
回顾



回顾

基本术语：样本、训练集、测试集、欠拟合、过拟合





样本：算法要处理、分析、预测的对象及标签的实例

训练集：训练期间使用的一组样本

测试集：一组用于测试的样本

欠拟合：在训练集上表现不佳

过拟合：在训练数据上表现很好，但在测试数据上表现不佳

回顾

- 知识是通过学习获取的
- 人的学习，一般分为三种模式：
  - 有教师学习
  - 无教师学习
  - 增强学习

- 学习表现为神经元之间新连接的建立和已有连接的修改

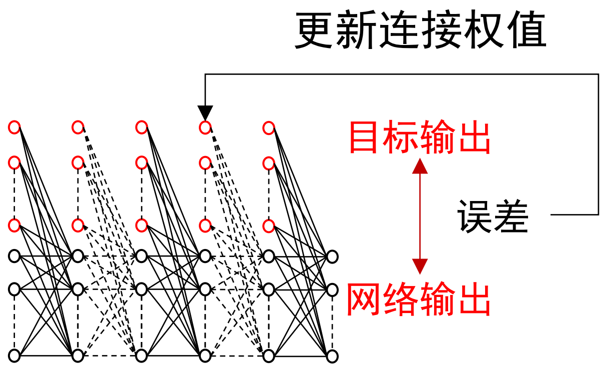
有教师学习：在教师的指导下，进行学习

无教师学习：没有教师的指导，自己学习

强化学习：在与环境的交互中，通过某种奖励机制学习

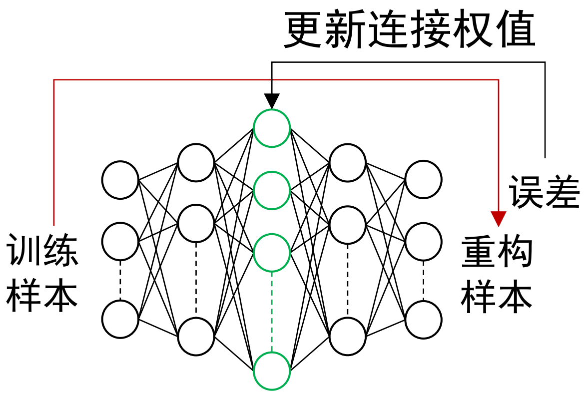
回顾

- 学习是指按照某种规则改变连接权值的过程。
- 类比人的三种学习模式，神经网络通过三种方式进行学习。



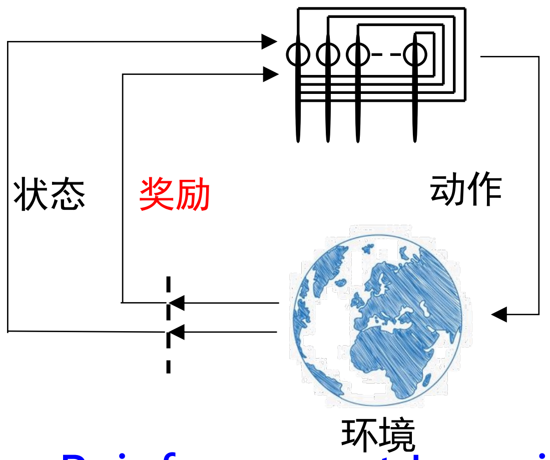
- Supervised Learning
- 有监督的学习

依据训练样本的目标输出与网络实际输出比较，更新连接权值



- Unsupervised Learning
- 无监督的学习

没有目标输出时，通过重构训练样本，更新连接权值



- Reinforcement Learning
- 强化学习

在与环境的交互中，以获得最大奖励为目标更新连接权值

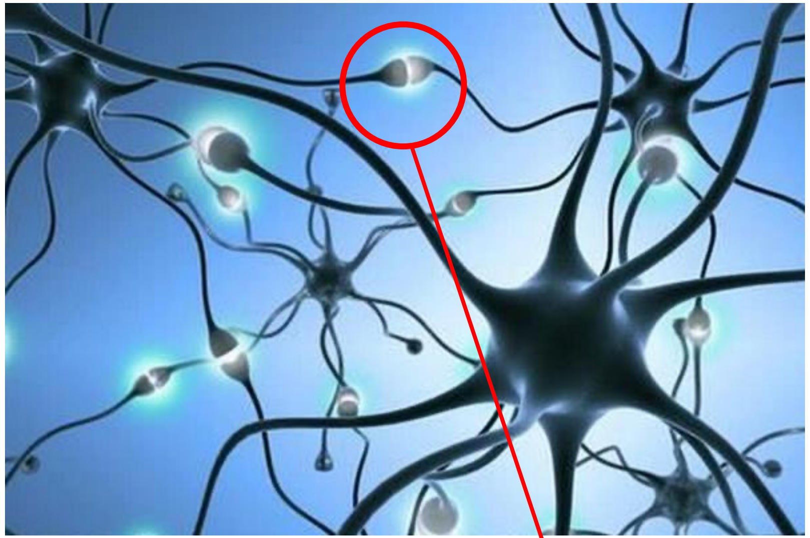
深度学习引论

第三章

反向传播算法

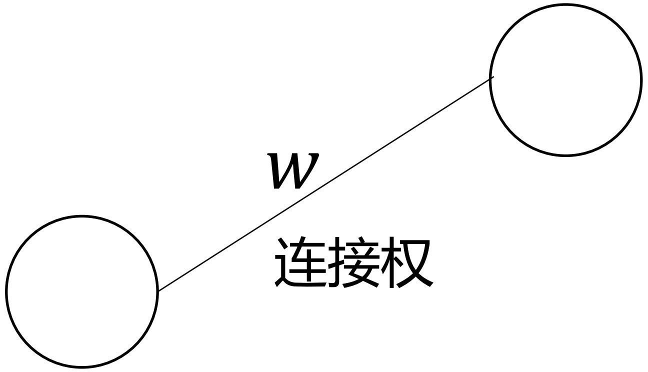
2022年 秋季

突触的可计算模型



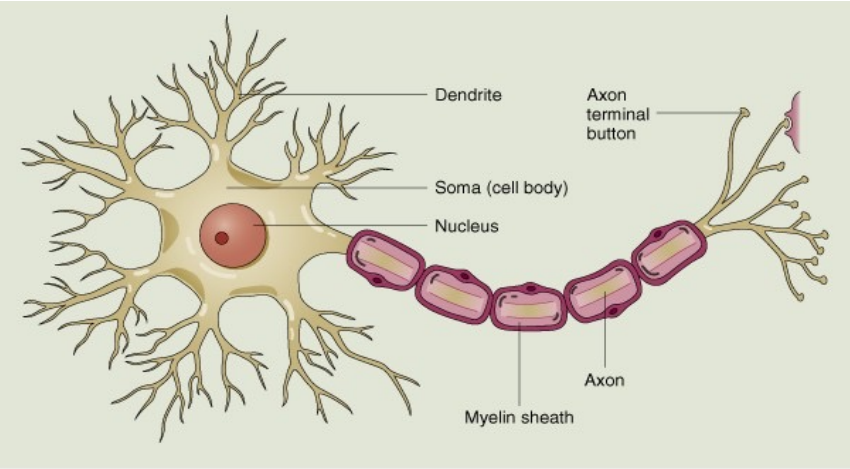
突触

≈

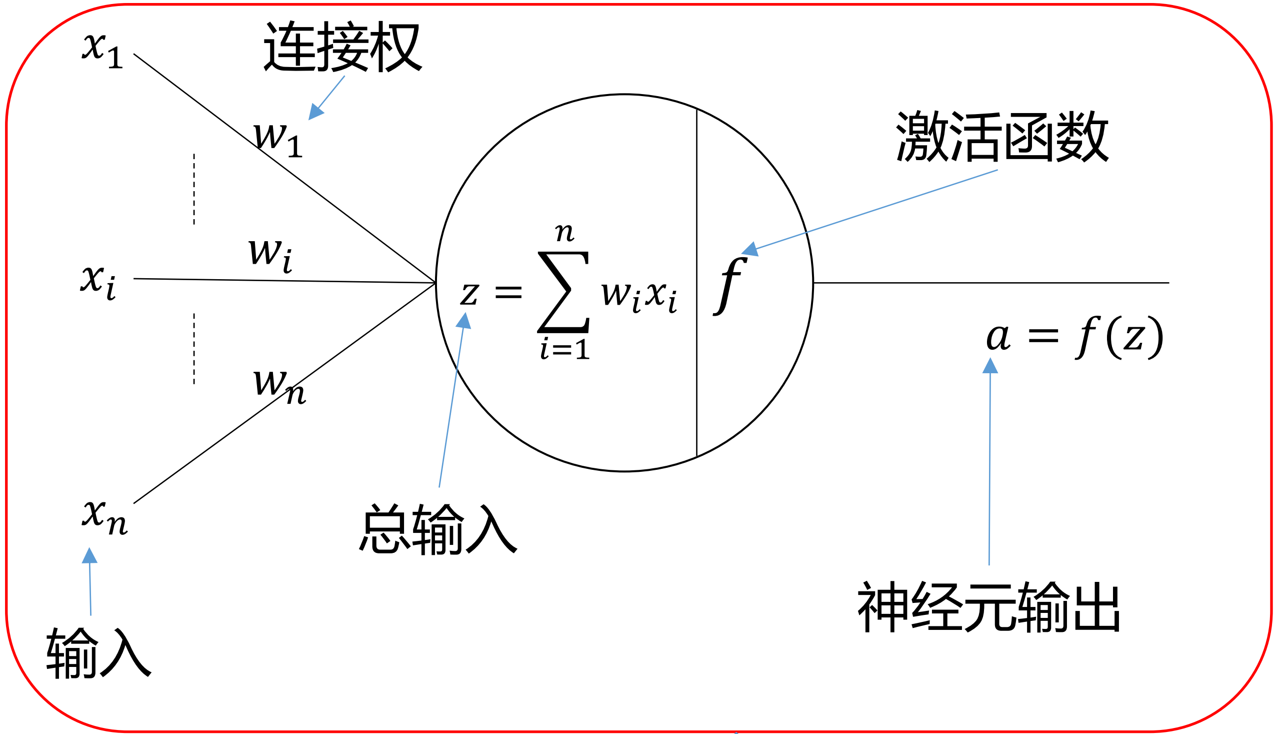


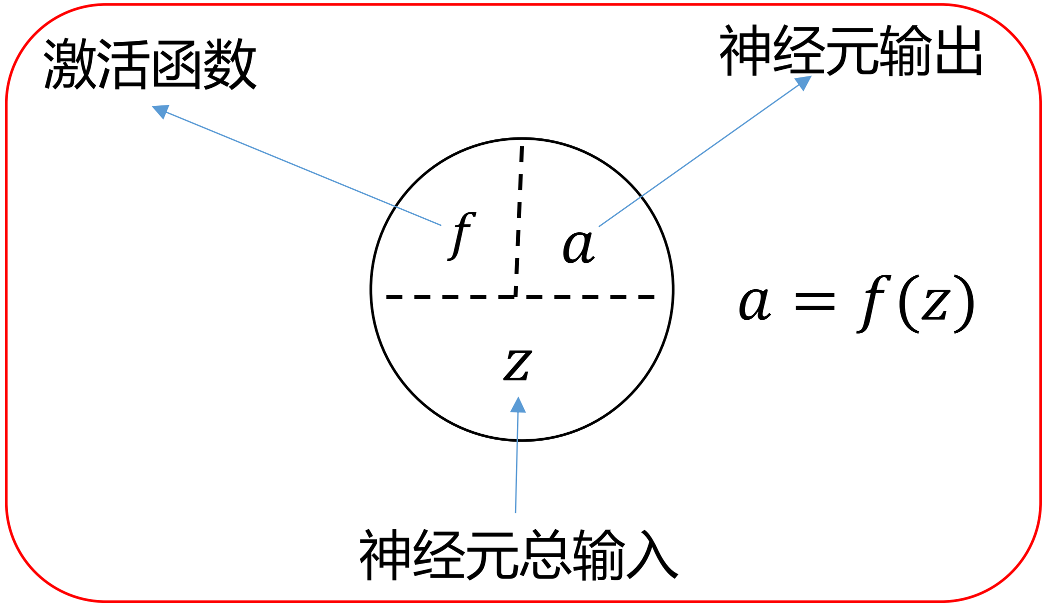
$w > 0$, 激励连接
$w = 0$, 无连接
$w < 0$, 抑制连接

神经元的可计算模型



≈



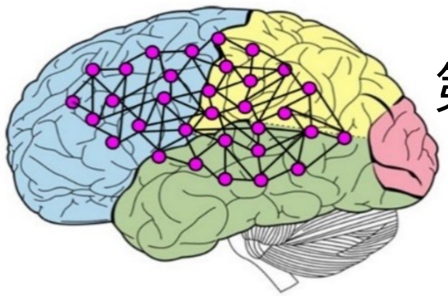


- 神经元上定义了三个概念
  - 神经元总输入
  - 神经元输出
  - 激活函数

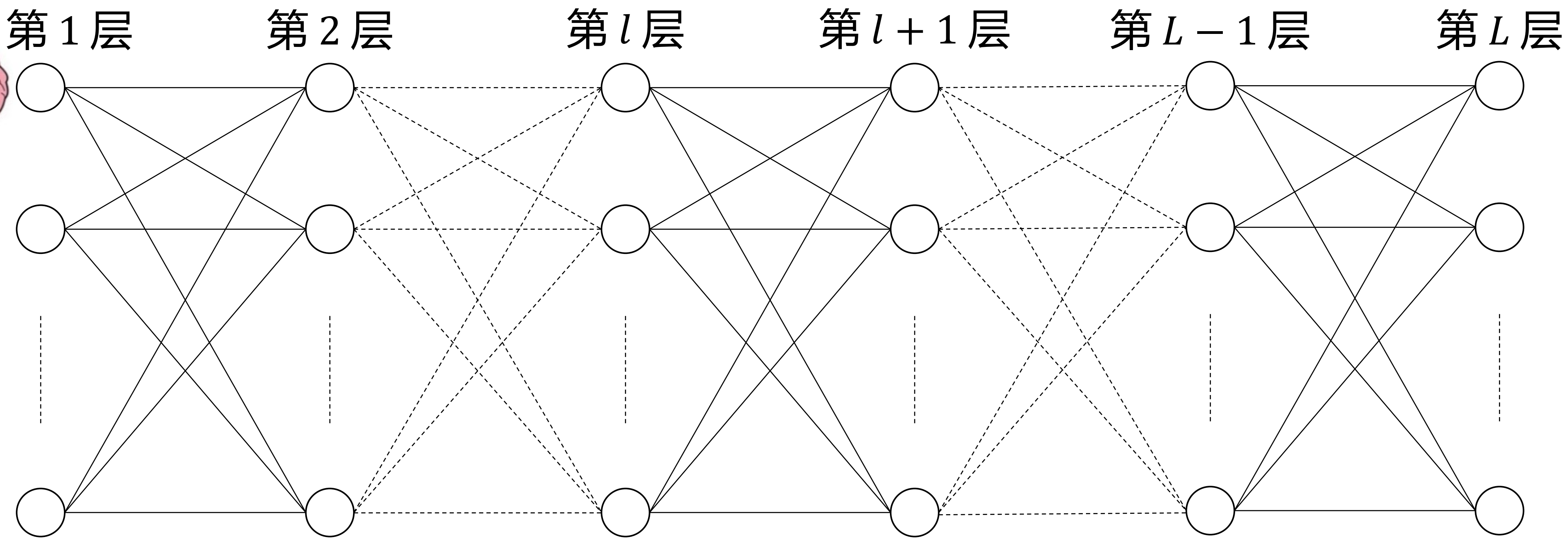
$$y = f(z) = f\left(\sum_{i=1}^{n} w_i x_i\right)$$

$$z = \sum_{i=1}^{n} w_i x_i$$

神经网络的可计算模型



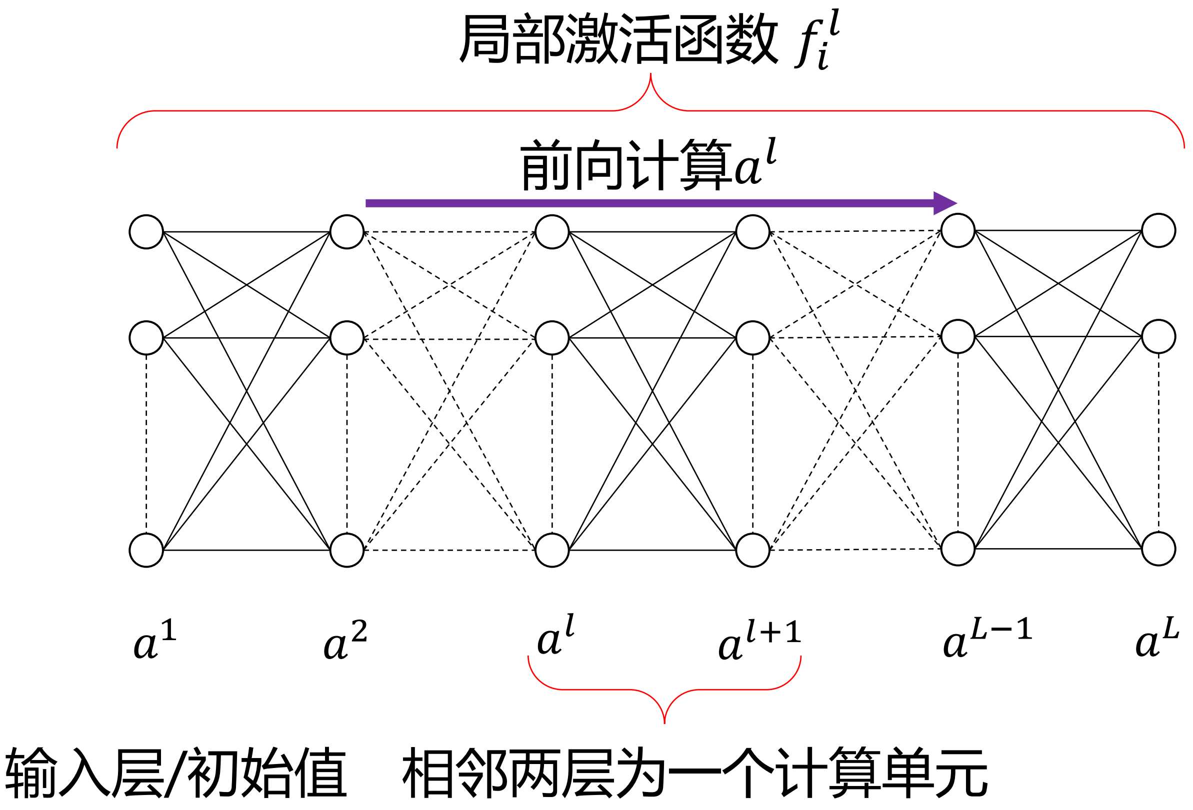
$\approx$



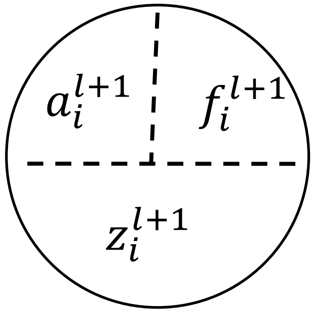
神经元	n_1		n_2		n_l		n_{l+1}		n_{L-1}		n_L
连接权		W^1		……		W^l		……		W^{L-1}	
预测	a^1		a^2		a^l		a^{l+1}		a^{L-1}		a^L

前向计算

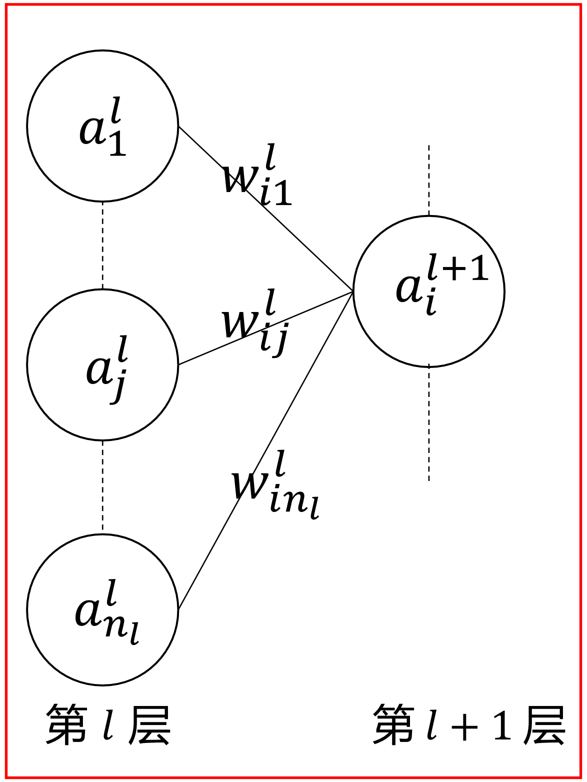
- 网络上定义一种运算：前向计算



第 $l+1$ 层的第 i 个神经元

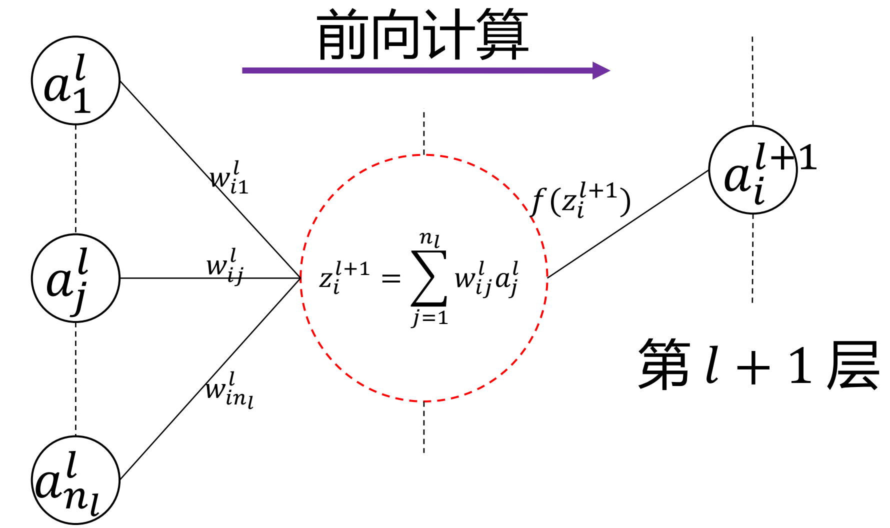


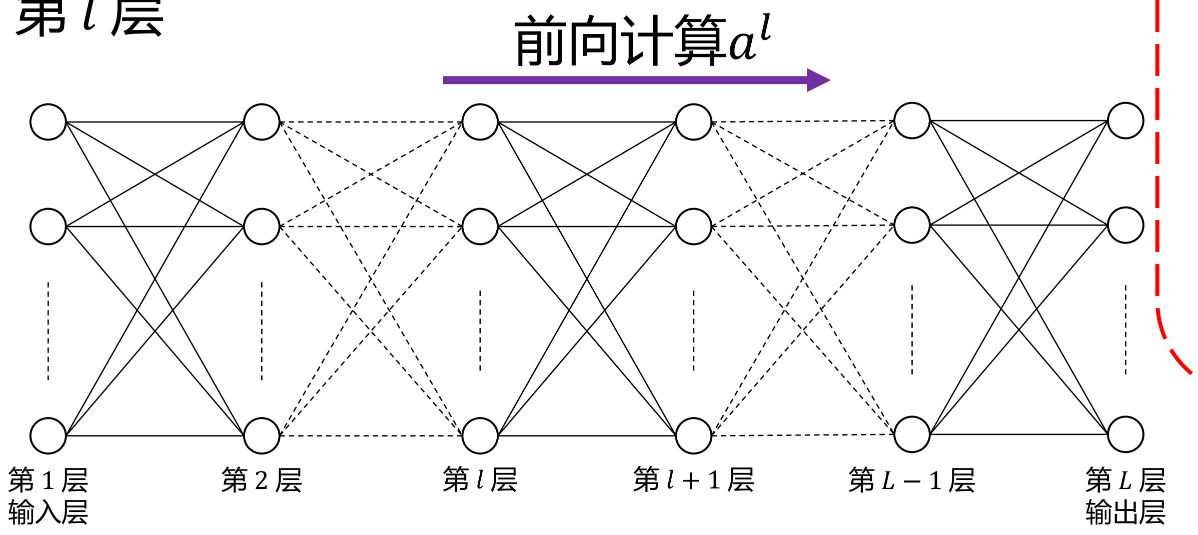
$$a_i^{l+1} = f_i^{l+1}(z_i^{l+1})$$



计算单元

仅一页 ppt 理解前向计算





标量形式

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

向量形式

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

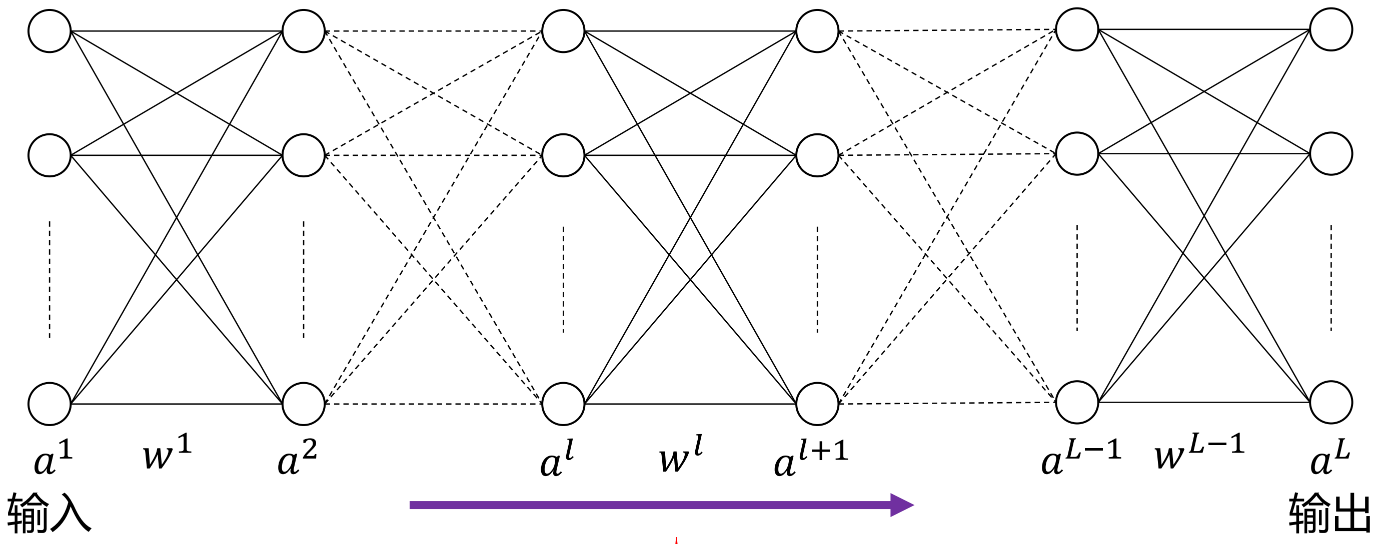
$$a_i^{l+1} = f\left(\sum_{j=1}^{n_l} w_{ij}^l a_j^l\right)$$

