北京大學

人工神经网络：概述

·AIPKU·

人工智能引论

主讲人：刘家瑛

Research Group

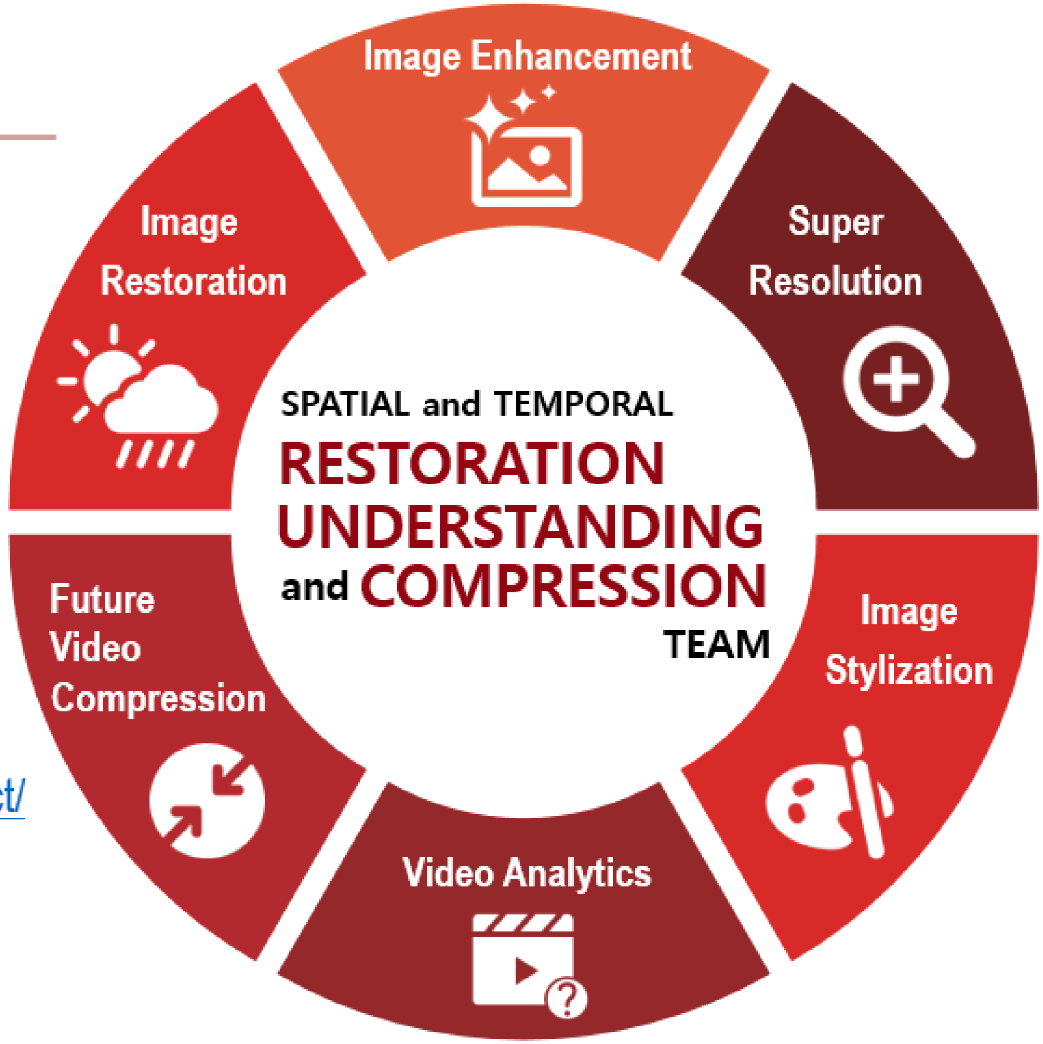
STRUCT Group

智能影像计算

北京大学 王选计算机研究所

Spatial and Temporal **Restoration**, **Understanding** and **Compression** Team

- PI: 刘家瑛
- 邮箱: liujiaying@pku.edu.cn
- 网页: http://www.wict.pku.edu.cn/struct/



主要内容

- 什么是人工神经网络

  Artificial Neural Network, ANN

- 卷积神经网络

  Convolution Neural Network, CNN

- 循环神经网络

  Recurrent Neural Network, RNN

- 前沿: 超越CNN和RNN

主要内容

- **什么是人工神经网络** (ANN)
- 卷积神经网络 (CNN)
- 循环神经网络 (RNN)
- 前沿: 超越CNN和RNN

2018年图灵奖获得者

‘Godfathers of AI’