Algorithm:

$Input\ W^l, a^1$
$for\ l = 1: L$
$\quad a^{l+1} = fc\left(W^l, a^l\right)$
$return$

function $fc\left(W^l, a^l\right)$
$for\ i = 1: n_{l+1}$
$\quad z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l$
$\quad a_i^{l+1} = f\left(z_i^{l+1}\right)$
end

非线性映射 / 动力学系统



- 两种观点看待神经网
  - 非线性映射
  - 动力学系统

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

$$R^{n_1} \longrightarrow R^{n_L}$$

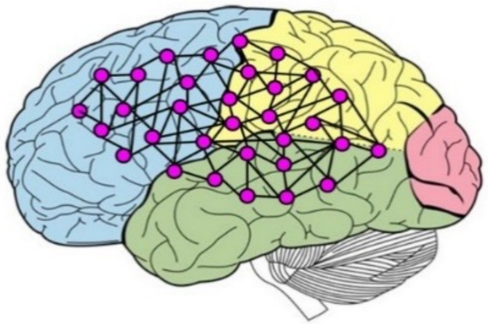
非线性映射

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

离散时间的动力学系统

神经网络的可计算模型

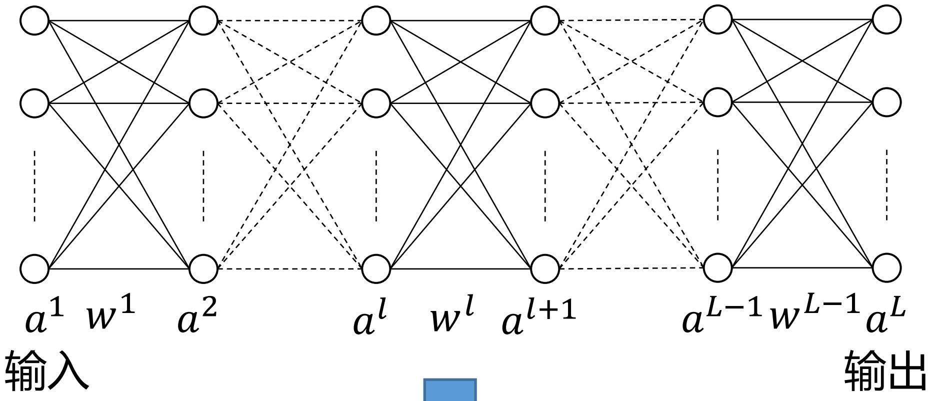
生物神经网络



生物智能

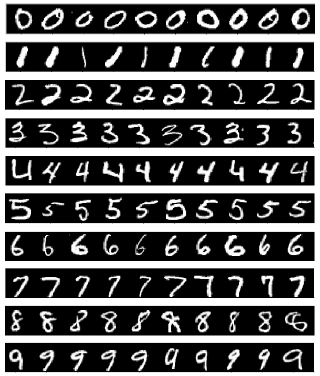
≈

人工神经网络

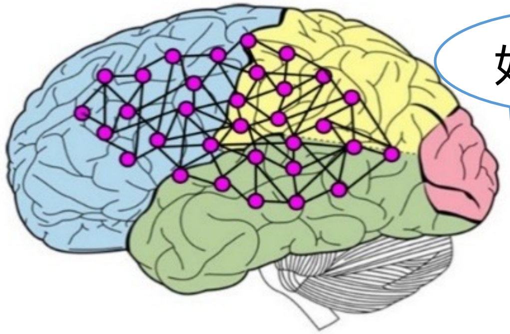


人工智能

手写体数字识别



生物神经网络

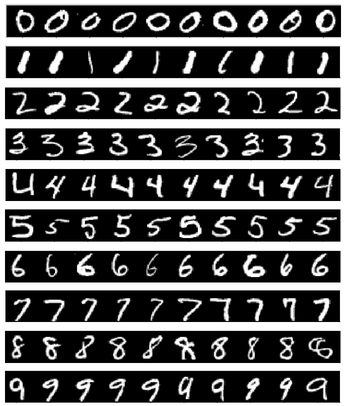


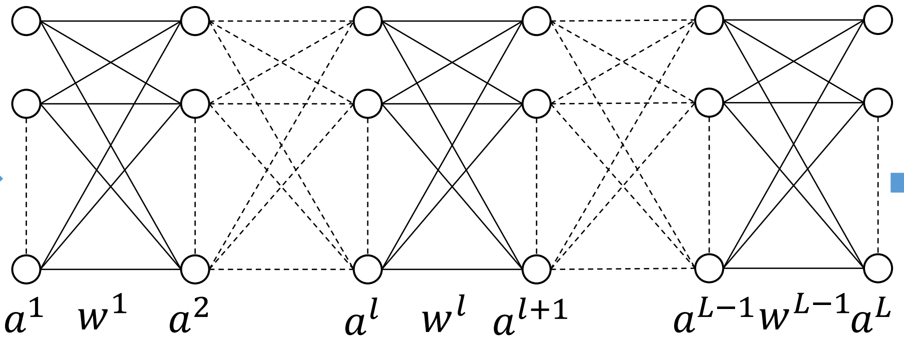
如此简单!

0
1
2
3
4
5
6
7
8
9

≈

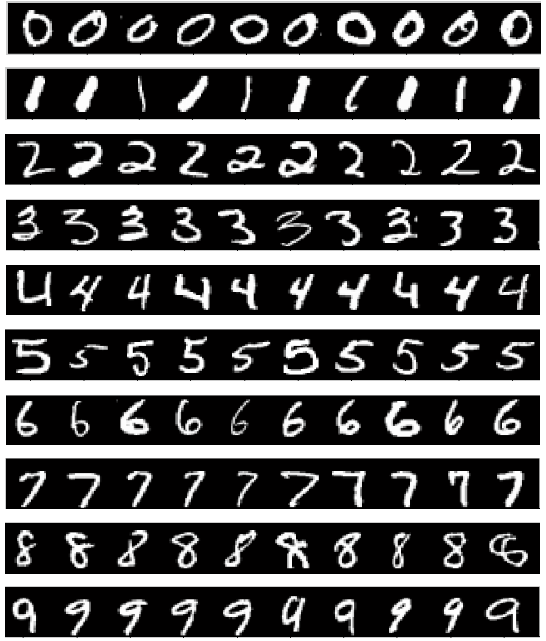
人工神经网络



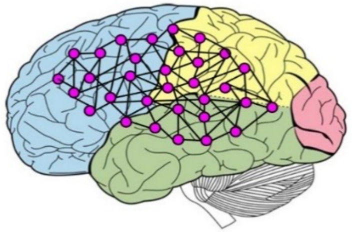


0
1
2
3
4
5
6
7
8
9

手写体数字识别



如此简单!



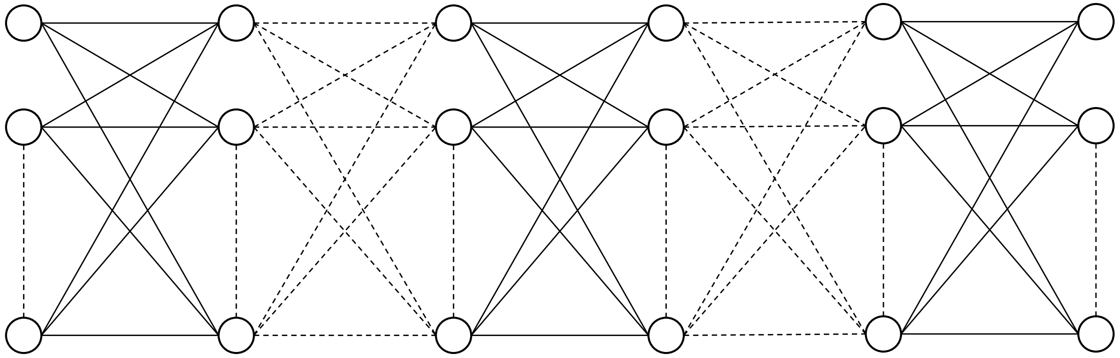
0
1
2
3
4
5
6
7
8
9

人类的大脑是如此强大，以至于任何一个小孩都能轻易地学会手写体数字识别。这里有两个重要因素:

1. 大脑具有神经网络结构
2. 大脑有学习的能力

下一步：如何建立人工神经网络的学习算法来训练人工神经网络模型?

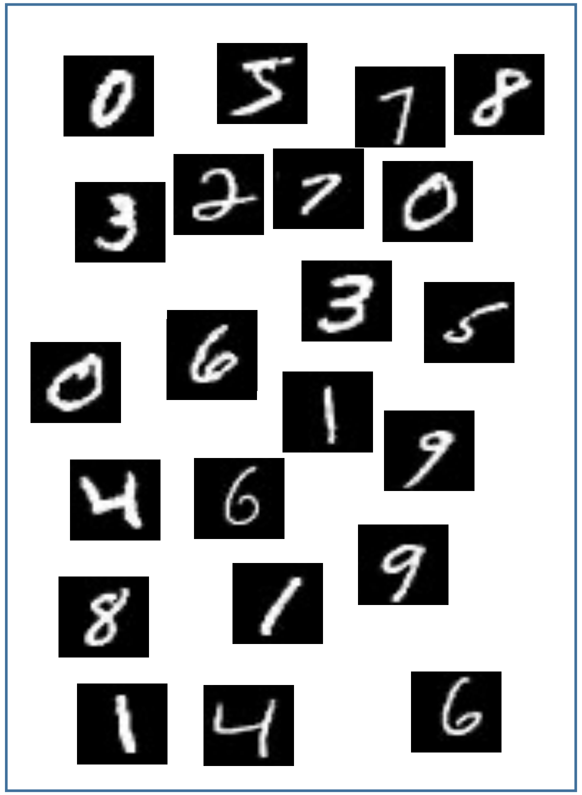
人工神经网络



提纲

- 网络性能刻画：性能函数
- 寻找最佳性能：最速梯度下降法
- 反向传播原理
- 反向传播算法

网络性能刻画：性能函数



训练集

训练

父亲知道正确答案

有监督学习

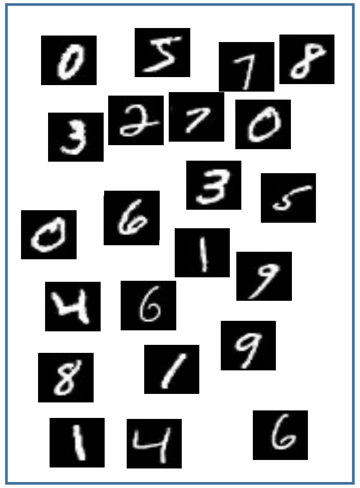
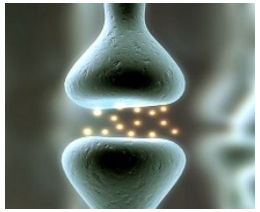
9

5

两个重要因素：

1. 需要有一个方法来度量正确答案和小孩的答案之间的误差——性能函数
2. 需要有一种机制来改变小孩的知识系统——学习算法

网络性能刻画：性能函数


更新知识系统

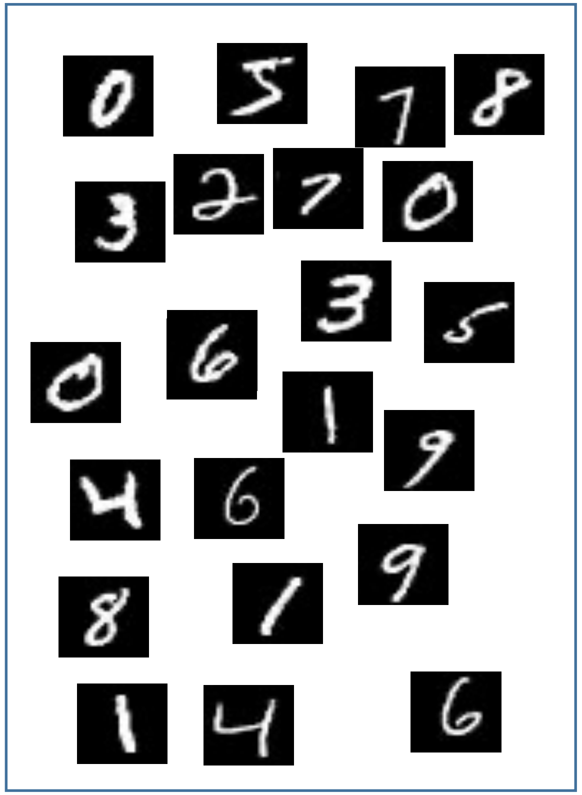
--更新突触连接强度



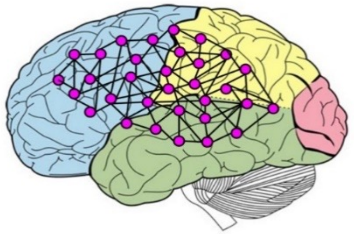
训练

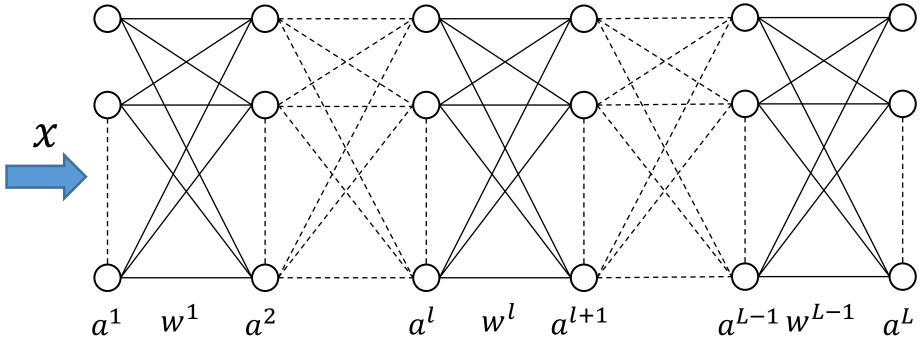
小孩的神经网络产生的预测		父亲的目标输出		误差
9	−	6	=	3
5	−	5	=	0

网络性能刻画：性能函数



训练数据集





更新网络连接权：学习算法

网络预测

$$a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$$

目标

$$y^L = \begin{bmatrix} y_1^L \\ \vdots \\ y_{n_L}^L \end{bmatrix}$$

$J(a^L, y^L)$

性能函数或代价函数 $J(a^L, y^L)$ 用来刻画 a^L 和 y^L之间的距离，$J(a^L, y^L)$ 本质上是$(w^1, \cdots, w^L)$的函数，即

$$J = J(w^1, \cdots, w^L).$$

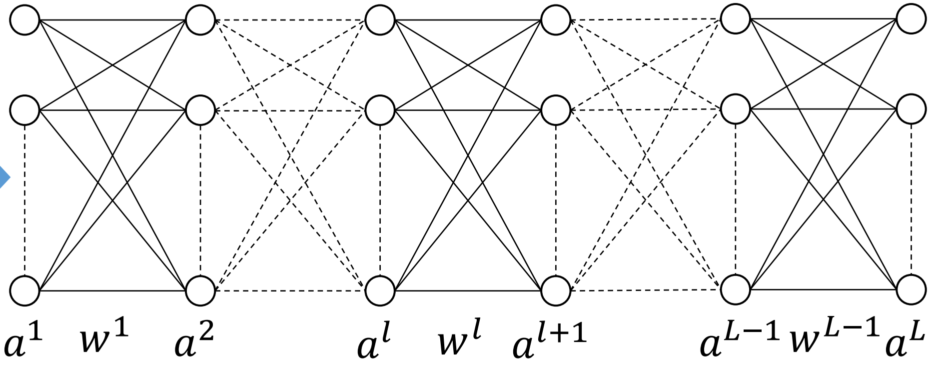
有监督的学习

训练数据

$D = \{(x, y^L)\}$

训练样本 (x, y^L)

输入 x



网络预测　　　　目标

$$\underbrace{a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix} \qquad y^L = \begin{bmatrix} y_1^L \\ \vdots \\ y_{n_L}^L \end{bmatrix}}_{\text{性能函数}}$$

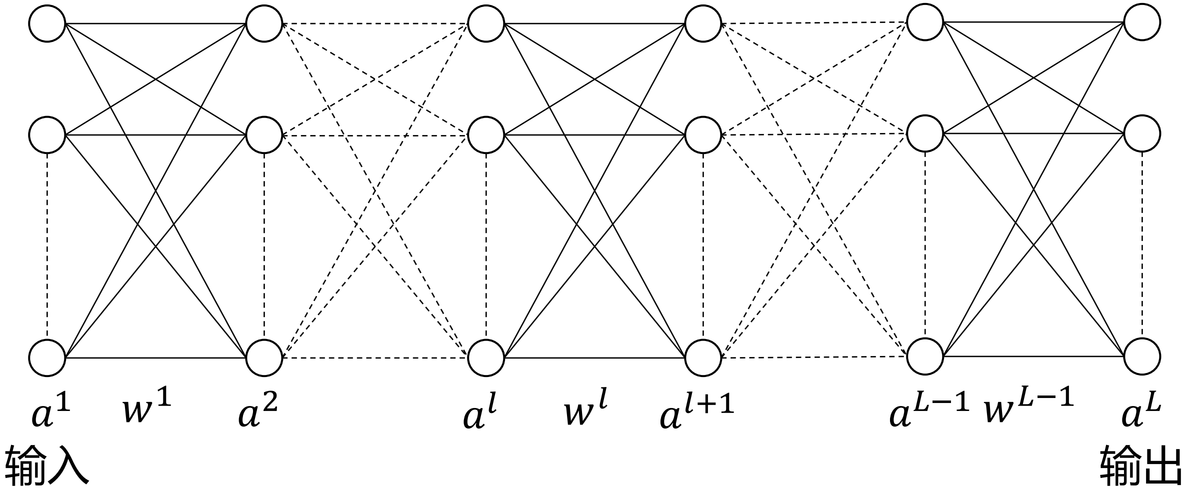
更新网络连接权：学习算法

$$J(a^L, y^L) = J(w^1, \cdots, w^L)$$

在有监督学习中，每个训练样本包含输入样本和对应的目标输出。

问题：如何构造性能函数

网络性能刻画：性能函数



性能函数 J 可以刻画整个网络的性能。如果 J 很小，意味着网络的预测 a^L 距离目标输出 y^L 很近，网络表现良好。因为 J 是变量 $(w^1,\cdots,w^L)$ 的函数，网络表现良好意味着找到一组合适的 $(w^1,\cdots,w^L)$ 使得 J 很小。寻找合适的 $(w^1,\cdots,w^L)$ 的过程叫做网络的学习。

问题: 如何学习？

目标　　网络预测

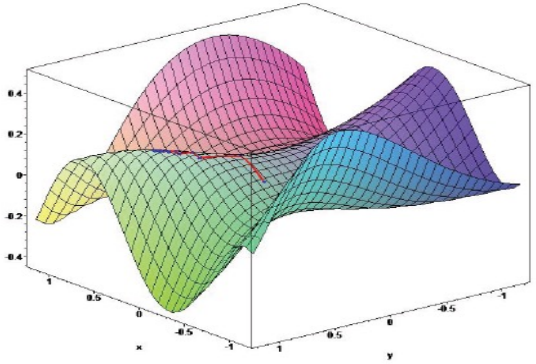
$$y^L = \begin{bmatrix} y_1^L \\ \vdots \\ y_{n_L}^L \end{bmatrix} \qquad a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$$

构造性能函数的方法有多种，一个常见的性能函数如下：

$$e_j = a_j^L - y_j^L$$

$$J = \frac{1}{2}\sum_{j=1}^{n_L} e_j^2 = J(w^1,\cdots,w^L)$$

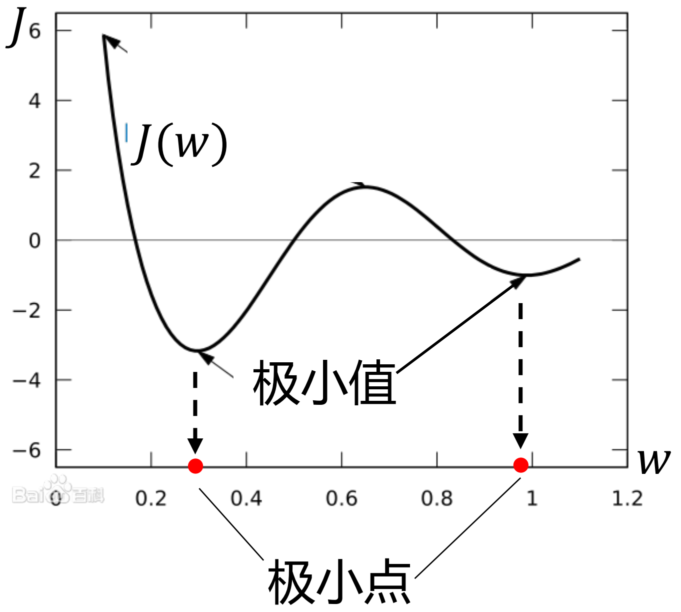
显然，J 是 $w^1,\cdots,w^L$ 的函数。



网络性能刻画：性能函数

学习是使得网络输出a^L 逐渐向 y^L 靠近的过程，也就是要使得性能函数 J 取极小值。性能函数 $J = J(w^1, \cdots, w^{L-1})$ 是变量 $w^l(l = 1, \cdots, L)$ 的函数，因此网络的学习就是寻找性能函数J的极小点$w^l(l = 1, \cdots, L)$的过程。

问题: 如何寻找 J 的极小点?



目标　　　　网络预测

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

一个常见的性能函数：

$$J = \frac{1}{2}\sum_{j=1}^{n_L} e_j^2 = J(w^1, \cdots, w^L)$$

J 是 $w^1, \cdots, w^L$ 的函数

$\boxed{\text{学习 = 寻找性能函数} J \text{ 的极小点 } w^l(l = 1, \cdots, L)}$

提纲

- 网络性能刻画：性能函数
- 寻找最佳性能：最速梯度下降法
- 反向传播原理
- 反向传播算法

极小点

对于任意非线性函数

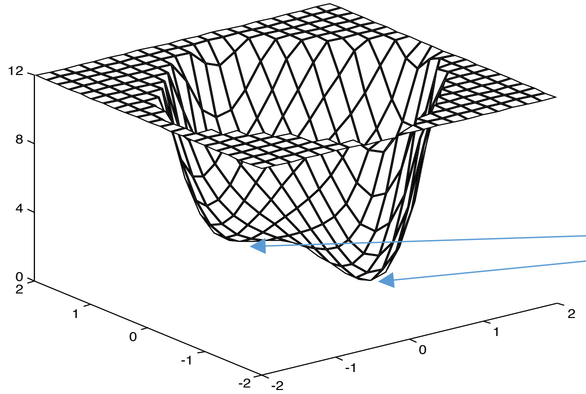
$$J(w), w \in R^n$$

如果对于任意非常接近w^*的点w都有

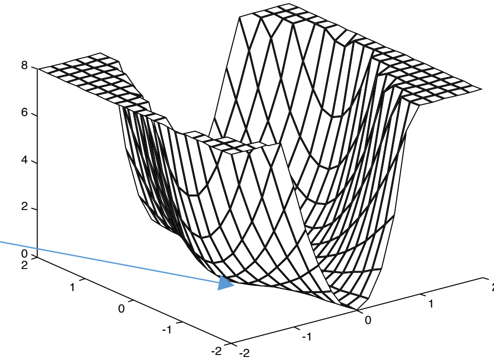
$$J(w^*) \le J(w)$$

称w^*为极小点，称$f(w^*)$为极小值。

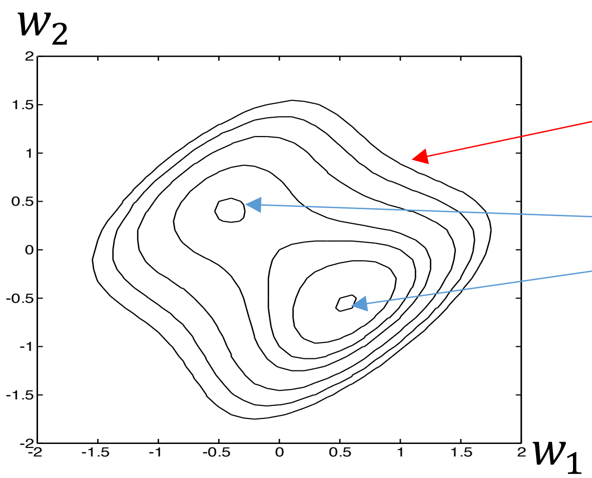
$$J(w_1, w_2) = (w_2 - w_1)^4 + 8w_1w_2 - w_1 + w_2 + 3$$



$$J(w_1, w_2) = (w_1^2 - 1.5w_1w_2 + 2w_2^2)w_1^2$$

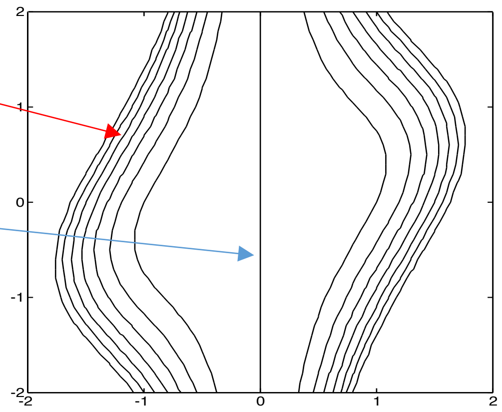


极小值



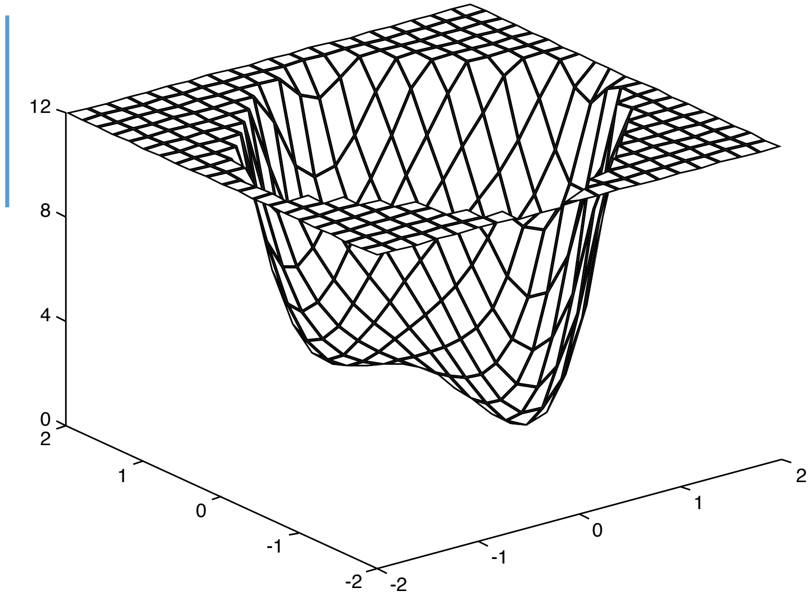
等值线

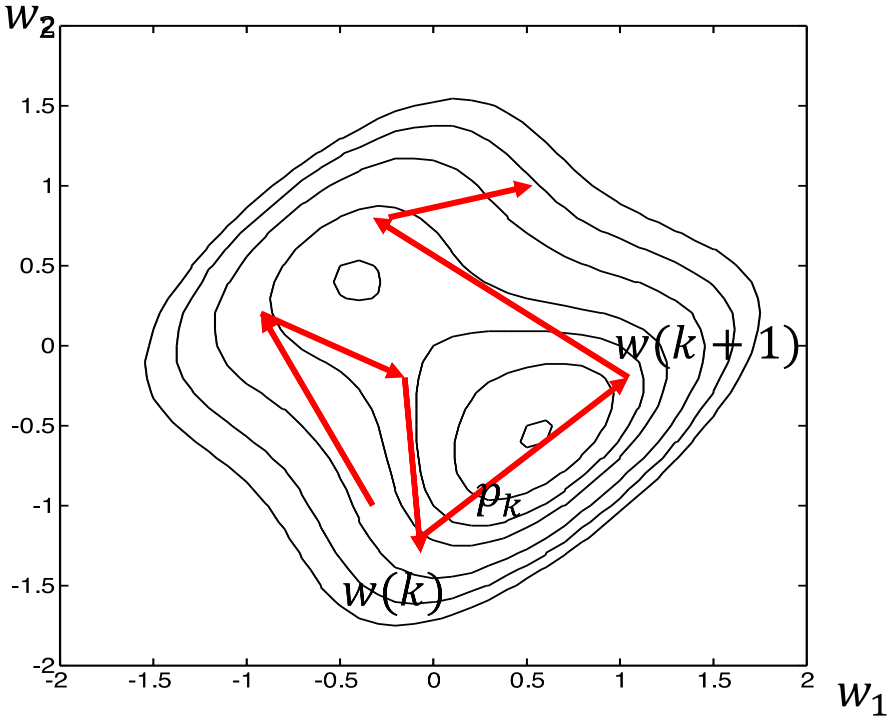
极小点



问题：
如何寻找极小点?

迭代法

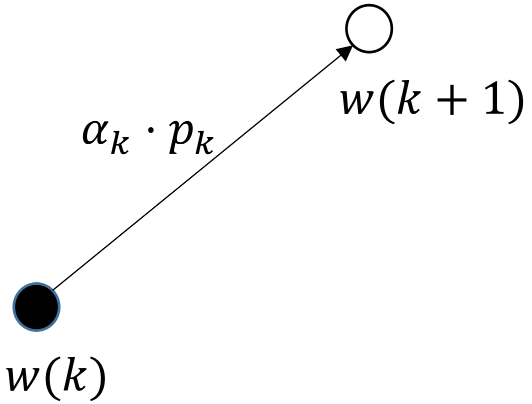




逐步迭代求极小点

$$w(k+1) = w(k) + \alpha_k \cdot p_k$$

首先需要一个起始点 $w(0)$，然后逐次迭代计算

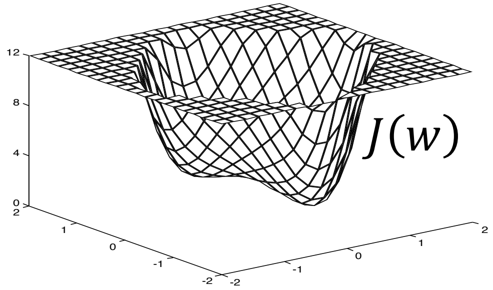


p_k 搜索方向

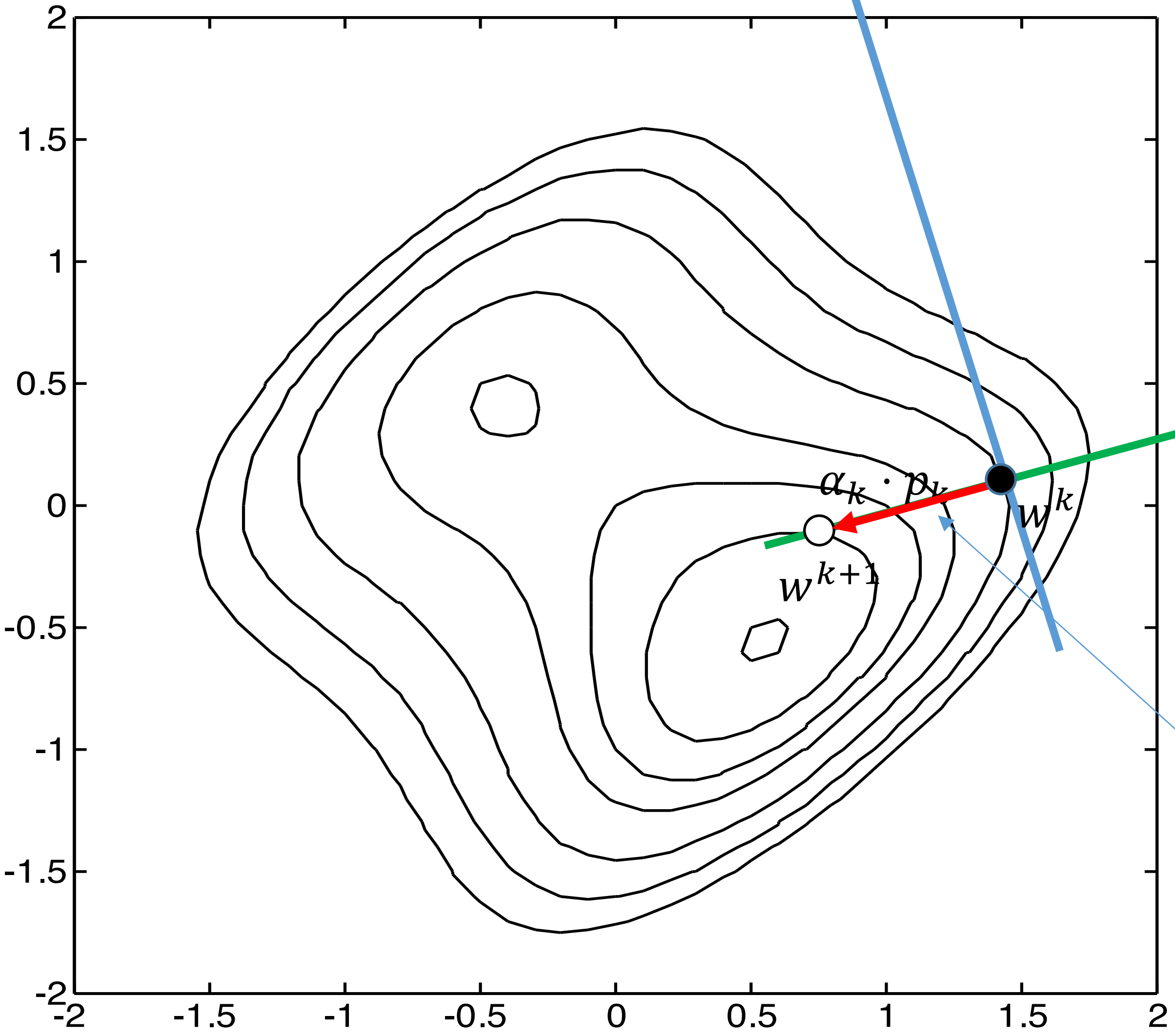
α_k 第 k 步的学习率

问题: 如何确定好的搜索方向 p_k?

最速梯度下降法



函数值变化最慢的方向



函数值上升最快的方向

梯度:

$$g_k = \nabla J(w)\Big|_{w(k)} = \frac{\partial J}{\partial w}\Big|_{w(k)} = \begin{pmatrix} \dfrac{\partial J}{\partial w_1} \\ \vdots \\ \dfrac{\partial J}{\partial w_n} \end{pmatrix}\Bigg|_{w(k)}$$

最速下降算法:

$$p_k = -g_k$$

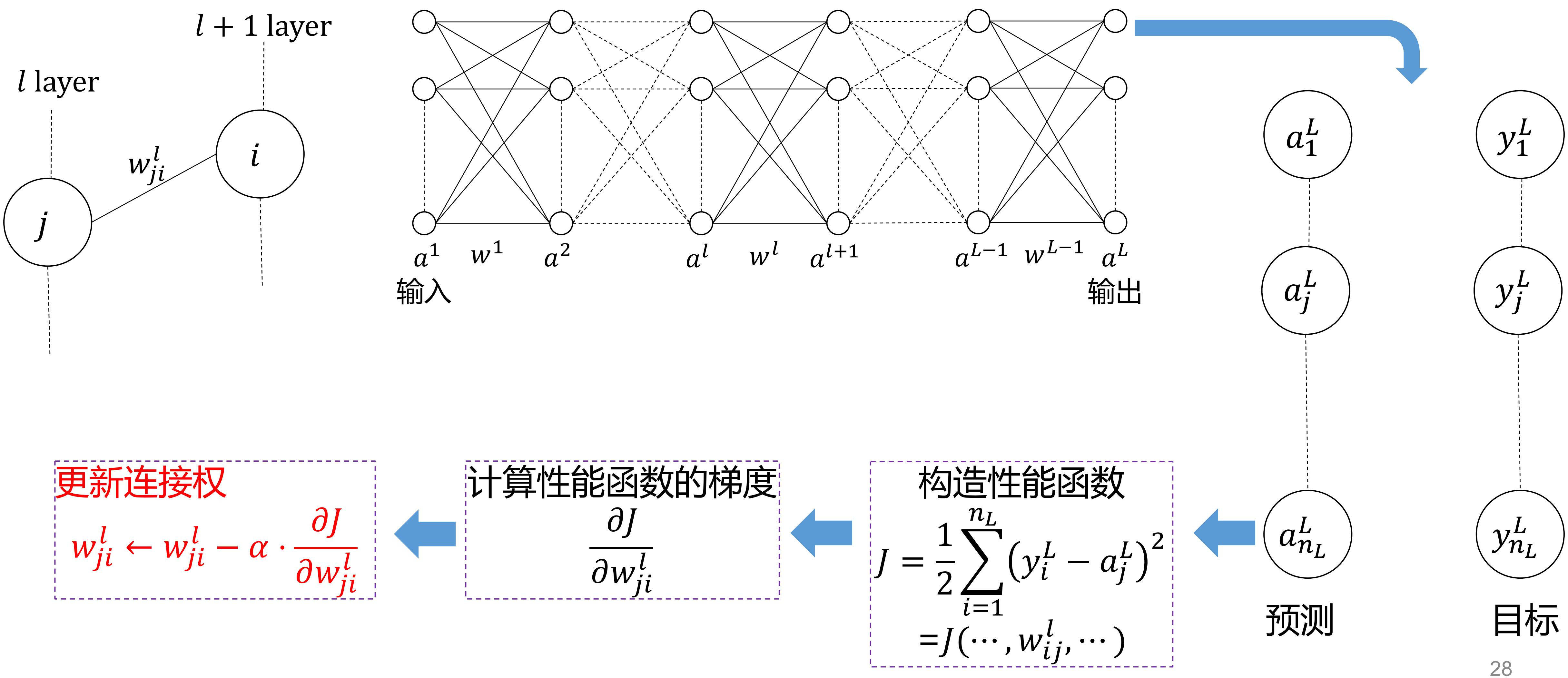
$$w(k+1) = w(k) - \alpha_k \cdot g_k$$

即

$$w(k+1) = w(k) - \alpha_k \cdot \frac{\partial J}{\partial w}\Big|_{w(k)}$$

函数值下降最快的方向

最速梯度下降算法

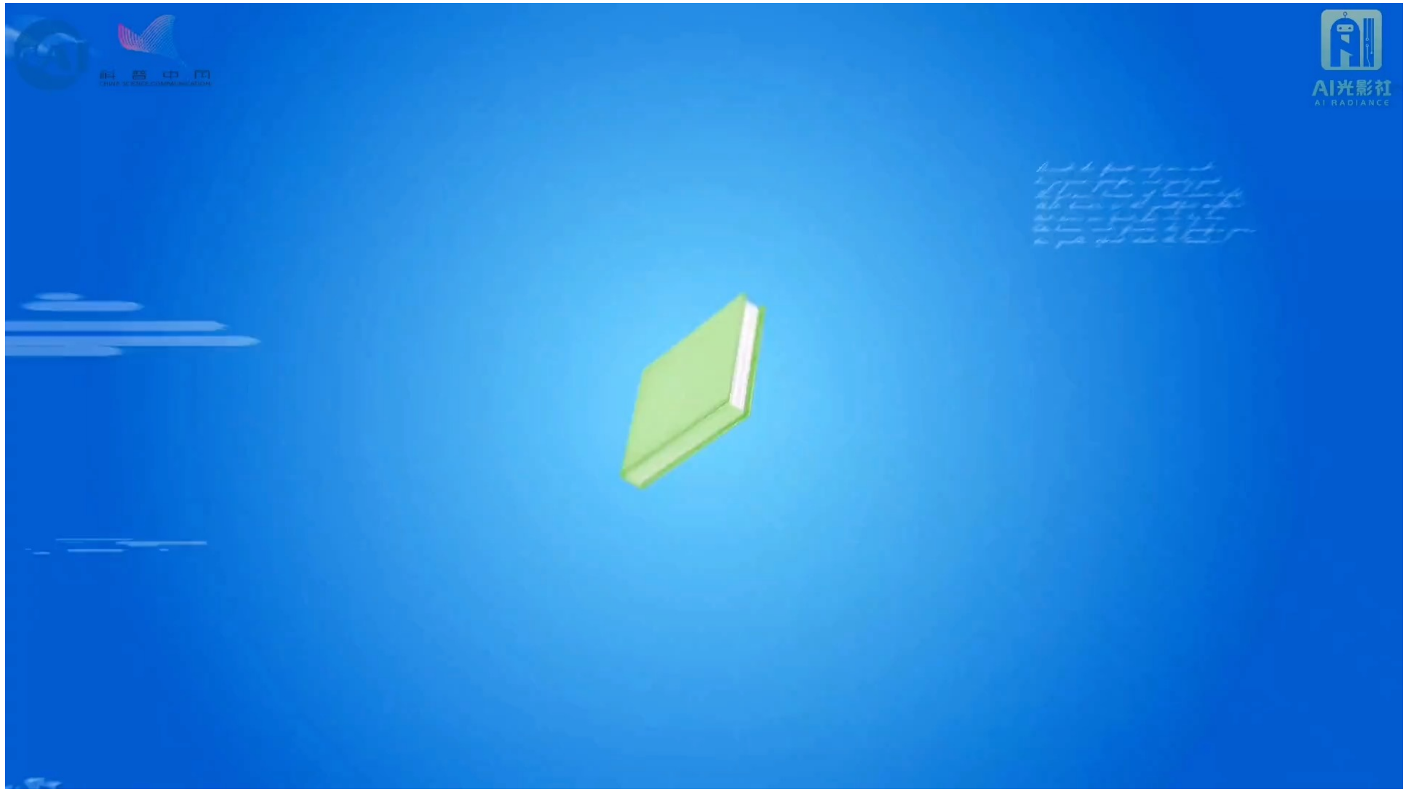


$l+1$ layer
l layer
w_{ji}^l
i
j
a^1 w^1 a^2 a^l w^l a^{l+1} a^{L-1} w^{L-1} a^L
输入
输出
a_1^L
a_j^L
$a_{n_L}^L$
y_1^L
y_j^L
$y_{n_L}^L$
预测
目标
更新连接权
$w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$
计算性能函数的梯度
$\frac{\partial J}{\partial w_{ji}^l}$
构造性能函数
$J = \frac{1}{2}\sum_{i=1}^{n_L}\left(y_i^L - a_j^L\right)^2$
$=J(\cdots, w_{ij}^l, \cdots)$