- Yann LeCun
- Geoffrey Hinton
- Yoshua Bengio

The New York Times

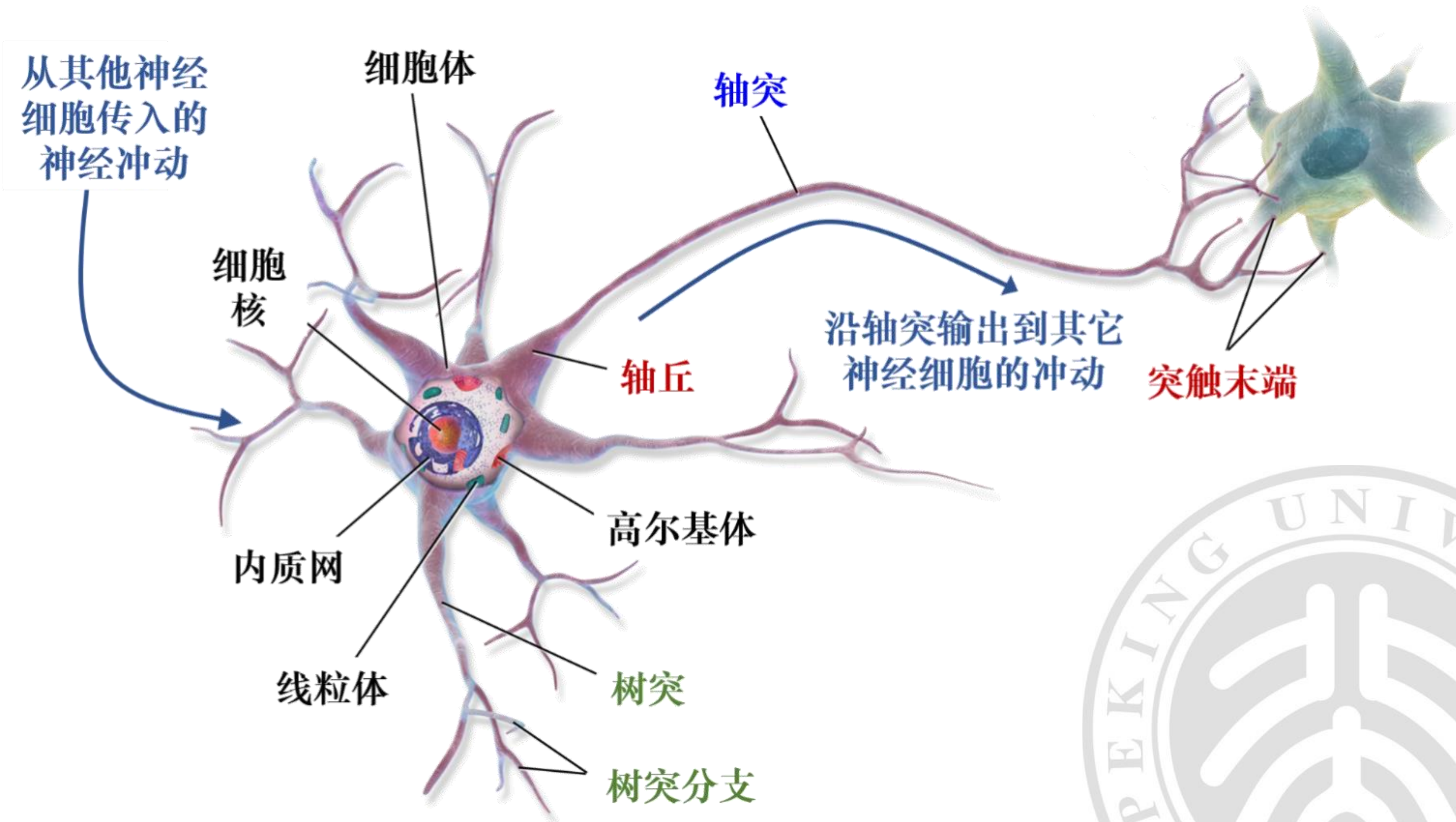
Three Pioneers in Artificial Intelligence Win Turing Award

Source: https://www.nytimes.com/2019/03/27/technology/turing-award-hinton-lecun-bengio.html