最速梯度下降算法

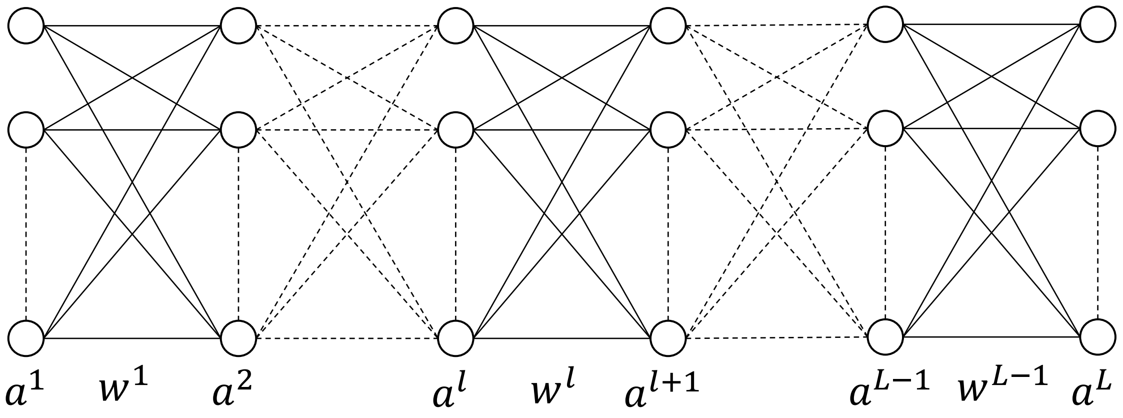
中国人工智能学会

公众号



https://mp.weixin.qq.com/s/wfZy6Ia0jcd7eBikmieeyA

最速梯度下降算法

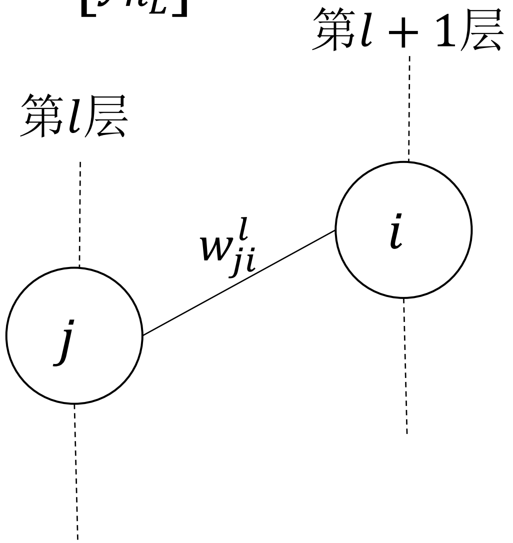


目标

$$y^L = \begin{bmatrix} y_1^L \\ \vdots \\ y_{n_L}^L \end{bmatrix}$$

预测

$$a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$$



最速梯度下降算法

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

1. 计算性能函数的梯度

$$\frac{\partial J}{\partial w_{ji}^l}$$

2. 更新网络连接权值

$$w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$$

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

问题：如何计算 $\frac{\partial J}{\partial w_{ji}^l}$？

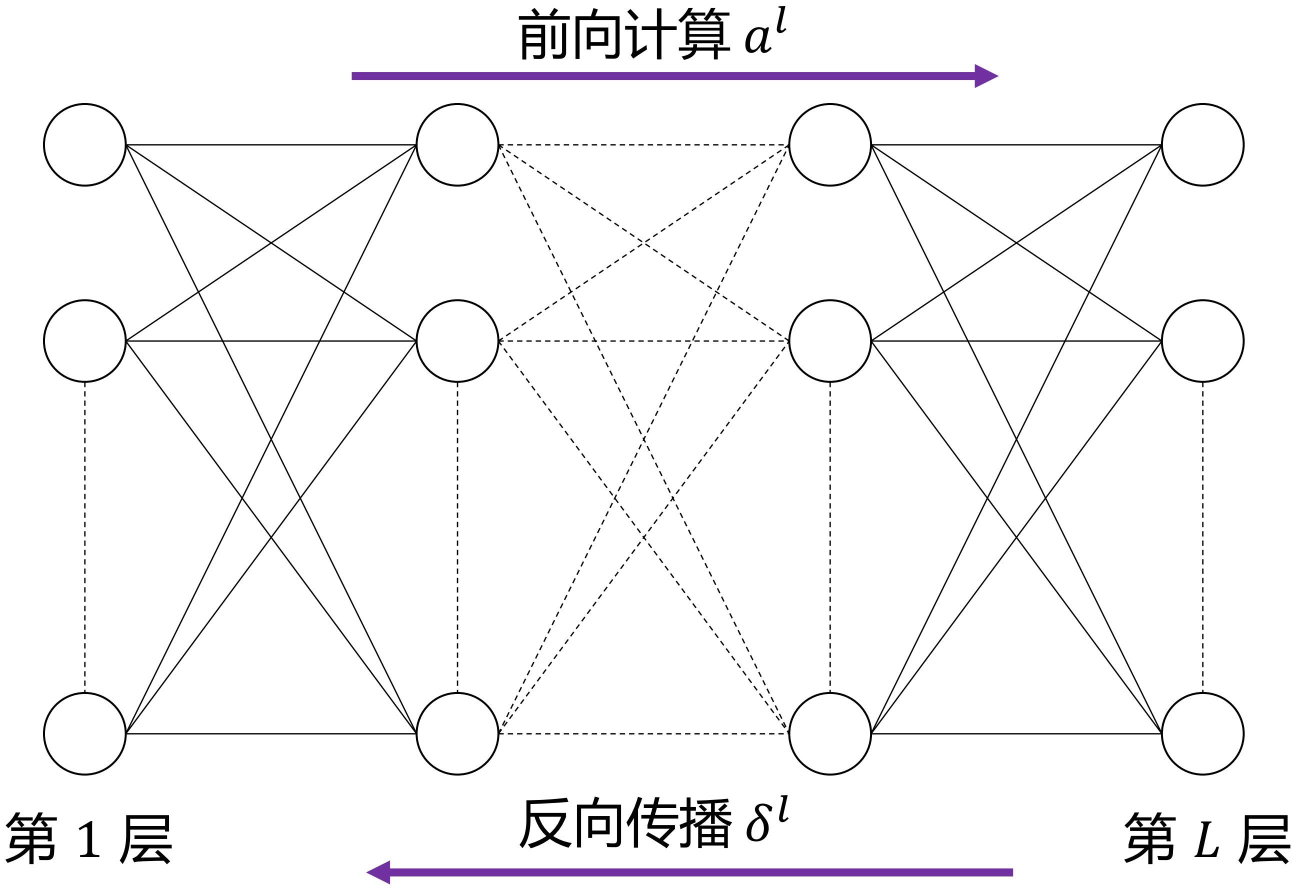
答案：

使用著名的反向传播算法

提纲

- 网络性能刻画：性能函数
- 寻找最佳性能：最速梯度下降法
- 反向传播原理
- 反向传播算法

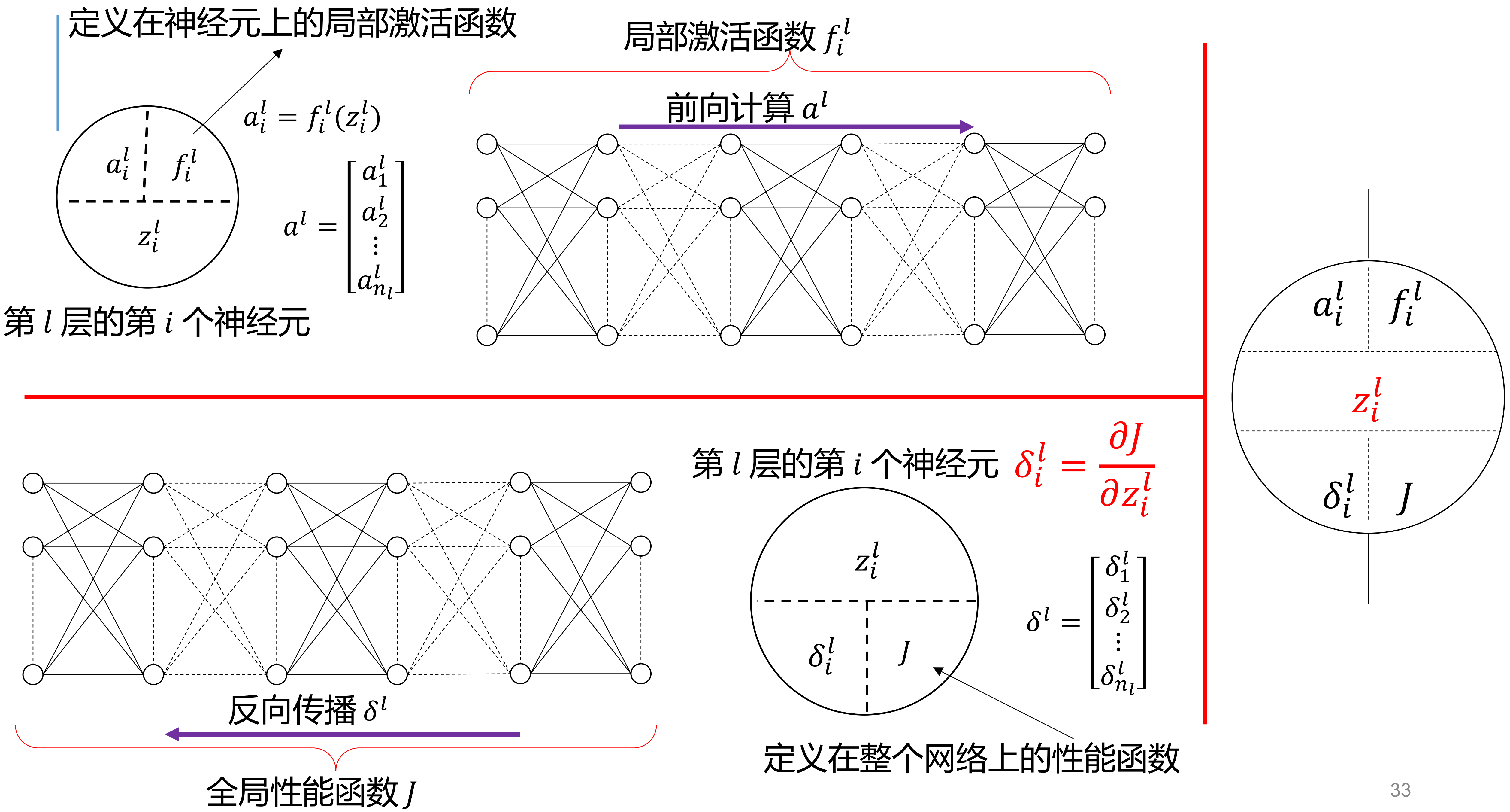
反向传播



反向传播是计算 $\frac{\partial J}{\partial w_{ji}^l}$ 的一种有效方法

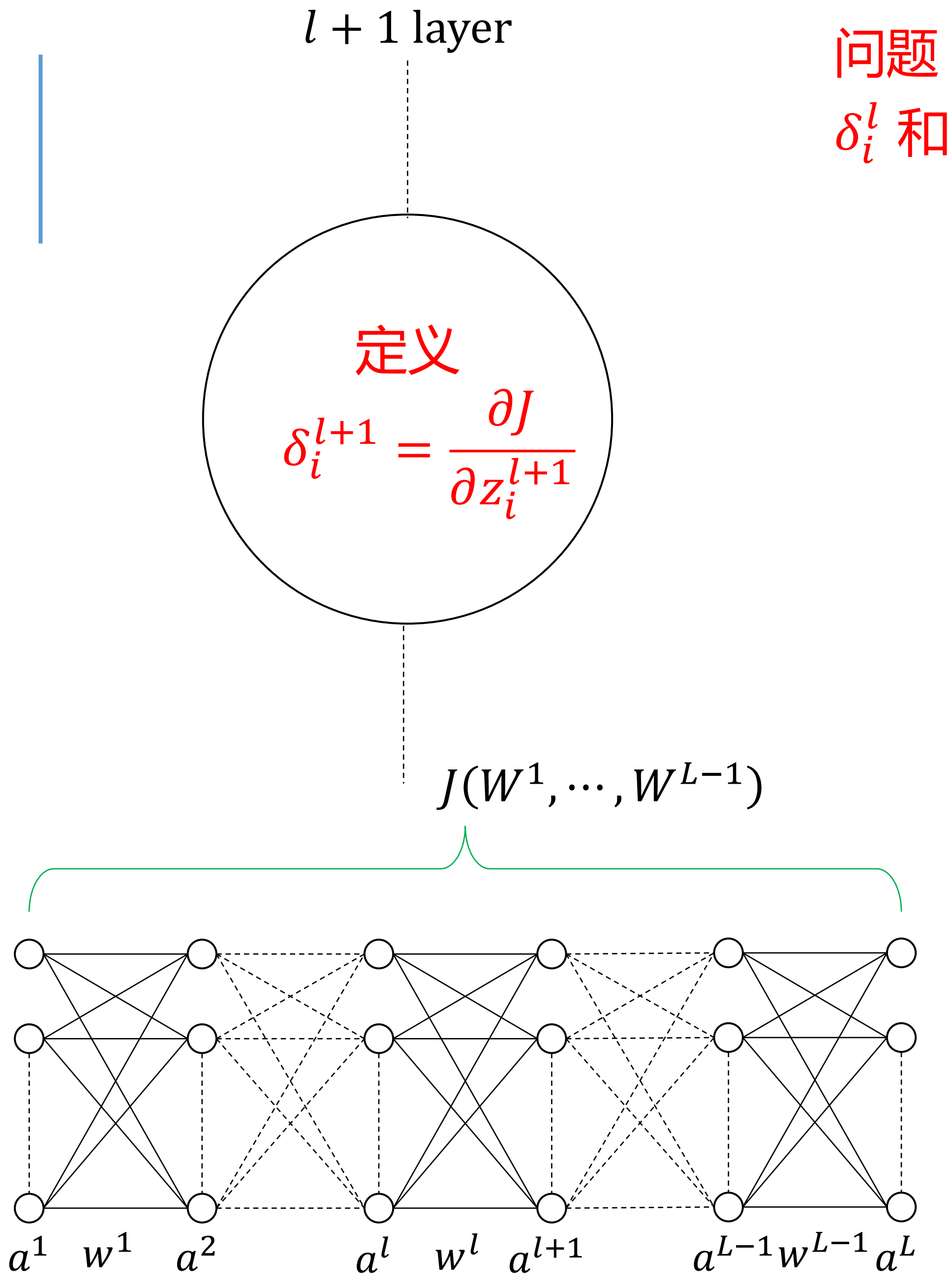
性能函数：

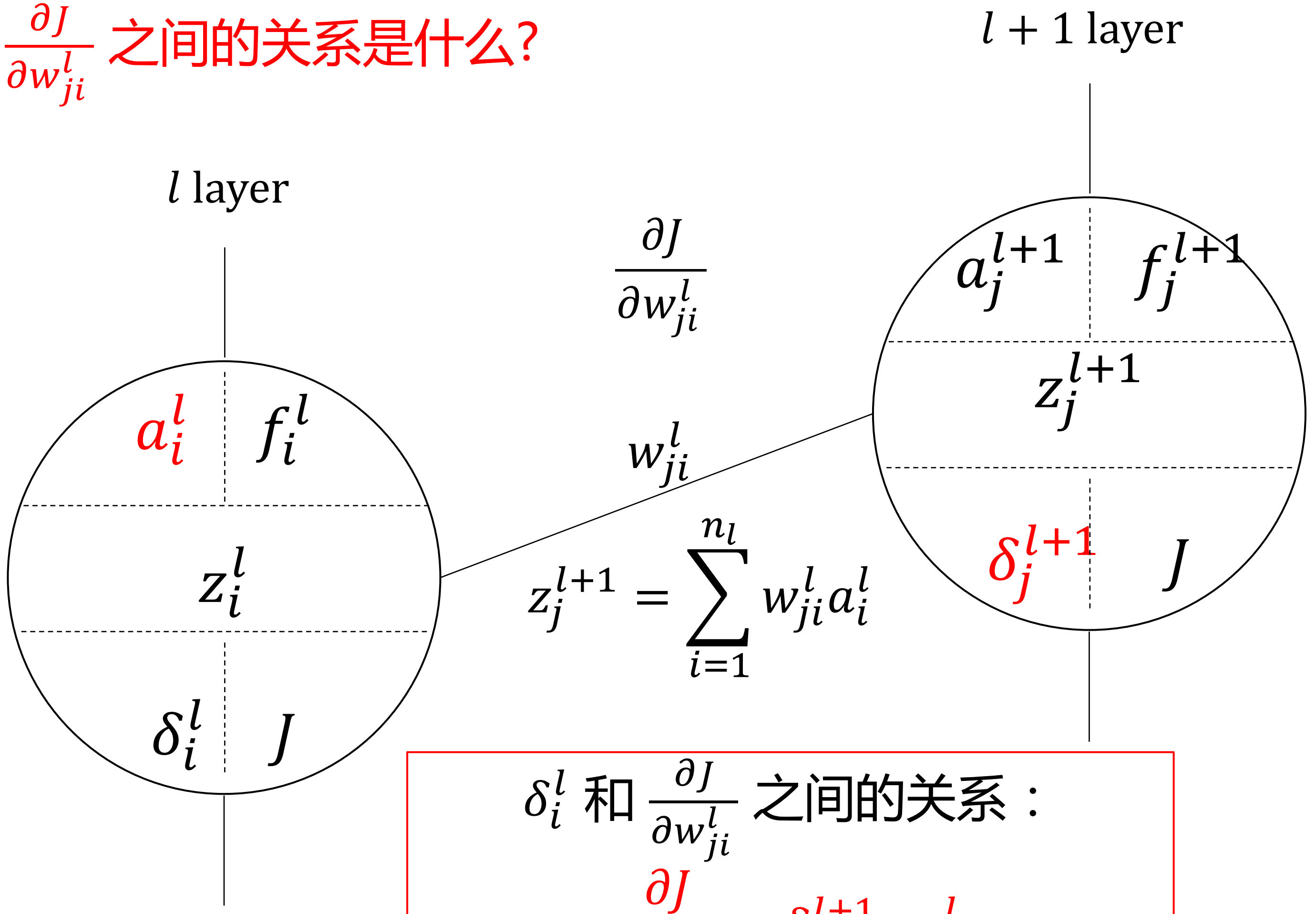
$$J = \frac{1}{2}\sum_{j=1}^{n_L} e_j^2 = \frac{1}{2}\sum_{j=1}^{n_L}(a_j^L - y_j^L)^2$$


定义在神经元上的局部激活函数
$a_i^l = f_i^l(z_i^l)$
a_i^l f_i^l z_i^l
$a^l = \begin{bmatrix} a_1^l \\ a_2^l \\ \vdots \\ a_{n_l}^l \end{bmatrix}$
第 l 层的第 i 个神经元
局部激活函数 f_i^l
前向计算 a^l
a_i^l f_i^l z_i^l δ_i^l J
第 l 层的第 i 个神经元 $\delta_i^l = \dfrac{\partial J}{\partial z_i^l}$
z_i^l δ_i^l J
$\delta^l = \begin{bmatrix} \delta_1^l \\ \delta_2^l \\ \vdots \\ \delta_{n_l}^l \end{bmatrix}$
反向传播 δ^l
全局性能函数 J
定义在整个网络上的性能函数

问题 1:

δ_i^l 和 $\frac{\partial J}{\partial w_{ji}^l}$ 之间的关系是什么?



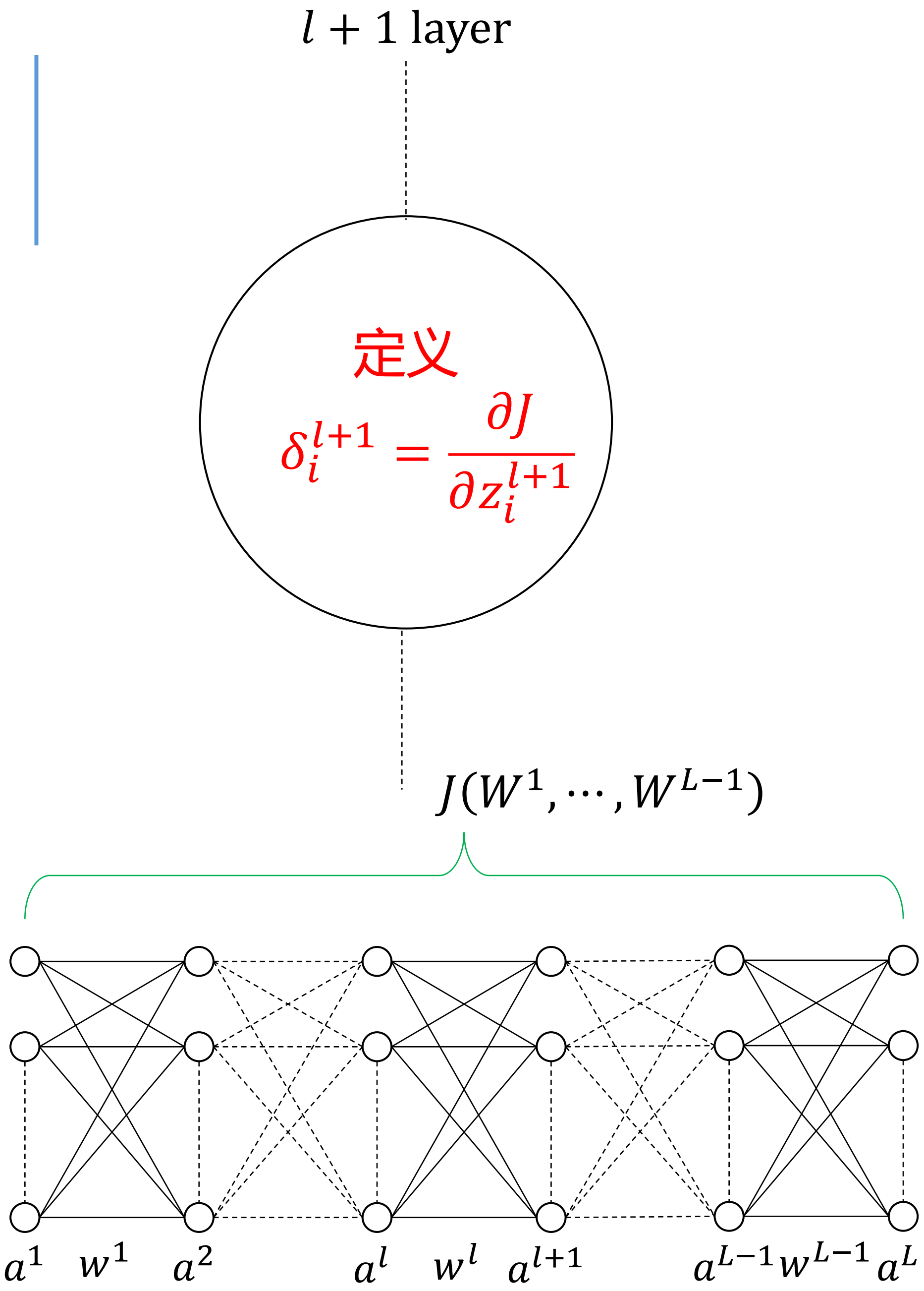


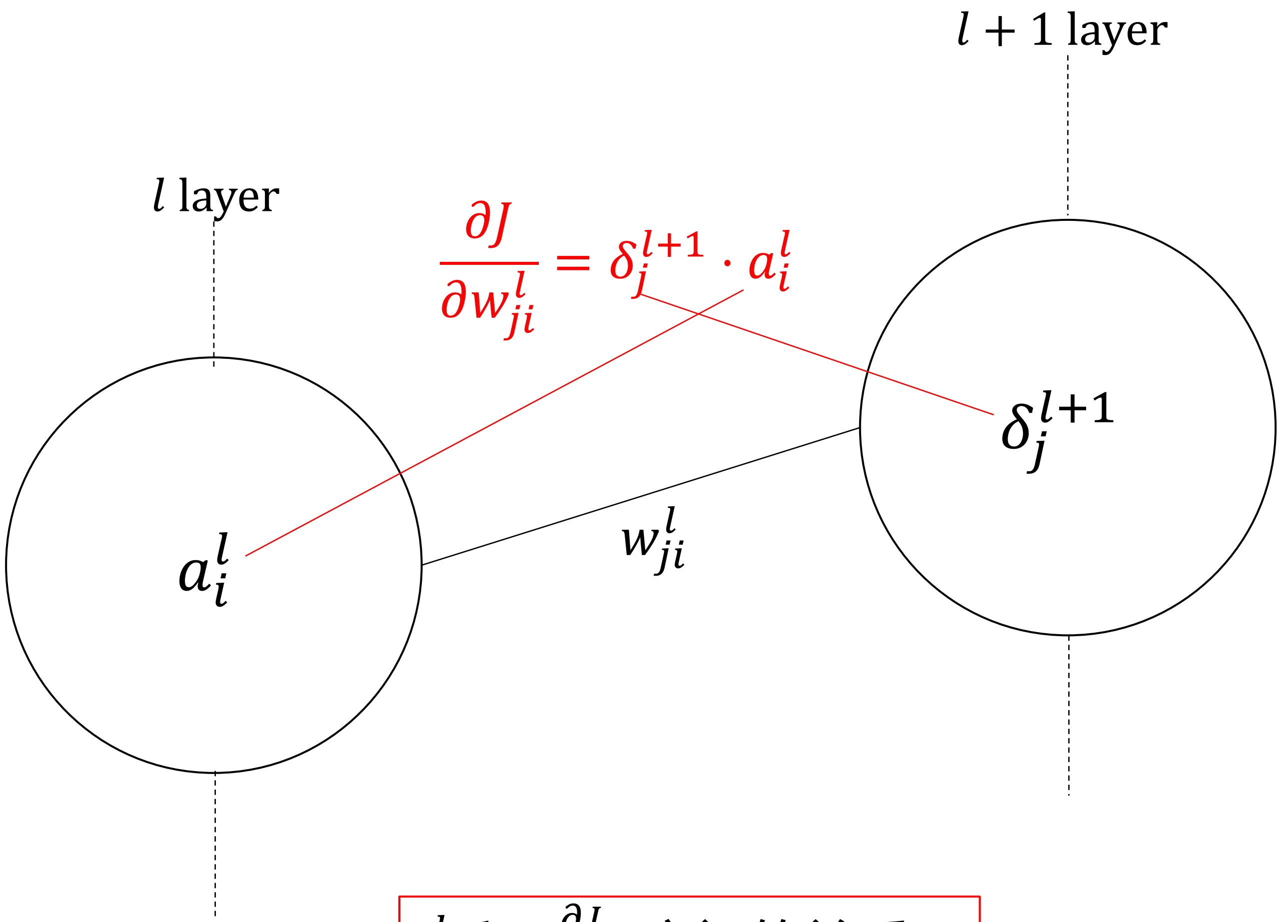
δ_i^l 和 $\frac{\partial J}{\partial w_{ji}^l}$ 之间的关系：

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

为什么?

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

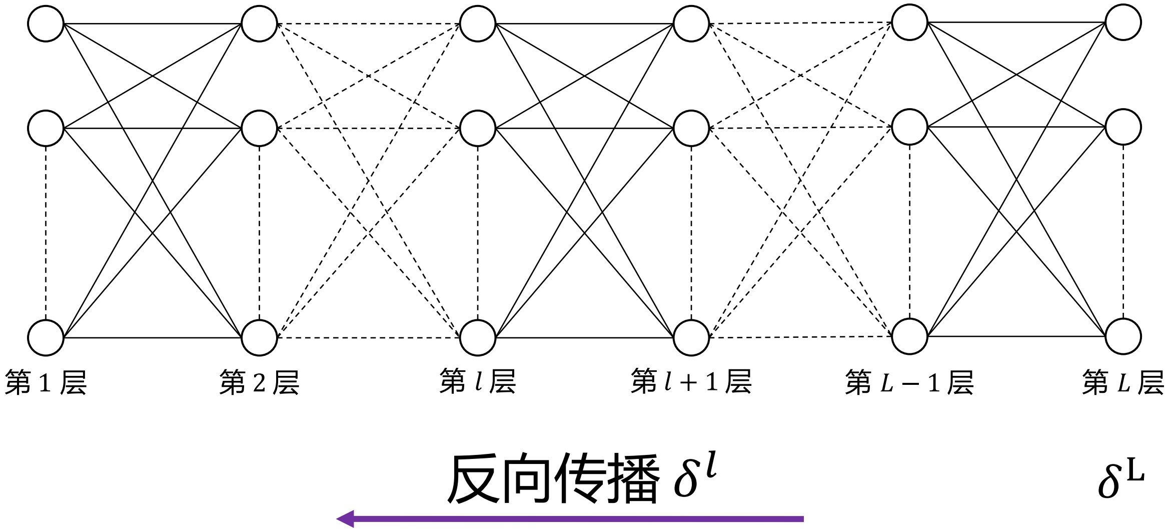




δ_i^l 和 $\frac{\partial J}{\partial w_{ji}^l}$ 之间的关系：

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

问题 2：
如何计算最后一层的 δ_i^L ？



按照定义

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

假设

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

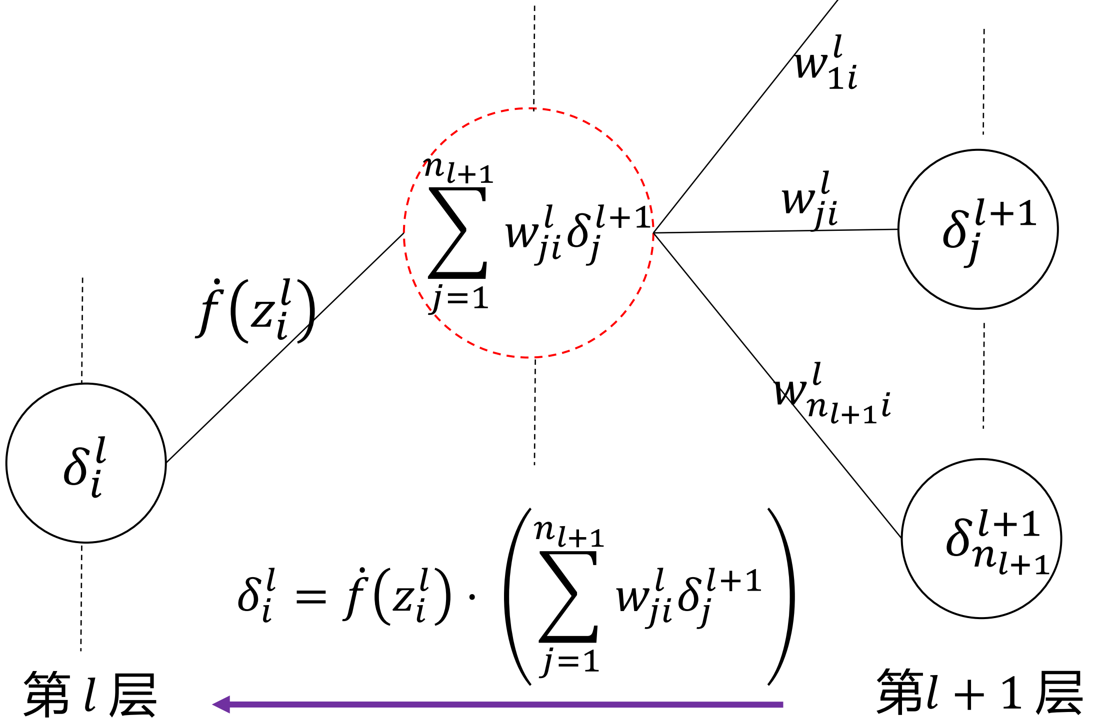
则

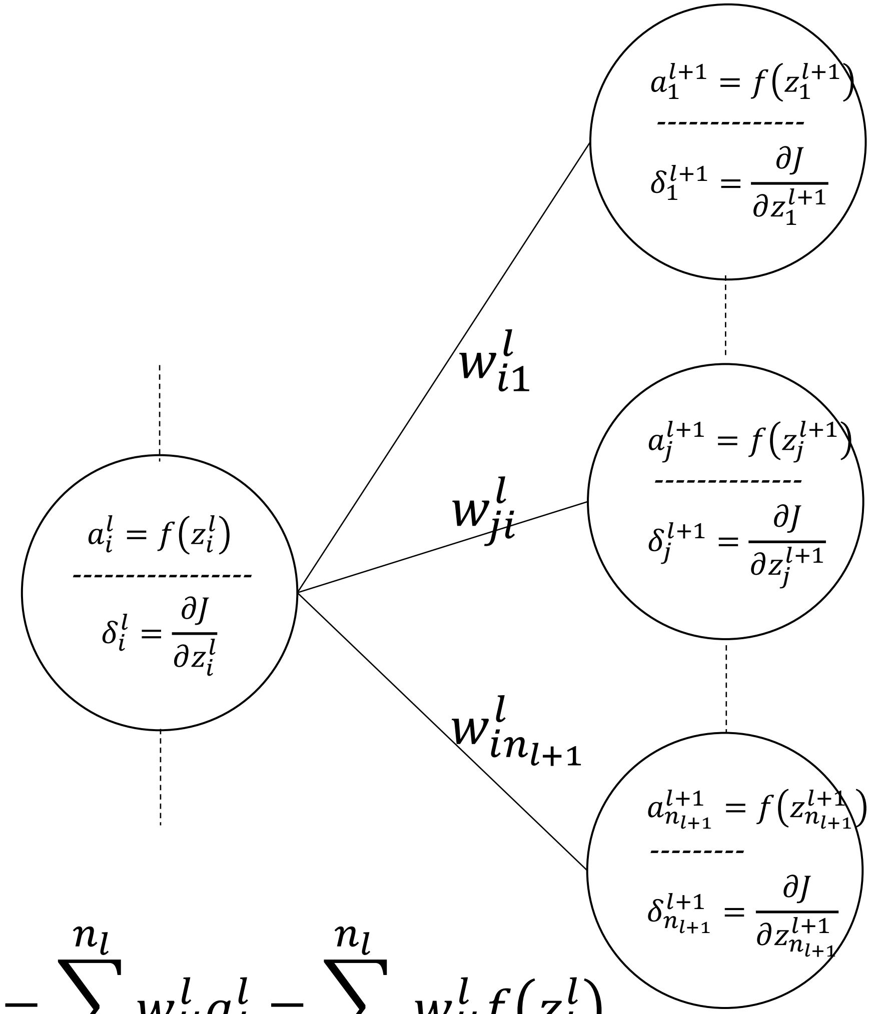
$$\delta_i^L = \frac{\partial J}{\partial z_i^L} = \left(a_i^L - y_i^L\right)\cdot\frac{\partial a_i^L}{\partial z_i^L} = \left(a_i^L - y_i^L\right)\cdot\dot{f}\left(z_i^L\right)$$

$$a_i^L = f\left(z_i^L\right)$$

第 L 层的第 i 个神经元

问题 3:
δ_i^l 和 δ_j^{l+1} 之间的关系





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

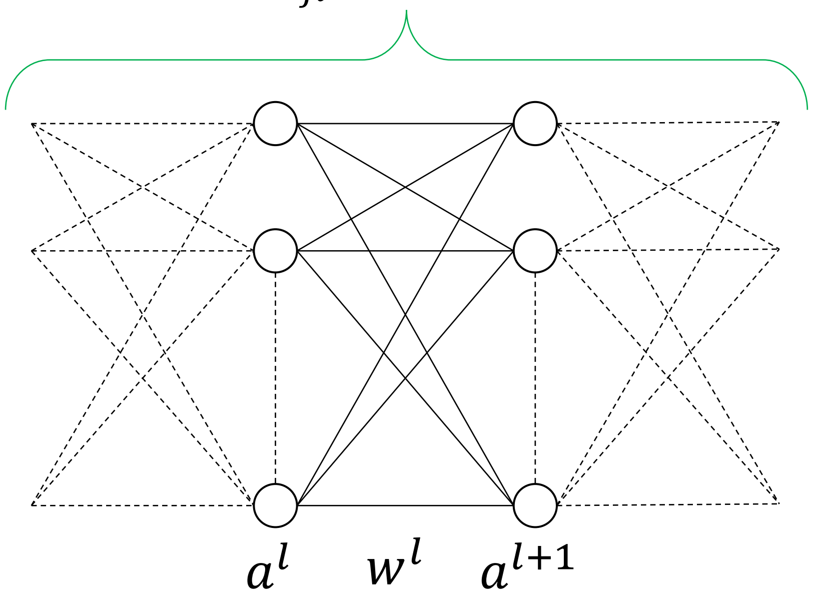
$$\delta_i^l = \frac{\partial J}{\partial z_i^l} = \sum_{j=1}^{n_{l+1}} \frac{\partial J}{\partial z_j^{l+1}} \cdot \frac{\partial z_j^{l+1}}{\partial z_i^l} = \sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot \frac{\partial z_j^{l+1}}{\partial z_i^l} = \sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot w_{ji}^l \dot{f}(z_i^l)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

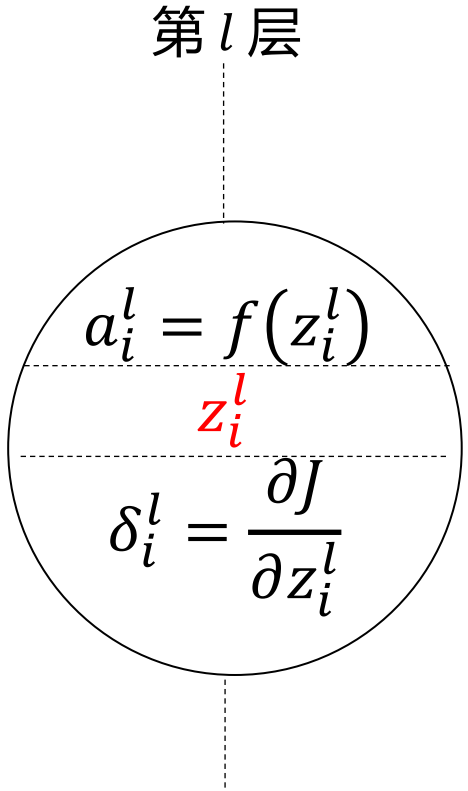
仅3页ppt理解反向传播：第1页

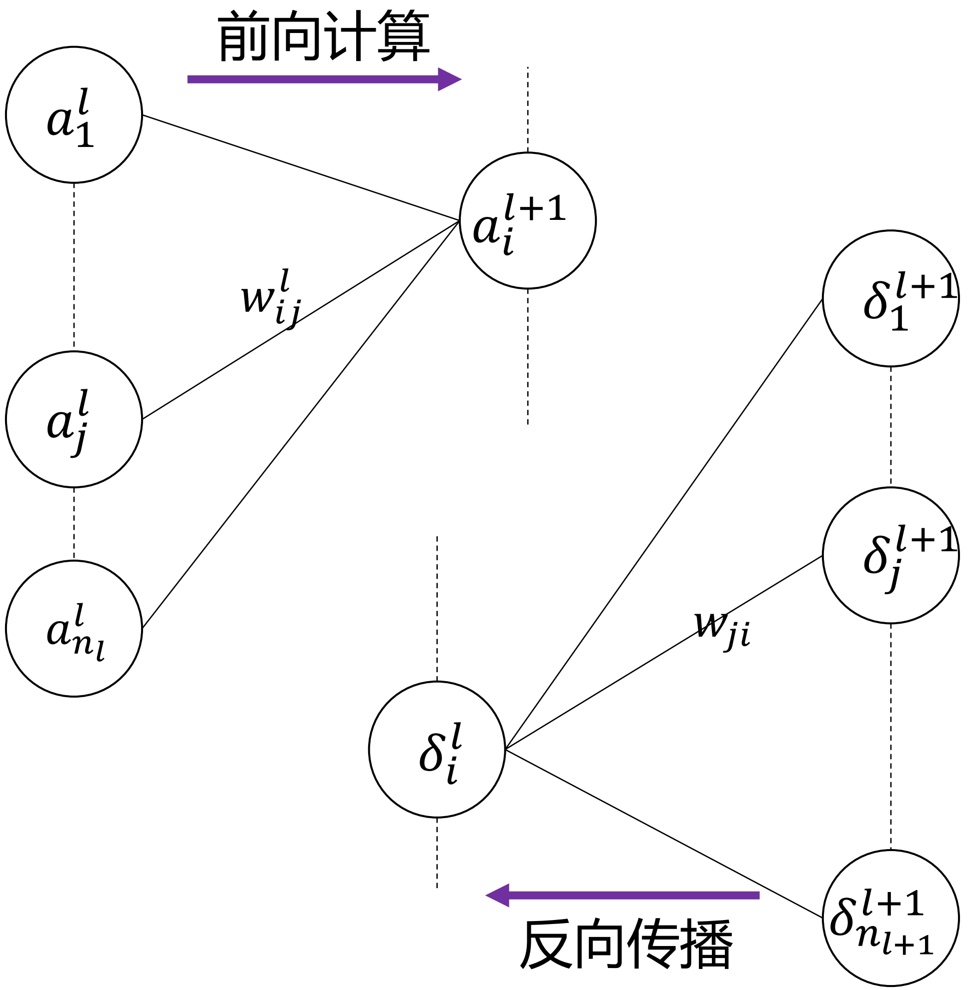
性能函数：$J(w^1,\cdots,w^L)$

权值更新规则：$w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$

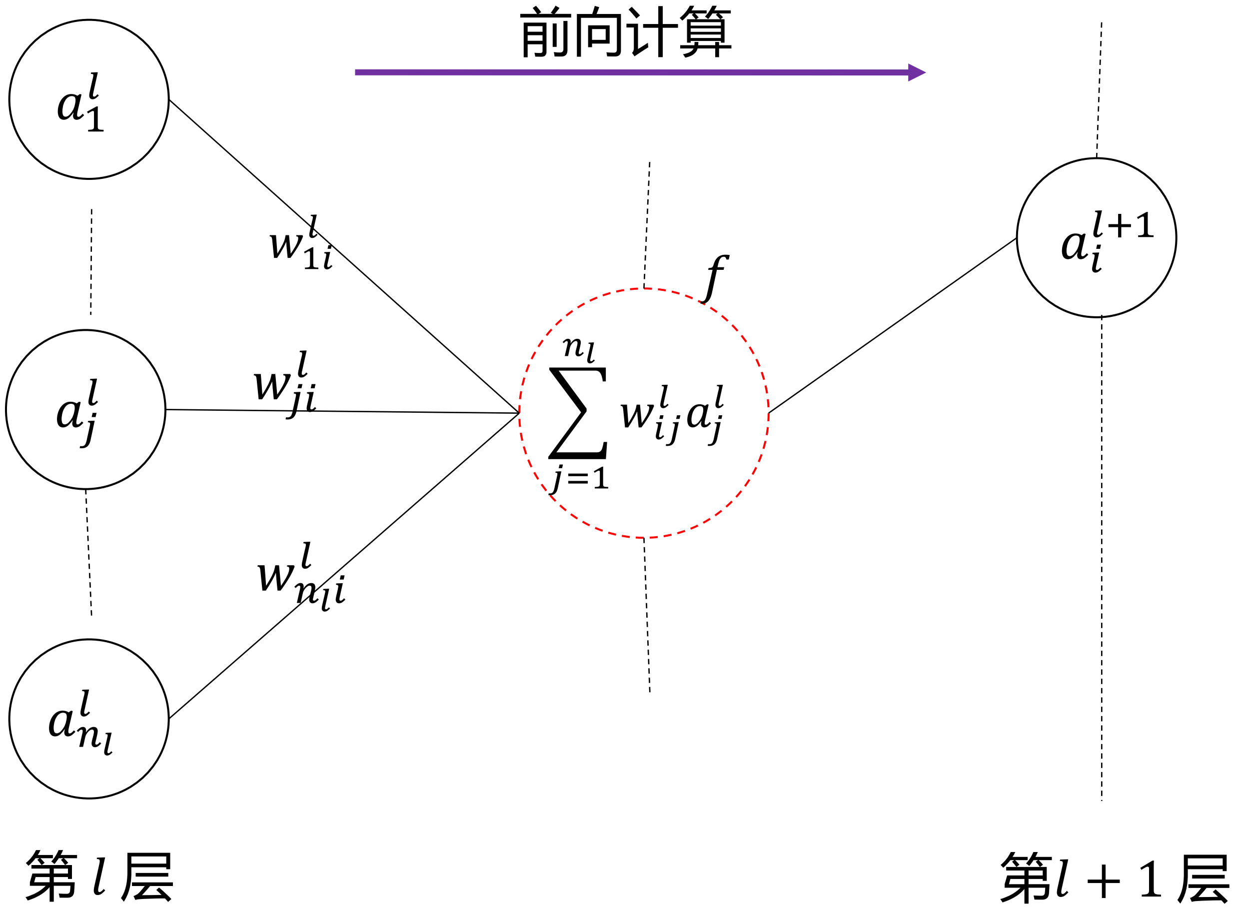
关系：$\frac{\partial J}{\partial w_{ji}^l} = \delta_j^{l+1} \cdot a_i^l$







仅3页ppt理解反向传播：第2页



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

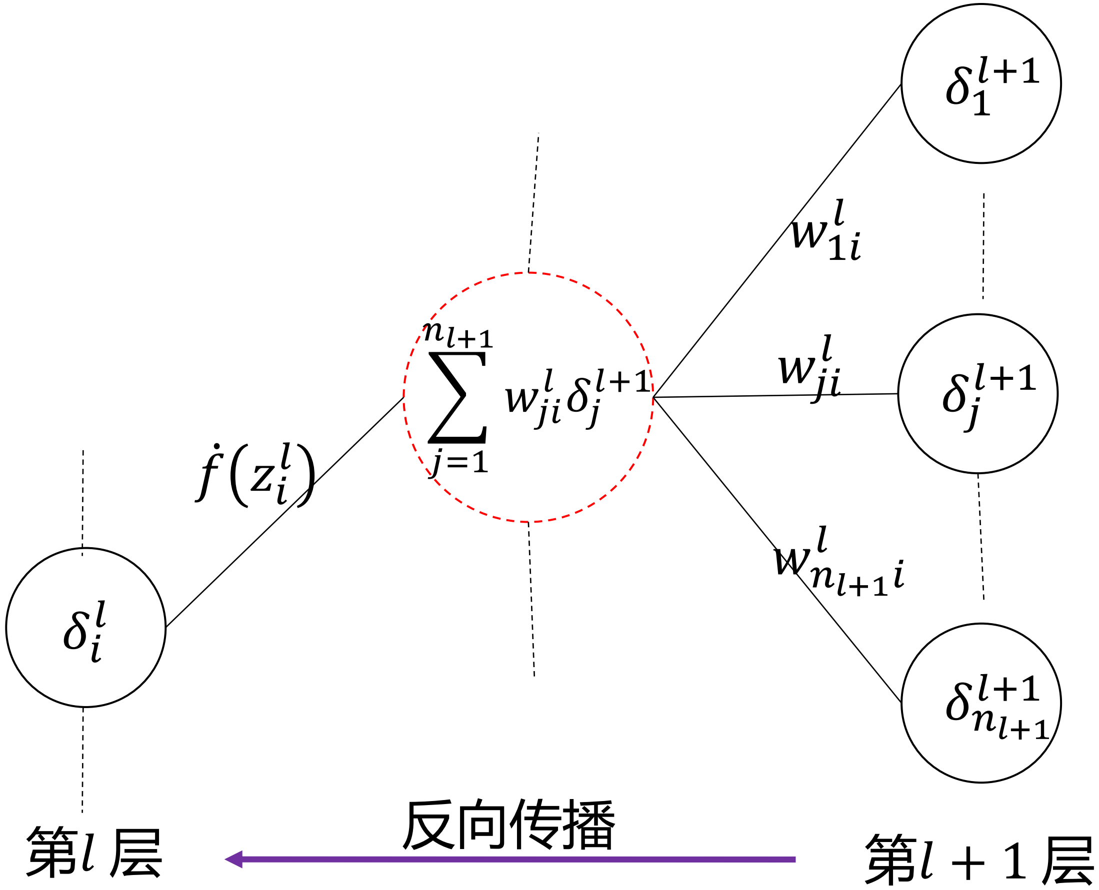
或--------------------

$$a_i^{l+1} = f\left(z_i^{l+1}\right)$$

$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l$$

仅3页ppt理解反向传播：第3页

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

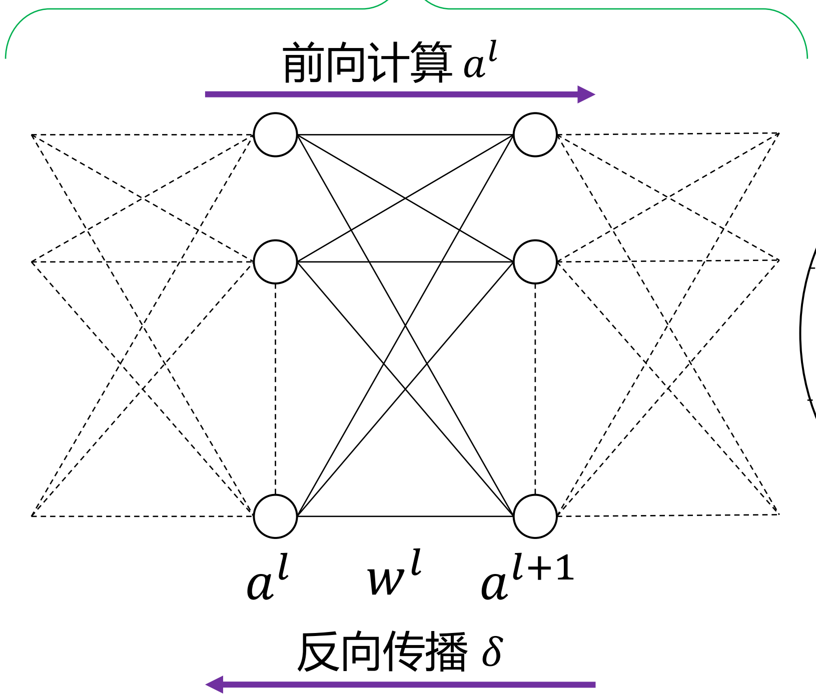


仅1页ppt理解反向传播

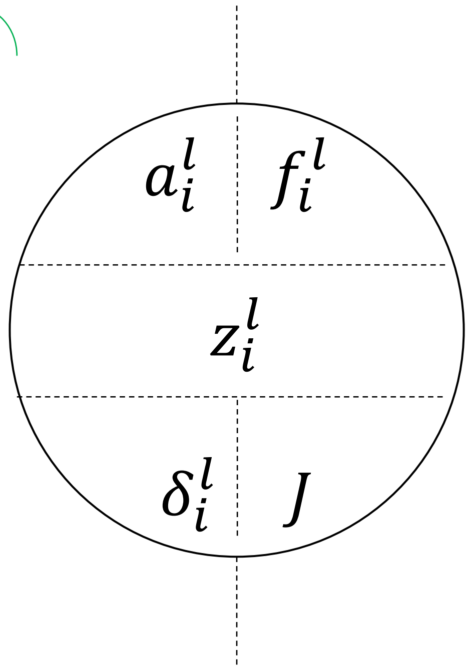
性能函数：$J(w^1,\cdots,w^L)$

权值更新规则：$w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$

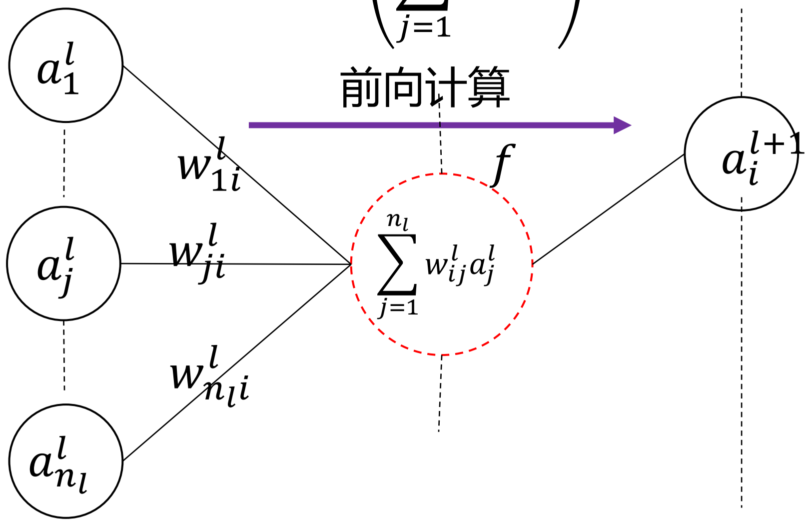
关系：$\frac{\partial J}{\partial w_{ji}^l} = \delta_j^{l+1} \cdot a_i^l$