什么是人工神经网络

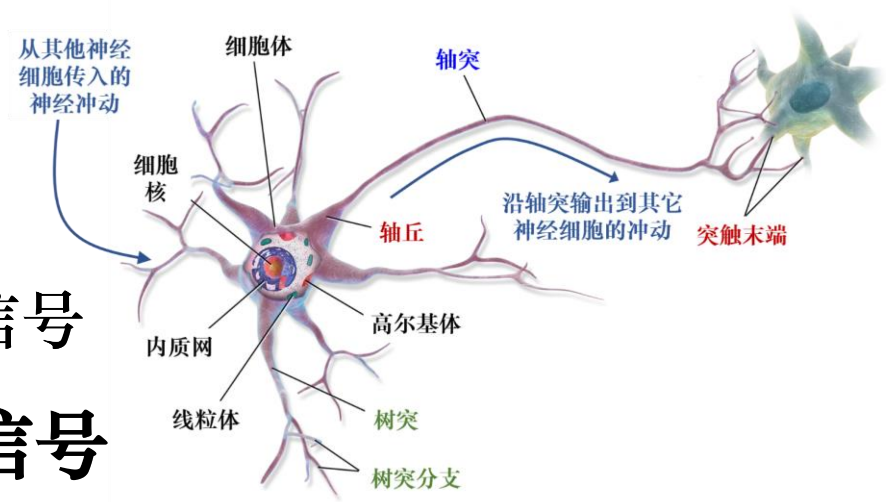
- 生物学基础
- 感知机模型
- 激活函数
- 多层感知机模型
- 反向传播算法
- 损失函数

典型大脑皮层神经元结构



典型大脑皮层神经元结构

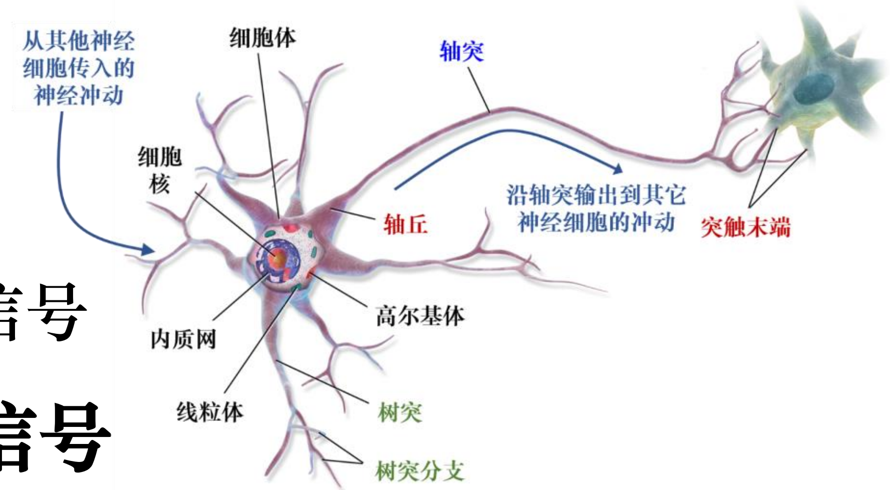
- **物理结构**
  - **轴突**: 分支结构，向外传输信号
  - **树突**: 从多个其它神经元接收信号



- **轴突和树突之间由突触传递信号**
  - 突触的神经递质释放引起下一个神经元树突的电位变化

典型大脑皮层神经元结构

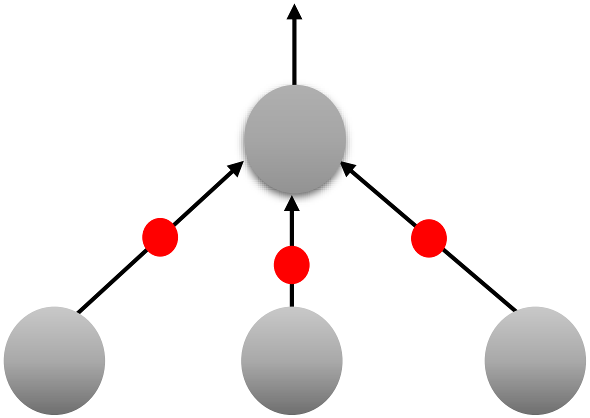
- **物理结构**
  - **轴突**: 分支结构，向外传输信号
  - **树突**: 从多个其它神经元接收信号



- **轴突和树突之间由突触传递信号**
  - 突触的神经递质释放引起下一个神经元树突的电位变化
- **神经冲动的产生**
  - 轴突和细胞体的轴丘根据细胞接收和整合的电位情况决定 → 是否产生神经冲动
  - 神经冲动由轴突传递到其它神经元

抽象神经元模型: 大脑如何工作

- 每个神经元接收来自其它神经元的信号
  - 少数神经元的冲动信号来自于感受器
  - 大脑皮层神经元之间用神经冲动进行交流
- 传入信号对神经元电位的影响受到突触上“权值”的控制
  - 根据递质受体的不同，可能有 激活 或者 抑制



抽象神经元模型: 大脑如何工作

- 受体的 “权值” 会自动适应调整
- 神经元连接起来 → 实现有 “计算任务” 的神经元网络
  - 从认识物体, 到学习语言, 制定计划, 控制身体 …
- 一个人类大脑大约包含 **10^{11} 数量级**的神经元和 10^{15} 数量级的突触连接 (权值)
  - 由于神经元数量十分庞大, 连接非常复杂, 大脑却可以在很短的时间响应一个任务?