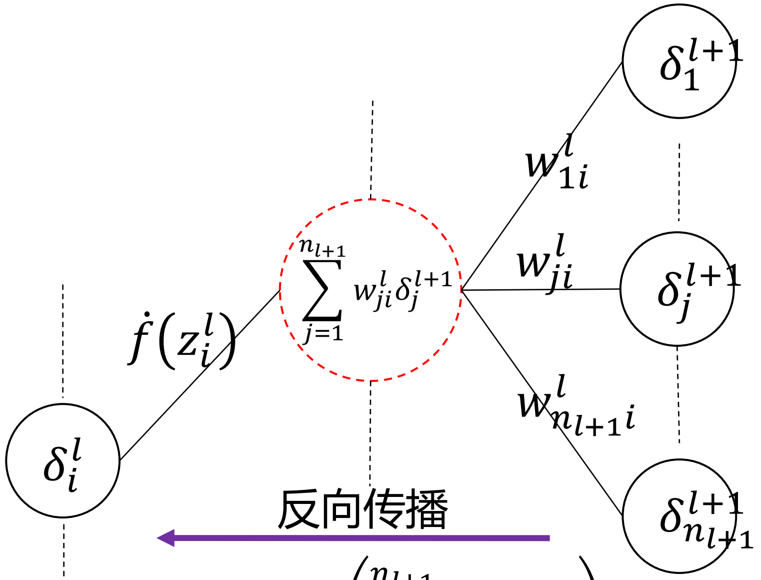


第 l 层的第 i 个神经元



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



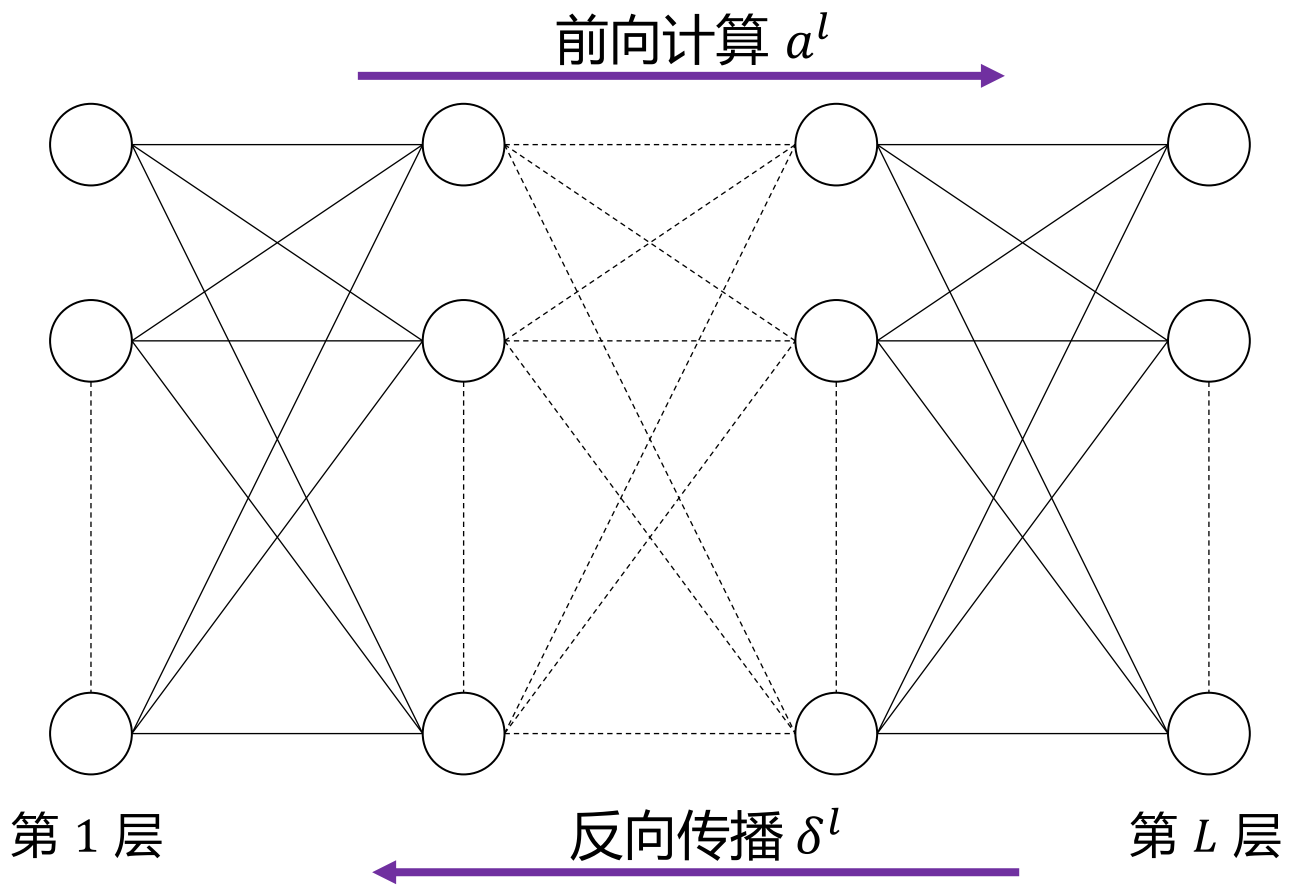


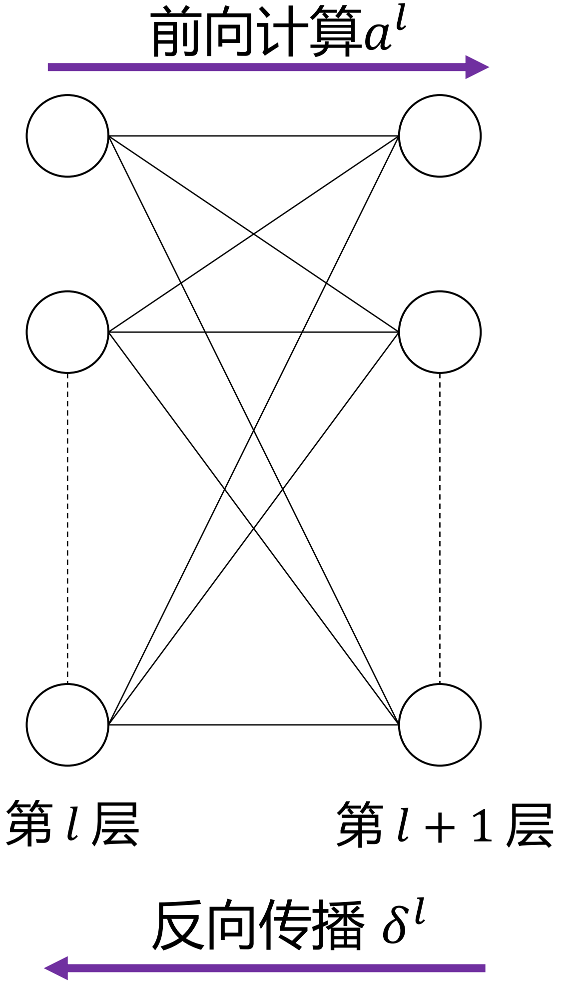
第 l 层 $\delta_i^l = \dot{f}(z_i^l) \cdot \left(\sum_{j=1}^{n_{l+1}} w_{ji}^l \delta_j^{l+1}\right)$ 第$l+1$层

大纲

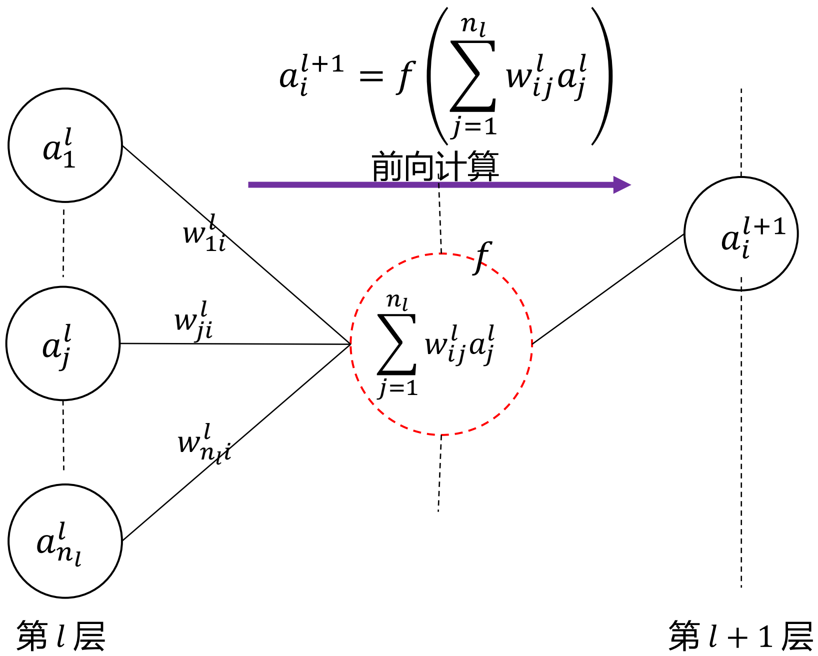
- 简短回顾：神经网络的可计算模型
- 网络性能刻画：性能函数
- 最速梯度下降法
- 反向传播原理
- 反向传播算法

反向传播算法





反向传播算法



function $fc(W^l, a^l)$

$for\ i = 1{:}\,n_{l+1}$

$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l$$

$$a_i^{l+1} = f\left(z_i^{l+1}\right)$$

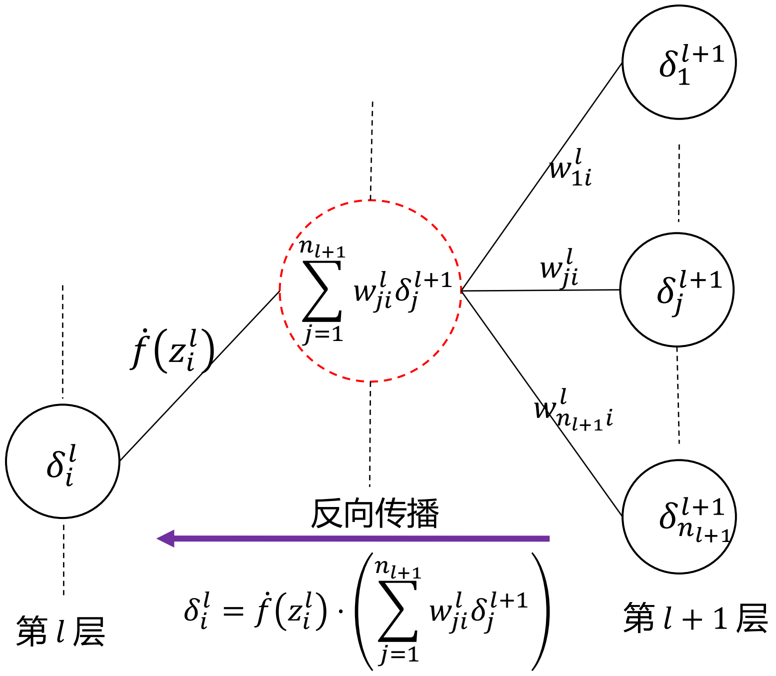
end

function $bc(W^l, \delta^{l+1})$

$for\ i = 1{:}\,n_l$

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

end



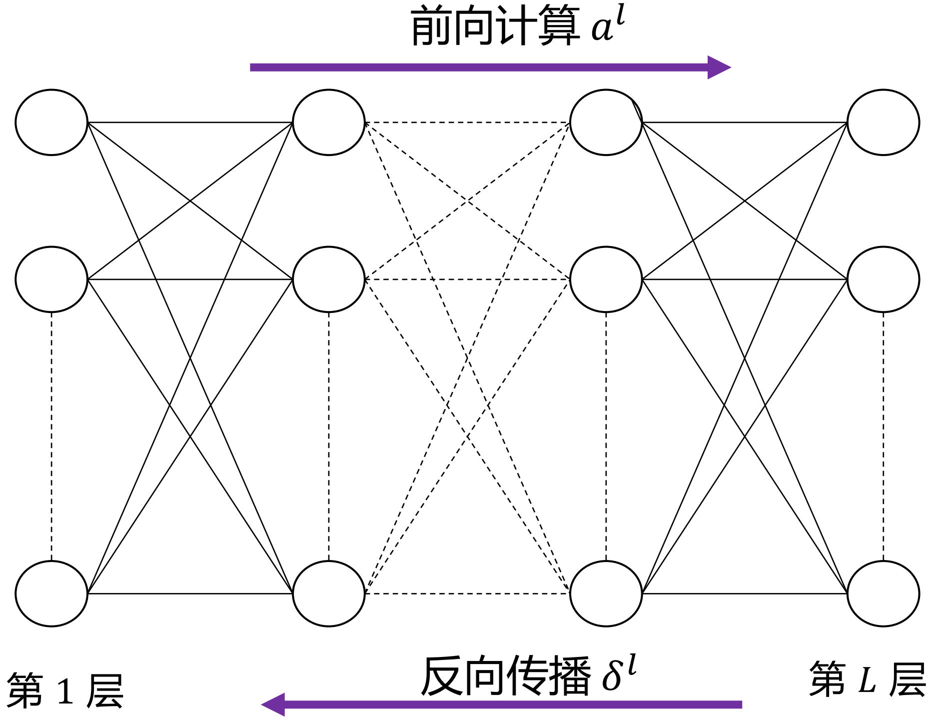
反向传播算法

训练数据集 $D = \{(x, y^L)\}$

一条训练样本 (x, y^L)

性能函数/代价函数 $J(x)$

x: 输入样本
y^L: 目标输出



反向传播算法

在线BP算法

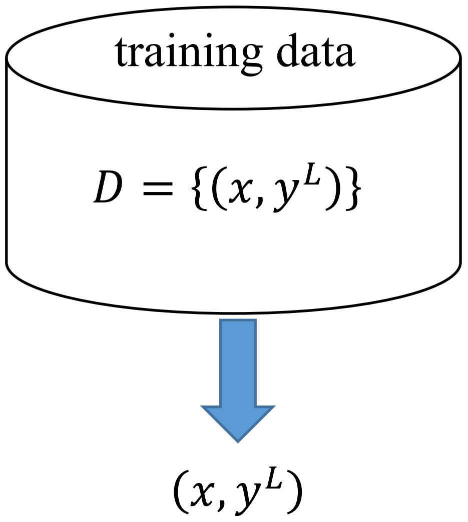
Step 1. Input the training data set $D = \{(x, y^L)\}$

Step 2. Initialize each w_{ij}^l, and choose a learning rate α

Step 3. for each $(x, y^L) \in D$

$$
\begin{aligned}
& a^1 \leftarrow x \in D_m; \\
& \text{for } l = 2{:}\,L \\
& \quad a^{l+1} \leftarrow fc\left(w^l, a^l\right); \\
& \text{end} \\
& J(x, y^L); \\
& \delta^L = \frac{\partial J(x, y^L)}{\partial z^L}; \\
& \text{for } l = L-1{:}\,2 \\
& \quad \delta^l \leftarrow bc\left(w^l, \delta^{l+1}\right); \\
& \text{end} \\
& \nabla J_{ji} \leftarrow \delta_j^{l+1} \cdot a_i^l; \\
& w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \nabla J_{ji}
\end{aligned}
$$

Step 4. Return to Step 3 until each w^lconverge



function $fc\left(w^l, a^l\right)$

$$
\begin{aligned}
& for\ i = 1{:}\,n_{l+1} \\
& \qquad z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l \\
& \qquad a_i^{l+1} = f\left(z_i^{l+1}\right) \\
& end
\end{aligned}
$$

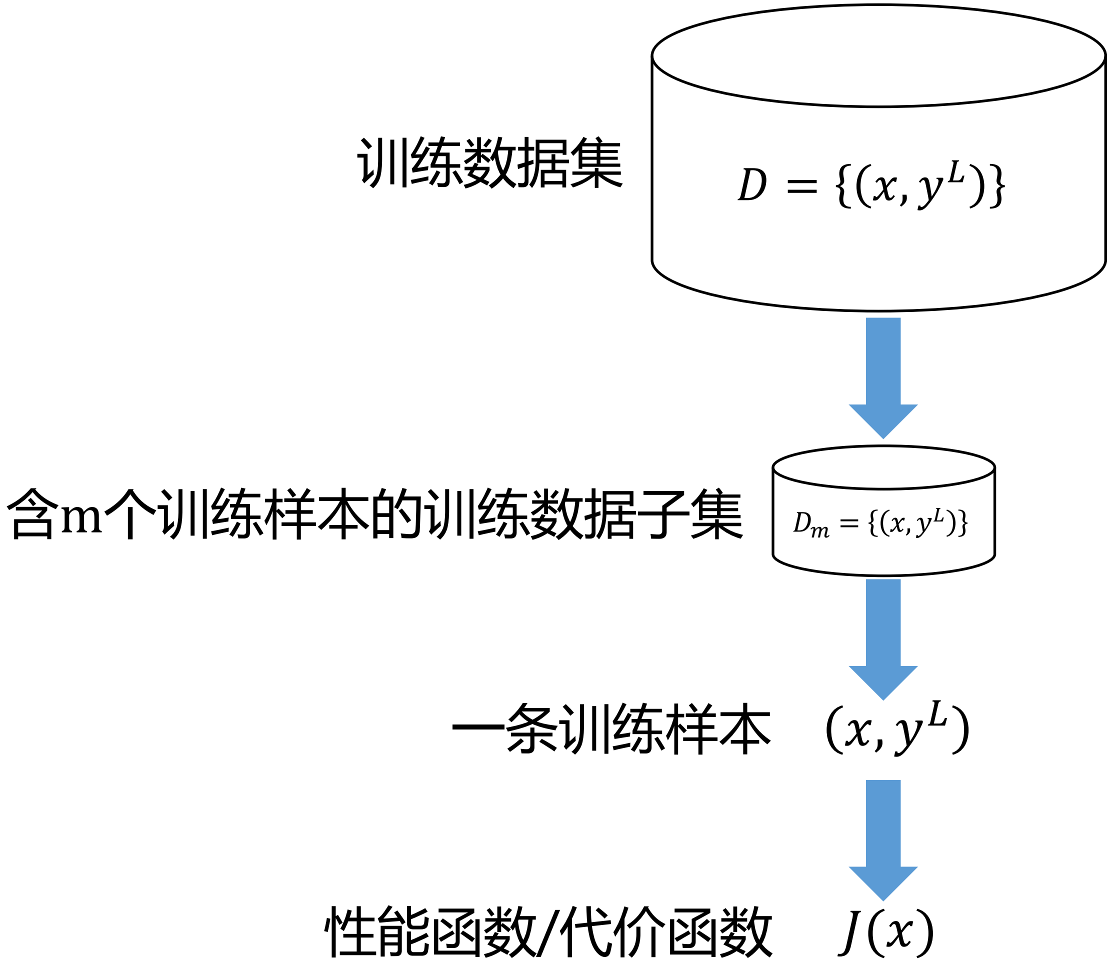
Relationship:

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

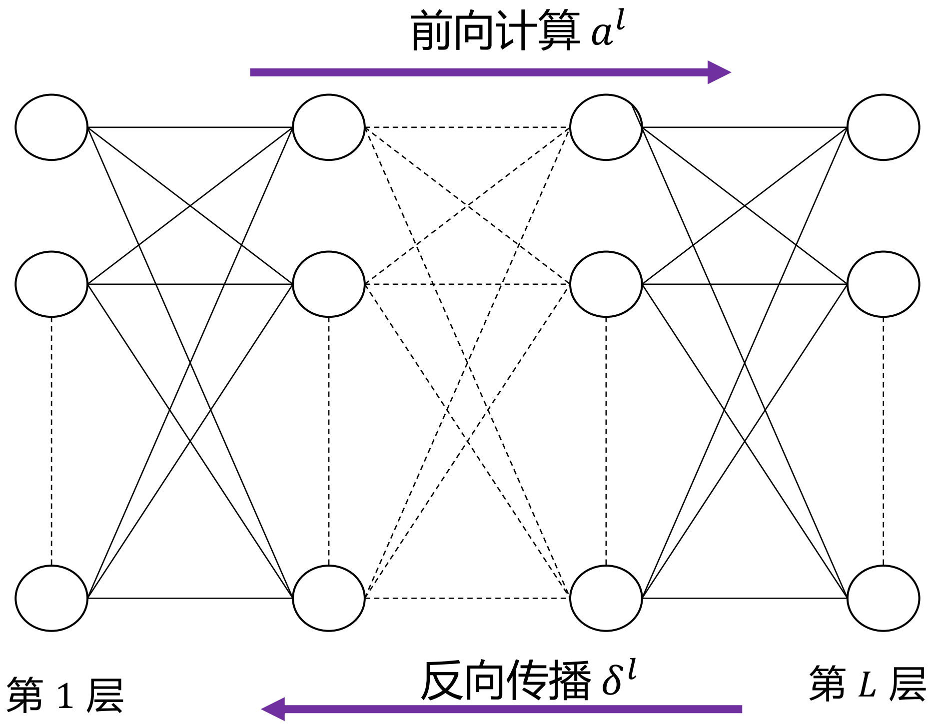
function $bc\left(w^l, \delta^{l+1}\right)$

$$
\begin{aligned}
& for\ i = 1{:}\,n_l \\
& \qquad \delta_i^l = \dot{f}\left(z_i^l\right) \cdot \left(\sum_{j=1}^{n_{l+1}} w_{ji}^l \delta_j^{l+1}\right) \\
& end
\end{aligned}
$$

反向传播算法



x: 输入样本
y^L: 目标输出



反向传播算法

批处理BP算法

Step 1. Input the training data set $D = \{(x, y^L)\}$

Step 2. Initialize each w_{ij}^l, and choose a learning rate α

Step 3. for each mini-batch sample $D_m \subseteq D$

$\nabla J_{ji} = 0;$

for each $(x, y^L) \in D_m$

$a^1 \leftarrow x \in D_m;$

for $l = 2 : L$

$a^{l+1} \leftarrow fc(w^l, a^l);$

end

$J(x, y^L);$

$$\delta^L = \frac{\partial J(x, y^L)}{\partial z^L};$$

for $l = L - 1 : 2$

$\delta^l \leftarrow bc(w^l, \delta^{l+1});$

end

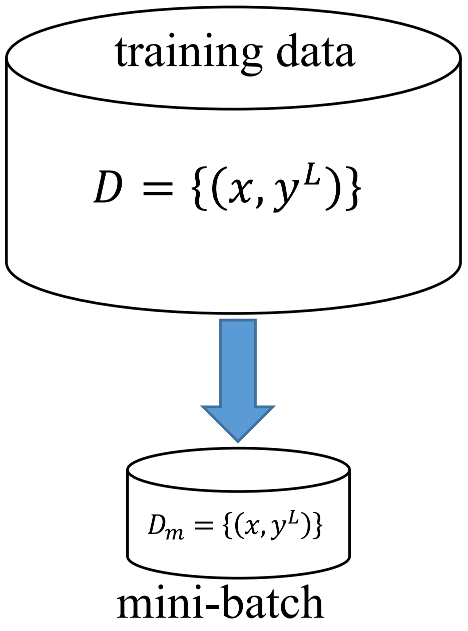
$\nabla J_{ji} \leftarrow \nabla J_{ji} + \delta_j^{l+1} \cdot a_i^l;$

end

$w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \nabla J_{ji};$

end

Step 4. Return to Step 3 until each w^lconverge



function $fc(w^l, a^l)$

$for\ i = 1 : n_{l+1}$

$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l$$

$$a_i^{l+1} = f(z_i^{l+1})$$

end

Relationship:

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

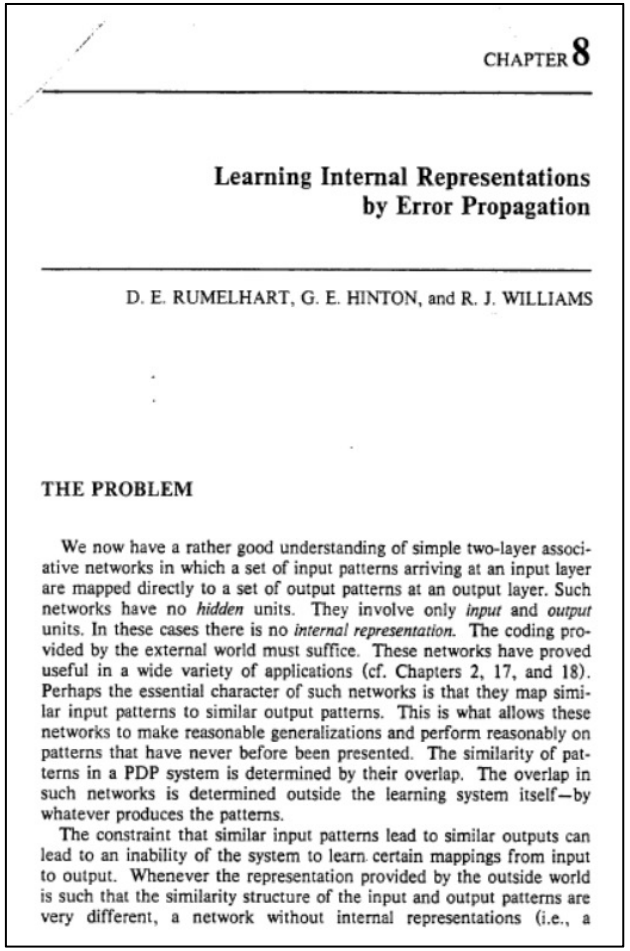
function $bc(w^l, \delta^{l+1})$

$for\ i = 1 : n_l$

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

end

反向传播（BP）算法简史



CHAPTER 8

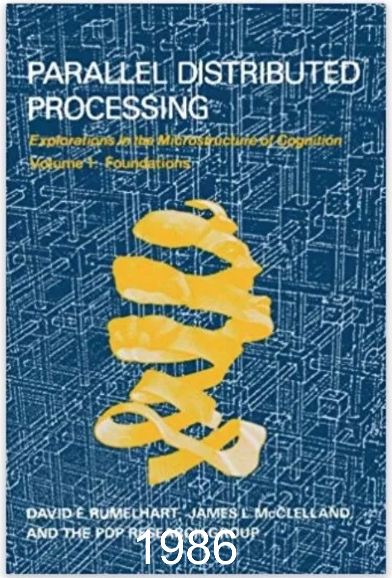
Learning Internal Representations by Error Propagation

D. E. RUMELHART, G. E. HINTON, and R. J. WILLIAMS

THE PROBLEM

We now have a rather good understanding of simple two-layer associative networks in which a set of input patterns arriving at an input layer are mapped directly to a set of output patterns at an output layer. Such networks have no *hidden* units. They involve only *input* and *output* units. In these cases there is no *internal representation*. The coding provided by the external world must suffice. These networks have proved useful in a wide variety of applications (cf. Chapters 2, 17, and 18). Perhaps the essential character of such networks is that they map similar input patterns to similar output patterns. This is what allows these networks to make reasonable generalizations and perform reasonably on patterns that have never before been presented. The similarity of patterns in a PDP system is determined by their overlap. The overlap in such networks is determined outside the learning system itself—by whatever produces the patterns.

The constraint that similar input patterns lead to similar outputs can lead to an inability of the system to learn certain mappings from input to output. Whenever the representation provided by the outside world is such that the similarity structure of the input and output patterns are very different, a network without internal representations (i.e., a

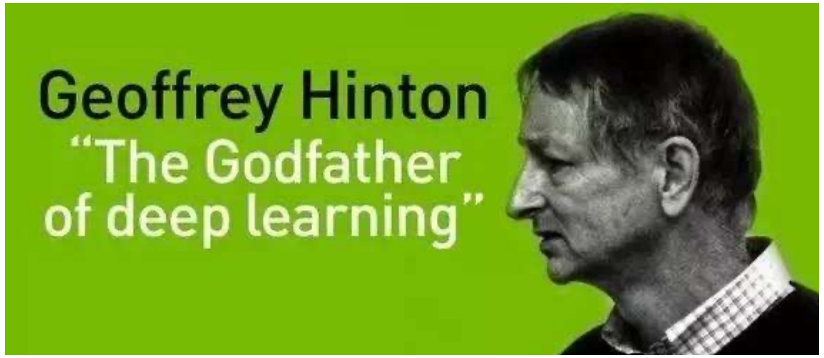


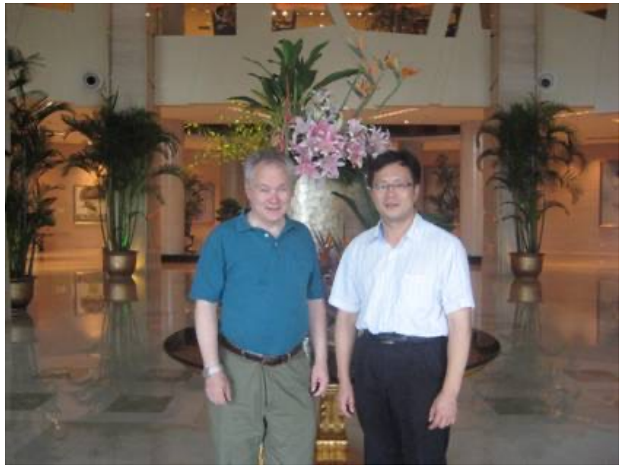
1986

人工智能

深度神经网络

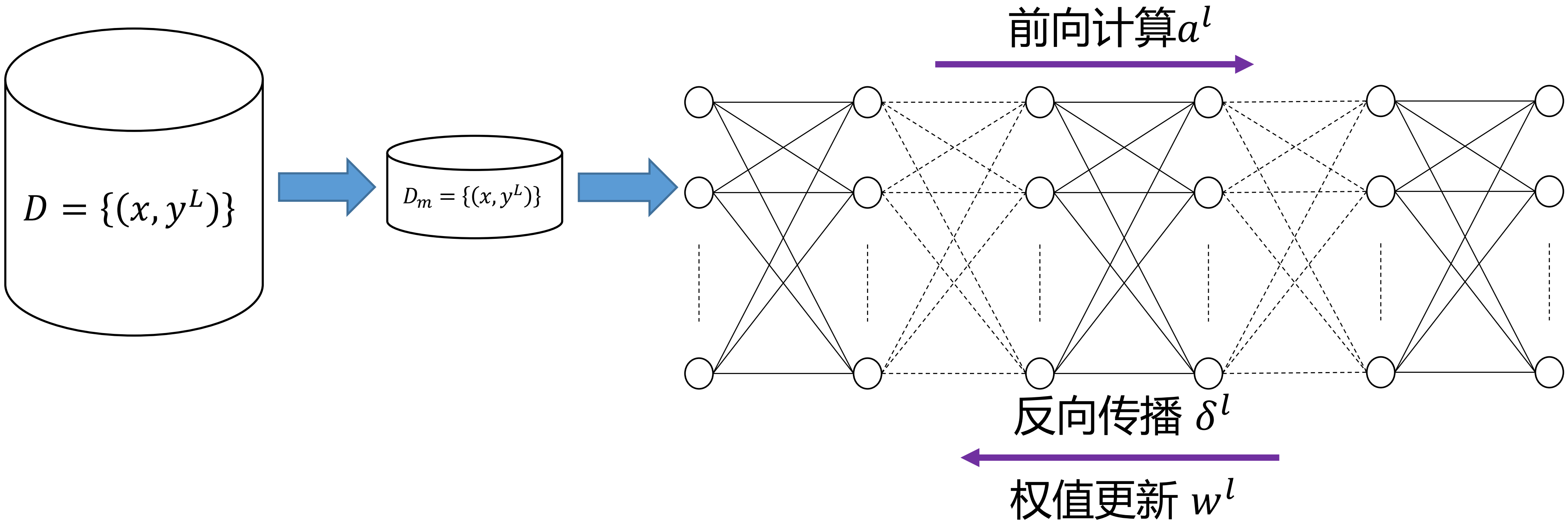
BP 学习算法



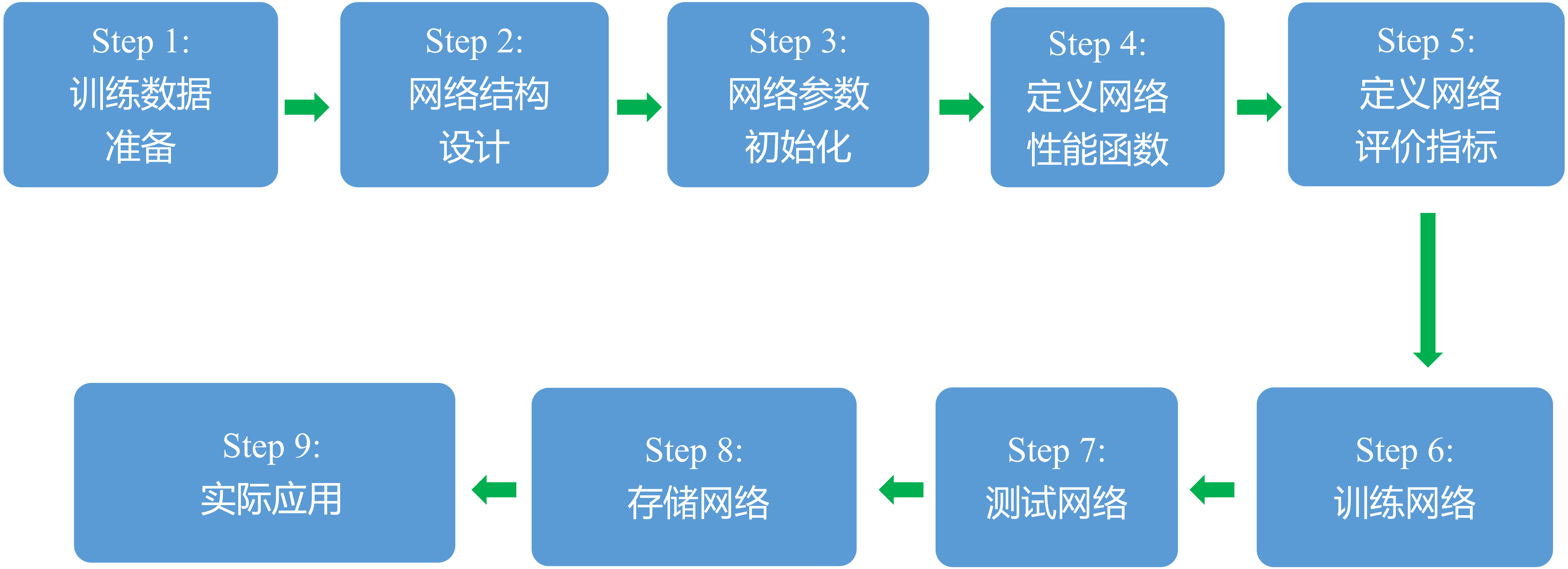


P. Werbos 教授

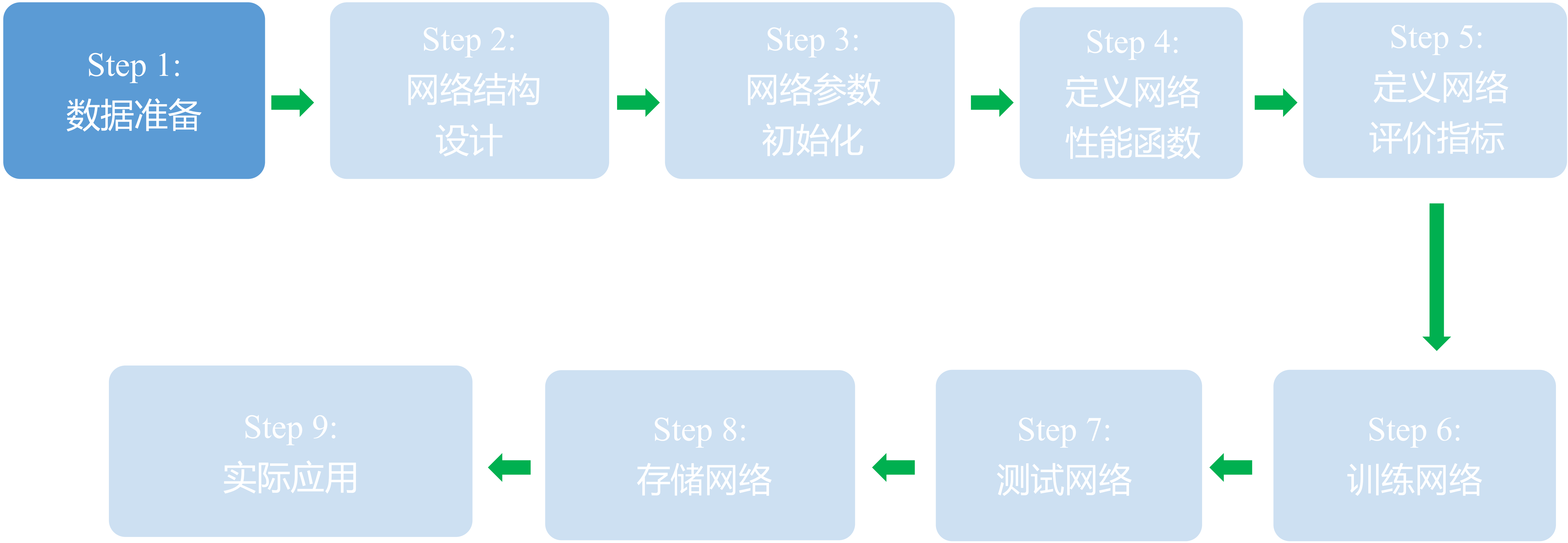
网络训练



网络训练步骤


Step 1:
训练数据
准备
Step 2:
网络结构
设计
Step 3:
网络参数
初始化
Step 4:
定义网络
性能函数
Step 5:
定义网络
评价指标
Step 6:
训练网络
Step 7:
测试网络
Step 8:
存储网络
Step 9:
实际应用

Step 1: 数据准备



Step 1:
数据准备
Step 2:
网络结构设计
Step 3:
网络参数初始化
Step 4:
定义网络性能函数
Step 5:
定义网络评价指标
Step 6:
训练网络
Step 7:
测试网络
Step 8:
存储网络
Step 9:
实际应用

Step 1: 数据准备

原始数据集

$D = \{(x, y^L)\}$

原图

$D_{train} = \{(x, y^L)\}$

训练数据集

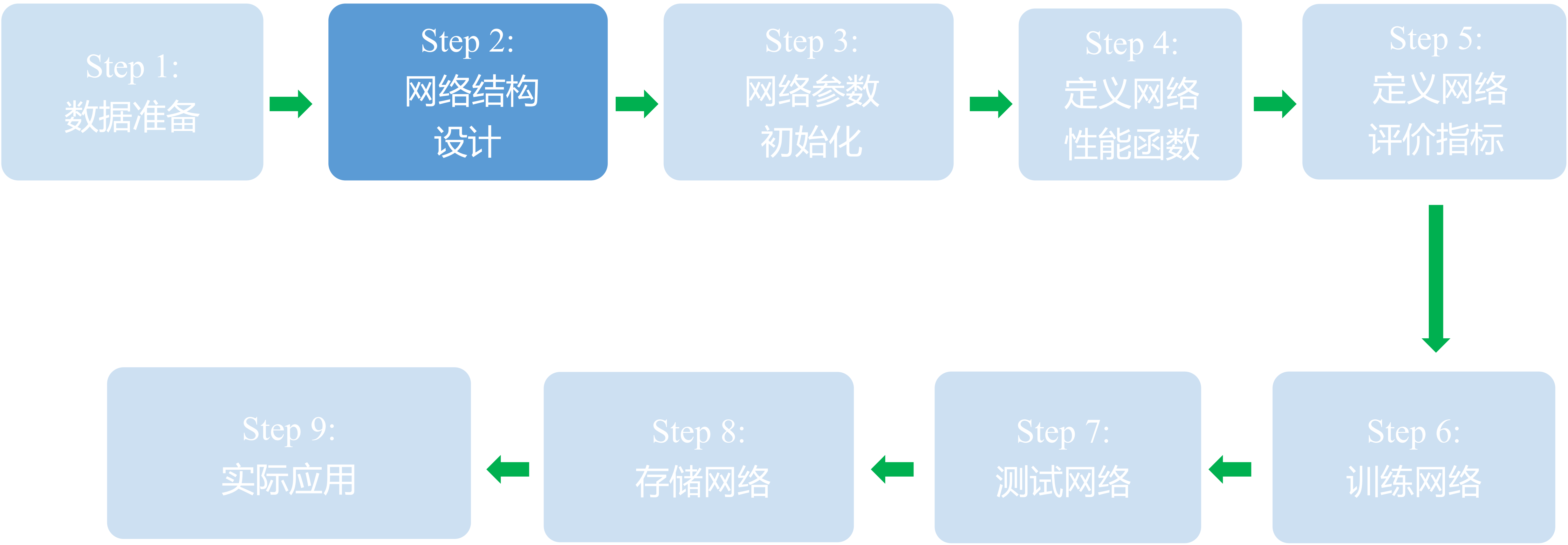
相对大一些
e.g. 80%

$D_{test} = \{(x, y^L)\}$

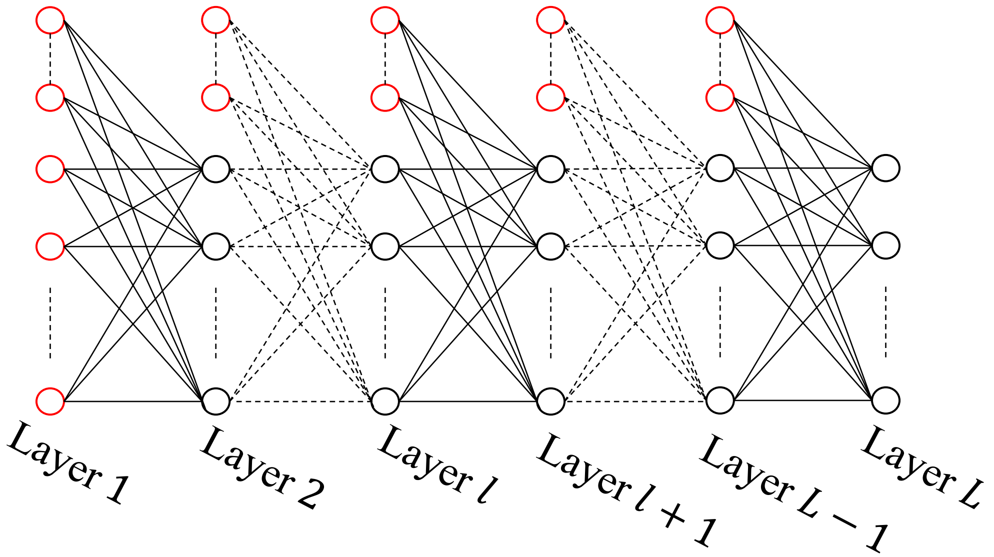
测试数据集

相对小一些
e.g. 20%

Step 2:网络结构设计



Step 2: 网络结构设计

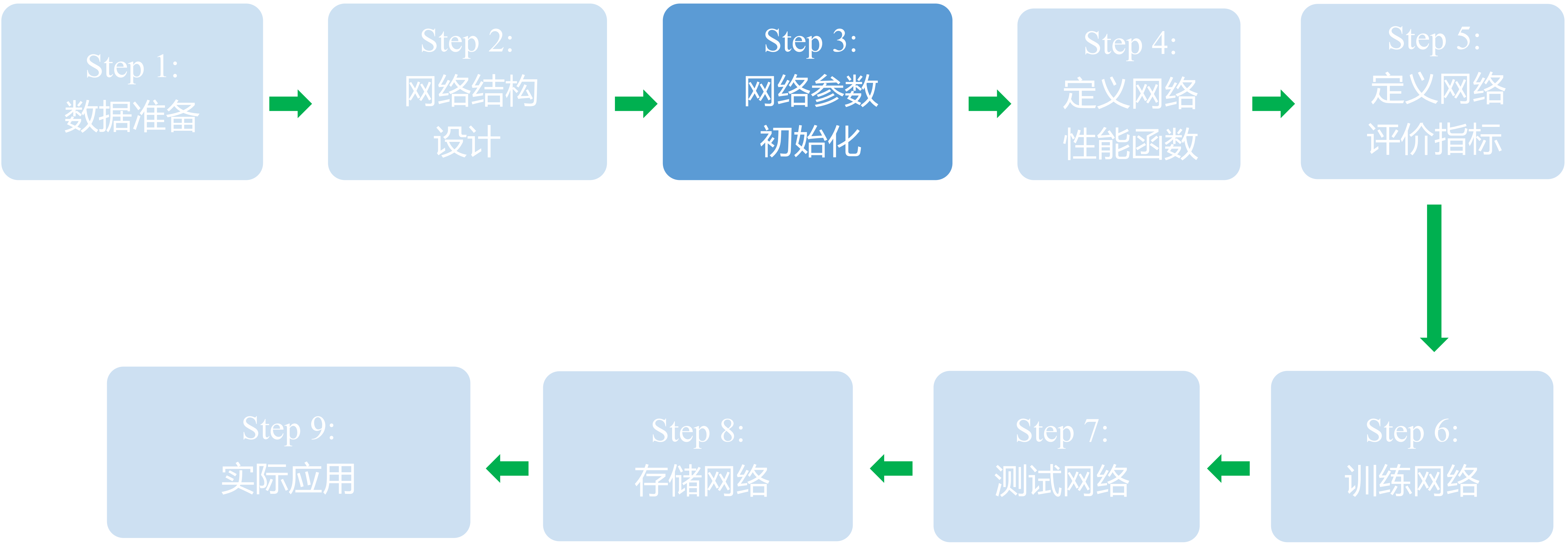


L=?

$(L \geqq 3)$

- 设计网络层数
- 设计每层神经元数目
  - 输入神经元数量
  - 表达神经元数量
- 设计激活函数

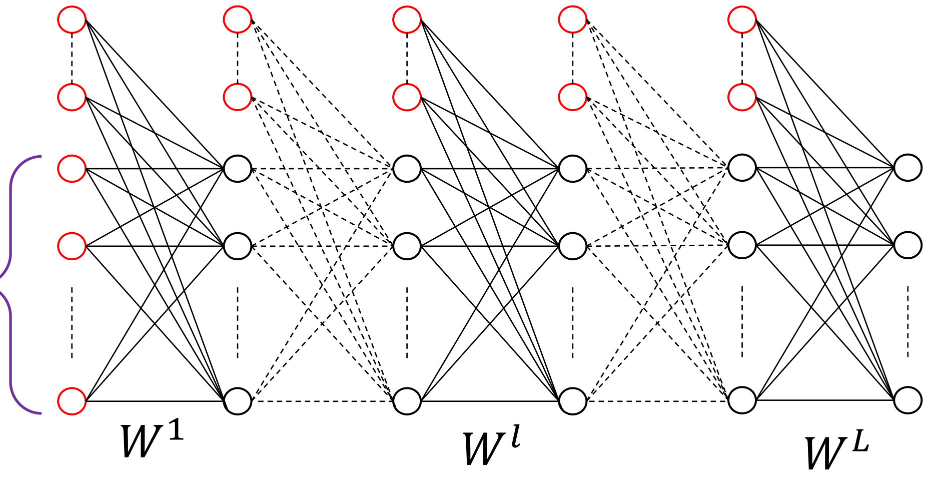
Step 3: 初始化网络参数



Step 3: 初始化网络参数

- 连接权矩阵
- 初始外部输入
- 学习率

- 初始外部输入



- 连接权矩阵

$$w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$$

- 学习率

$$W^l = \begin{bmatrix} w_{11}^l & \cdots & w_{1n_{l+1}}^l \\ \cdots & w_{ij}^l & \cdots \\ w_{n_l 1}^l & \cdots & w_{n_l \times n_{l+1}}^l \end{bmatrix}_{n_l \times n_{l+1}}$$

$$W^l = \begin{bmatrix} 0.7267, 0.0485, 0.8780 \\ 0.4068, 0.2969, 0.5010 \\ 0.8697, 0.4613, 0.4553 \end{bmatrix}_{3X3}$$

(e.g. 随机初始化)

Step 4: 定义性能函数

Step 1: 数据准备

Step 2: 网络结构设计

Step 3: 网络参数初始化

Step 4: 定义网络性能函数

Step 5: 定义网络评价指标

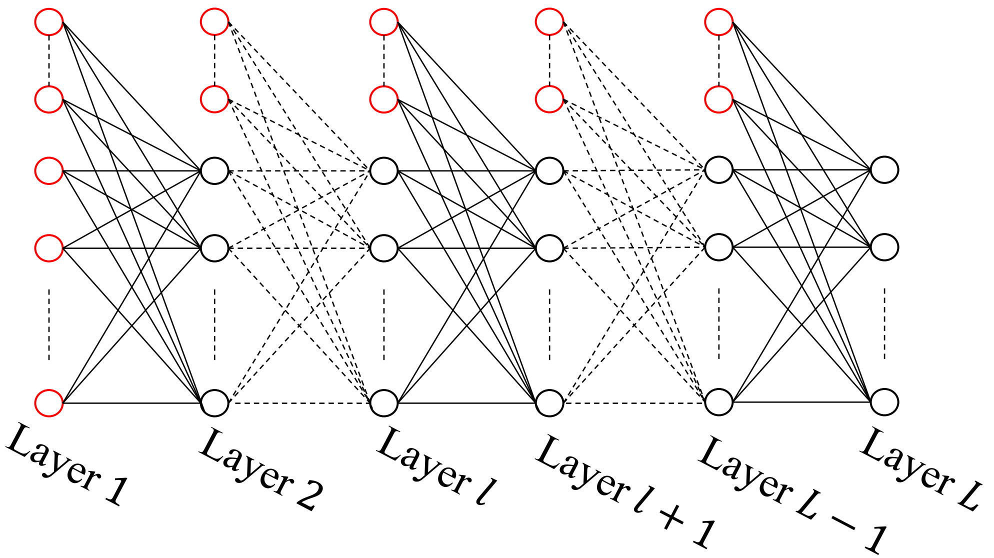
Step 6: 训练网络

Step 7: 测试网络

Step 8: 存储网络

Step 9: 实际应用

Step 4: 定义性能函数

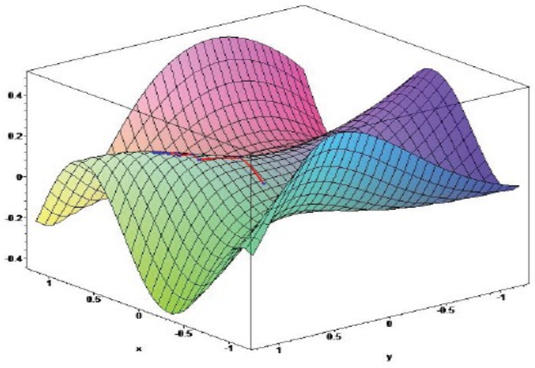


网络预测

$$a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$$

目标输出

$$y^L = \begin{bmatrix} y_1^L \\ \vdots \\ y_{n_L}^L \end{bmatrix}$$



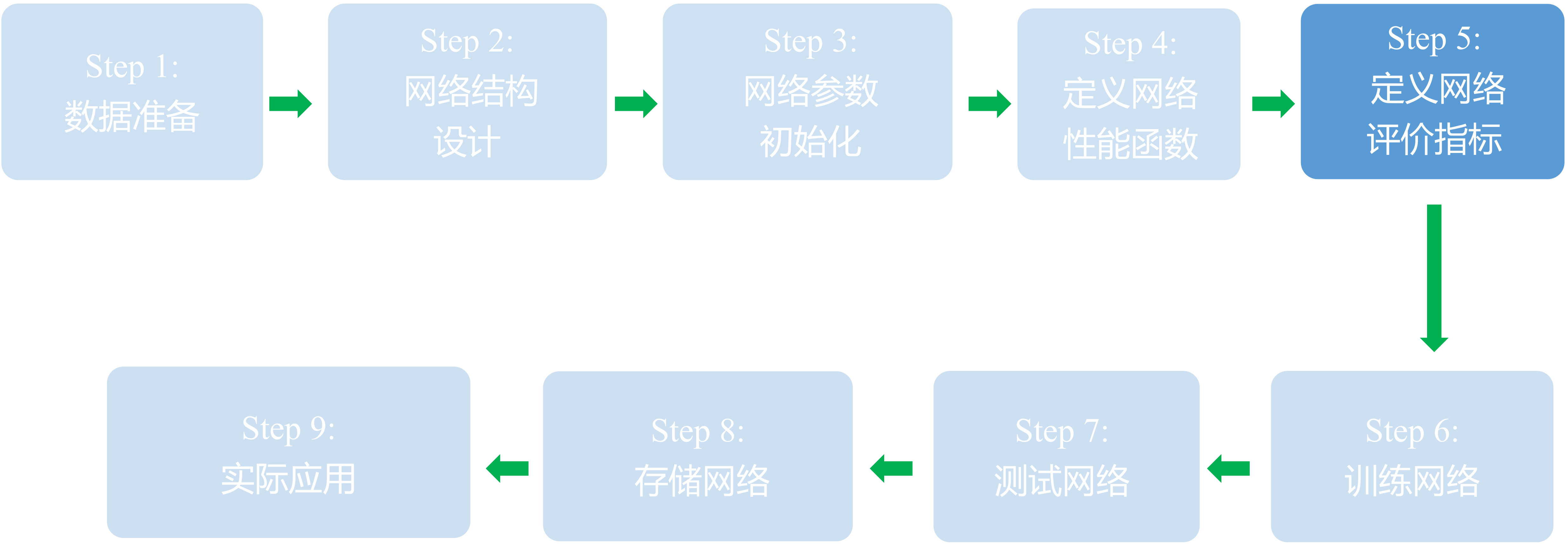
定义性能函数的方式是多种多样的。一种常见的性能函数如下：

$$e_j = a_j^L - y_j^L$$

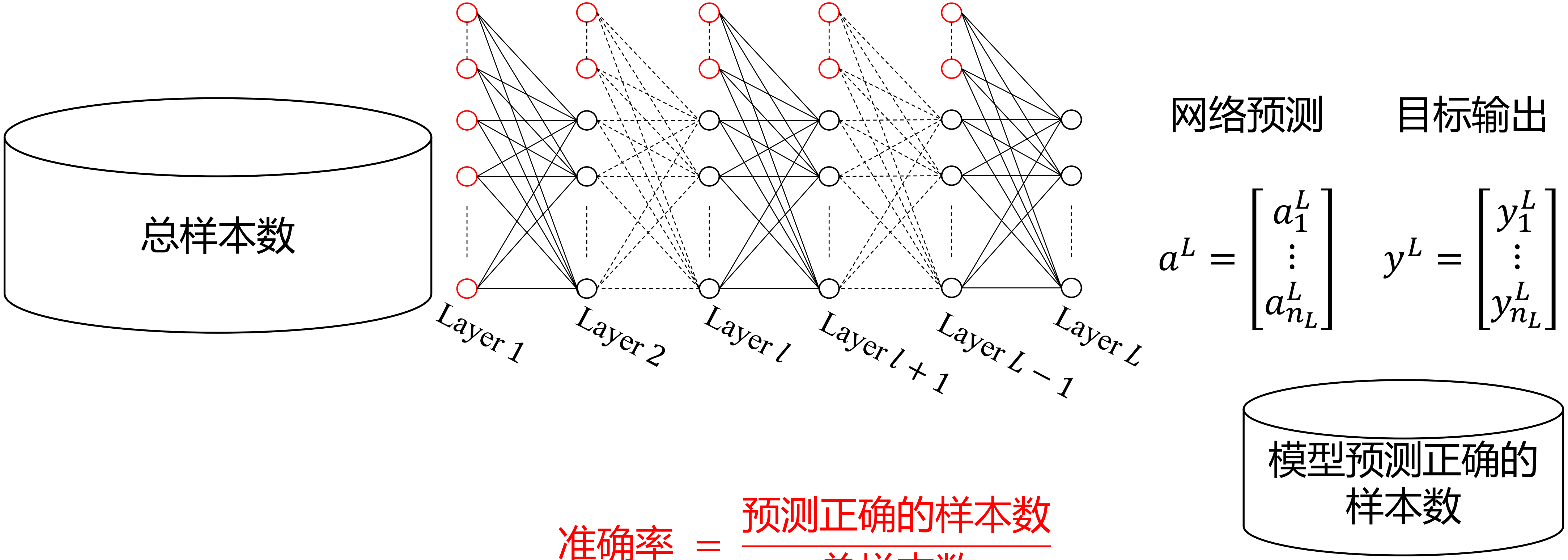
$$J = \frac{1}{2}\sum_{j=1}^{n_L} e_j^2 = J(w^1, \cdots, w^L)$$

显然，J 是 $w^1, \cdots, w^L$ 的函数。

Step 5: 定义网络评价指标



Step 5: 定义网络评价指标



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

$$\text{准确率} = \frac{\text{预测正确的样本数}}{\text{总样本数}}$$

Step 6：训练网络

- Step 1: 数据准备
- Step 2: 网络结构设计
- Step 3: 网络参数初始化
- Step 4: 定义网络性能函数
- Step 5: 定义网络评价指标
- Step 6: 训练网络
- Step 7: 测试网络
- Step 8: 存储网络
- Step 9: 实际应用

BP算法

Step 1. Input the training data set $D = \{(x, y^L)\}$

Step 2. Initialize each w_{ij}^l, and choose a learning rate α

Step 3. for each $D_m \subseteq D$

$\quad\quad \nabla J_{ji} = 0;$

$\quad\quad$ for each $(x, y^L) \in D_m$

$\quad\quad\quad a^1 \leftarrow x \in D_m;$

$\quad\quad\quad$ for $l = 2{:}L$

$\quad\quad\quad\quad a^{l+1} \leftarrow fc(w^l, a^l);$

$\quad\quad\quad$ end

$\quad\quad\quad J(x, y^L);$

$\quad\quad\quad \delta^L = \dfrac{\partial J(x, y^L)}{\partial z^L};$

$\quad\quad\quad$ for $l = L - 1{:}1$

$\quad\quad\quad\quad \delta^l \leftarrow bc(w^l, \delta^{l+1});$

$\quad\quad\quad$ end

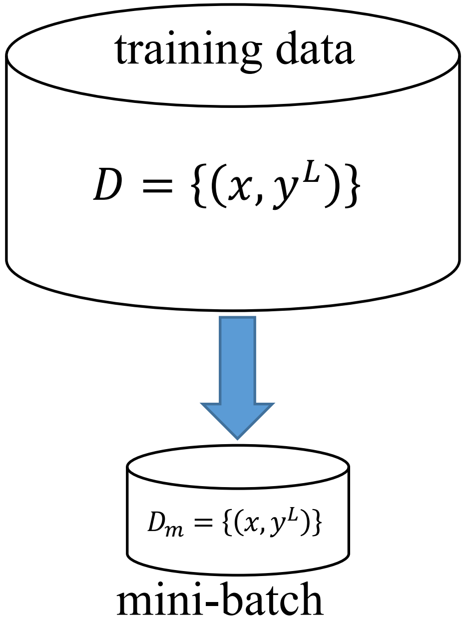
$\quad\quad\quad \nabla J_{ji} \leftarrow \nabla J_{ji} + \delta_j^{l+1} \cdot a_i^l;$

$\quad\quad$ end

$\quad\quad w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \nabla J_{ji};$

$\quad$ end

Step 4. Return to Step 3 until each w_{ji}^l converge



function $fc(w^l, a^l)$

$for\ i = 1{:}n_{l+1}$

$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l$$

$$a_i^{l+1} = f(z_i^{l+1})$$

end

Relationship:

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

function $bc(w^l, \delta^{l+1})$

$for\ i = 1{:}n_l$

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

end

Step 7: 测试网络

Step 1:
数据准备

Step 2:
网络结构设计

Step 3:
网络参数初始化

Step 4:
定义网络性能函数

Step 5:
定义网络评价指标

Step 6:
训练网络

Step 7:
测试网络

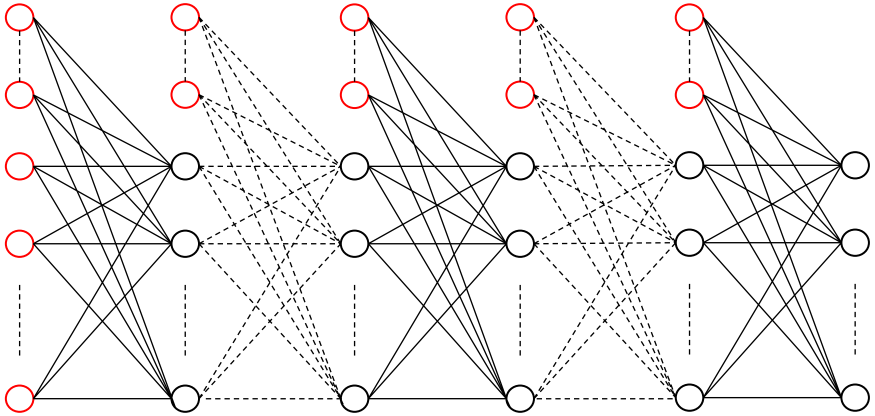
Step 8:
存储网络

Step 9:
实际应用

Step 7: 测试网络

$D_{train} = \{(x, y^L)\}$

训练集



测试神经网络

准确率 $= \dfrac{\text{预测正确的样本数}}{\text{总样本数}}$

$D_{test} = \{(x, y^L)\}$

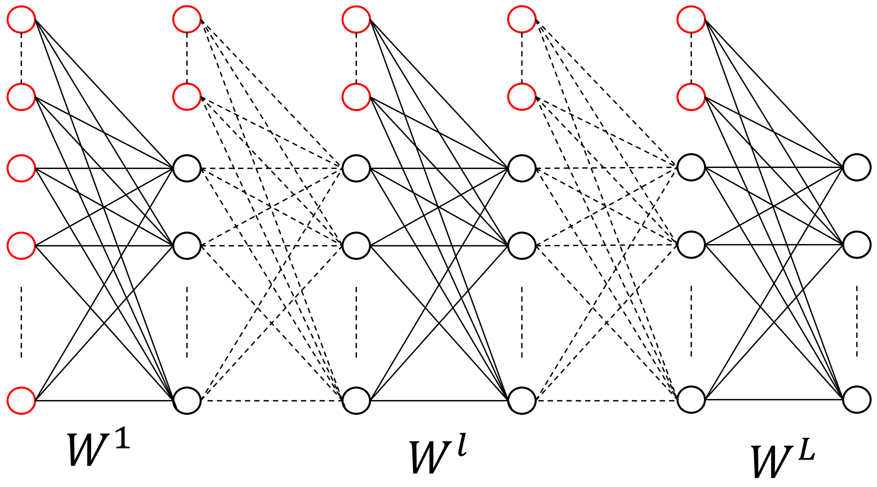
测试集

- 在训练集上测试
- 在测试集上测试

Step 8: 存储网络

Step 1: 数据准备 → Step 2: 网络结构设计 → Step 3: 网络参数初始化 → Step 4: 定义网络性能函数 → Step 5: 定义网络评价指标 → Step 6: 训练网络 → Step 7: 测试网络 → **Step 8: 存储网络** → Step 9: 实际应用

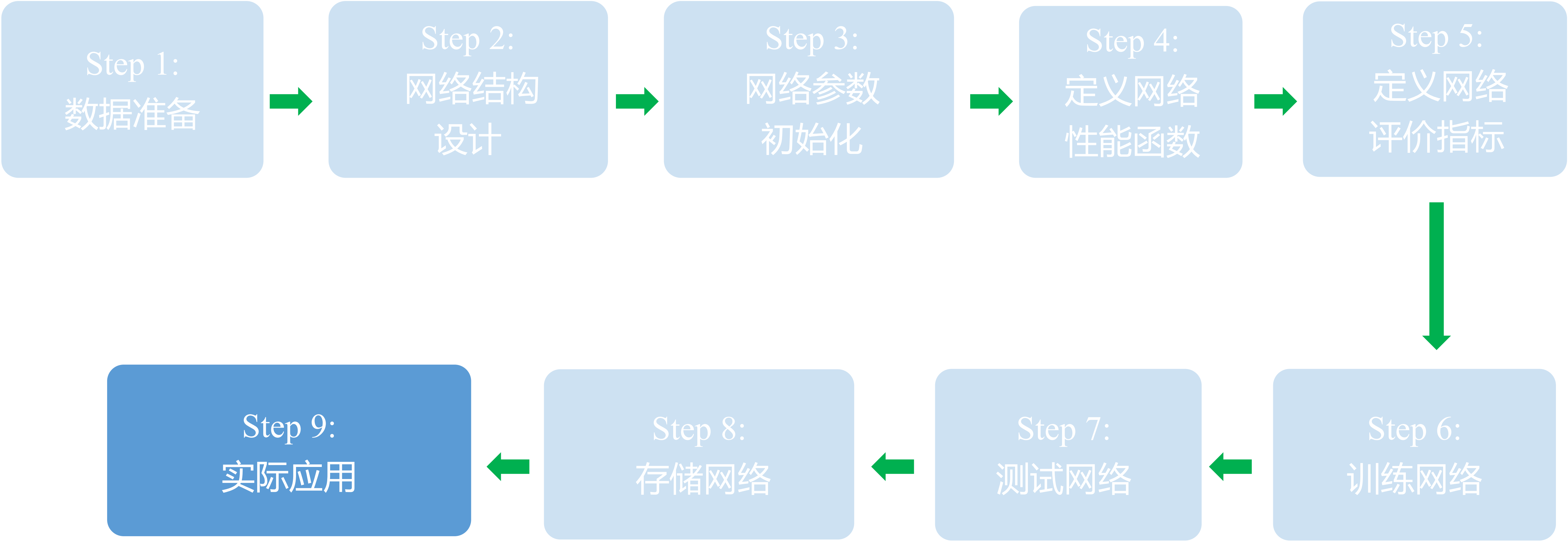
Step 8: 存储网络



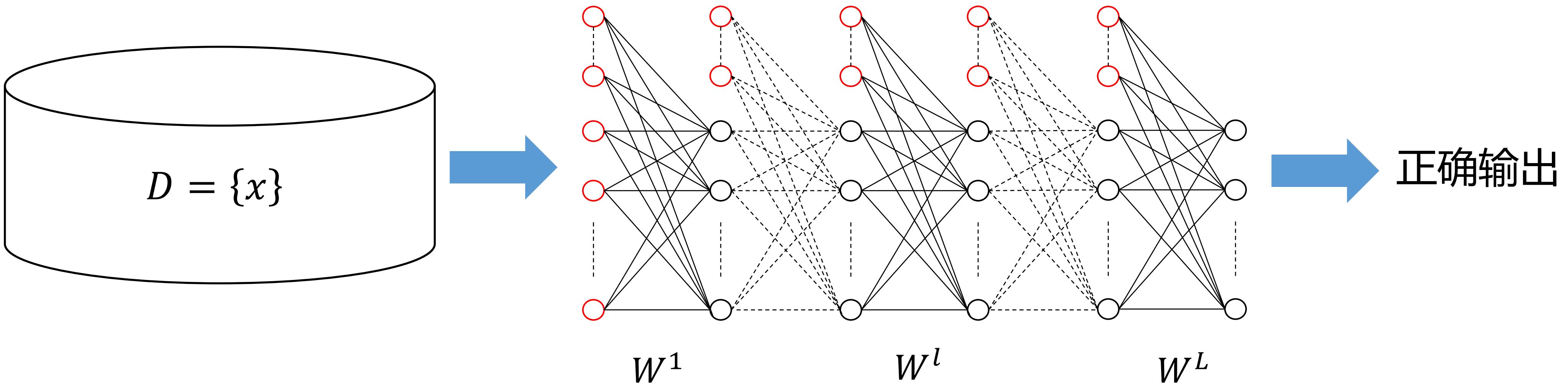
$$W^l = \begin{bmatrix} w_{11}^l & \cdots & w_{1n_{l+1}}^l \\ \cdots & w_{ij}^l & \cdots \\ w_{n_l1}^l & \cdots & w_{n_l \times n_{l+1}}^l \end{bmatrix}_{n_l \times n_{l+1}}$$

- 保存网络的所有参数
  - 每一个连接权矩阵
  - 模型层数
  - 每层的神经元数目
  - 学习率
  - ……

Step 9: 实际应用


Step 1:
数据准备
Step 2:
网络结构设计
Step 3:
网络参数初始化
Step 4:
定义网络性能函数
Step 5:
定义网络评价指标
Step 6:
训练网络
Step 7:
测试网络
Step 8:
存储网络
Step 9:
实际应用

Step 9: 实际应用



作业：编程实现反向传播算法

- 提供MATLAB模版
- 可以使用MATLAB或Python
- 下周上课前交（9/30）

Step 1. Input the training data set $D=\{(x,y^L)\}$

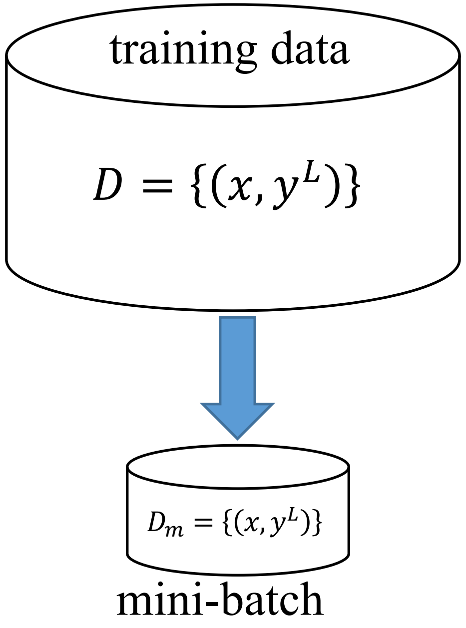
Step 2. Initialize each w_{ij}^l, and choose a learning rate α

Step 3. for each mini-batch sample $D_m \subseteq D$

$$
\begin{aligned}
&\nabla J_{ji}=0;\\
&\text{for each } (x,y^L)\in D_m\\
&\quad a^1 \leftarrow x\in D_m;\\
&\quad \text{for } l=2:L\\
&\quad\quad a^{l+1}\leftarrow fc(w^l,a^l);\\
&\quad \text{end}\\
&\quad J(x,y^L);\\
&\quad \delta^L=\frac{\partial J(x,y^L)}{\partial z^L};\\
&\quad \text{for } l=L-1:2\\
&\quad\quad \delta^l\leftarrow bc(w^l,\delta^{l+1});\\
&\quad \text{end}\\
&\quad \nabla J_{ji}\leftarrow \nabla J_{ji}+\delta_j^{l+1}\cdot a_i^l;\\
&\text{end}\\
&w_{ji}^l\leftarrow w_{ji}^l-\alpha\cdot\nabla J_{ji};\\
\end{aligned}
$$

end

Step 4. Return to Step 3 until each w^lconverge



function $fc(w^l,a^l)$

$for\ i=1:n_{l+1}$

$$z_i^{l+1}=\sum_{j=1}^{n_l} w_{ij}^l a_j^l$$

$$a_i^{l+1}=f(z_i^{l+1})$$

end

Relationship:

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

function $bc(w^l,\delta^{l+1})$

$for\ i=1:n_l$

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

end

课程信息

时间：2022年秋季学期 1-8周 周五 3-4节

线下：江安文科楼三区203

线上：

http://www.machineilab.org/

http://guoquan.net/

Thanks