大脑的模块化和分区结构

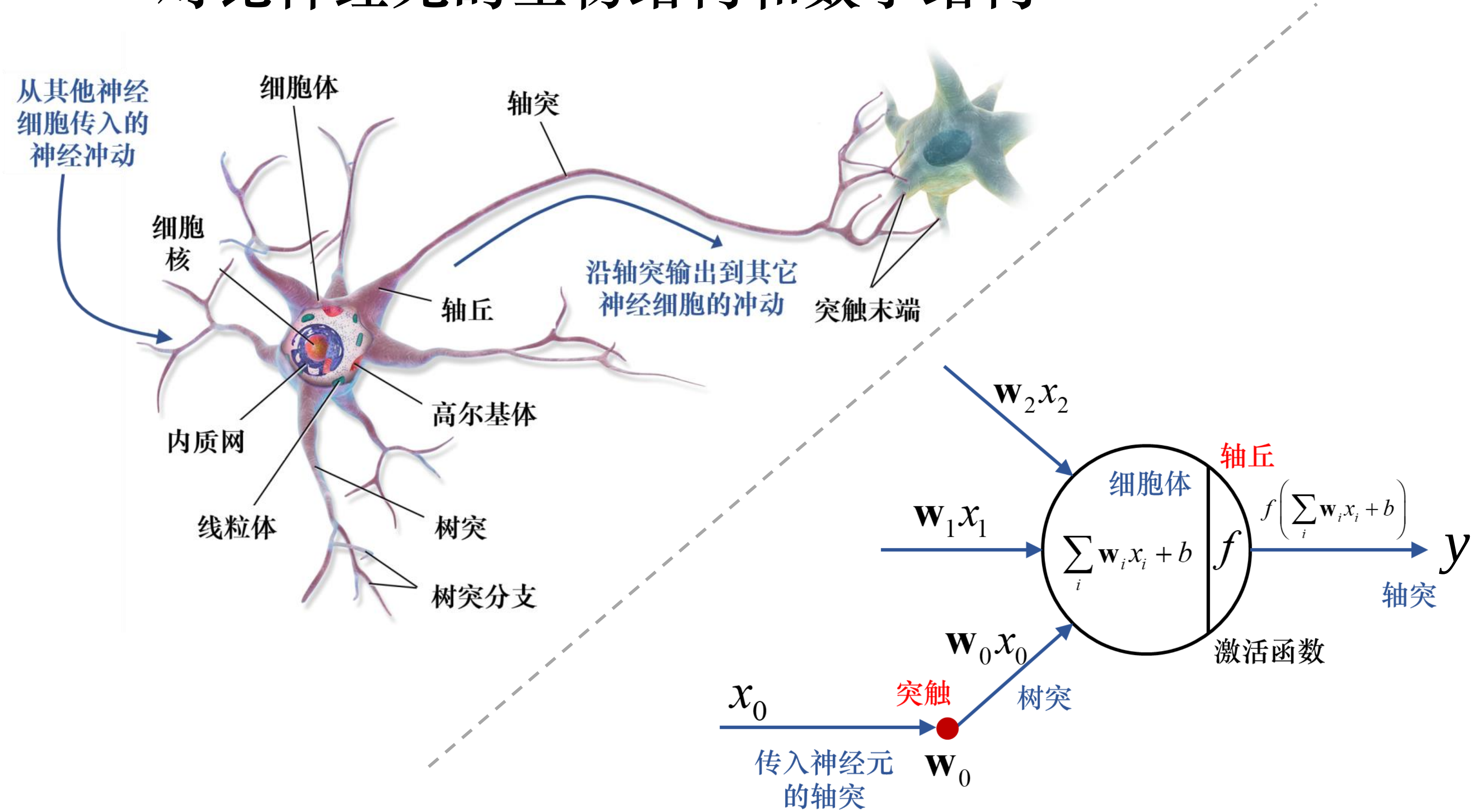
- 皮层的不同区域处理不同的任务
  - 现象: 脑部不同区域的损伤会导致不同机能丧失
  - 现象: 执行特定任务时观察到局部脑区血流量增加
- 大脑的不同区域在形态上是相似的
  - 现象: 幼年的脑部损伤时, 对应功能区会迁移到其它脑区

大脑的模块化和分区结构

- **皮质是由结构一致的神经细胞构成**
  - 在脑部发育和学习/习得的过程中，不同的脑区会被“训练”成具有不同功能的“专用硬件”
  - 因此大脑能够同时实现 **并行性** 和 **可塑性**
  - 传统计算机可以通过程序设计，实现并行性和可塑性
  - 近年，有了更强大的中央处理器和更高并行的图形处理器(GPU) → 计算机能够对大脑的神经元网络进行抽象的模拟

从生物结构到数学结构

- 对比神经元的生物结构和数学结构



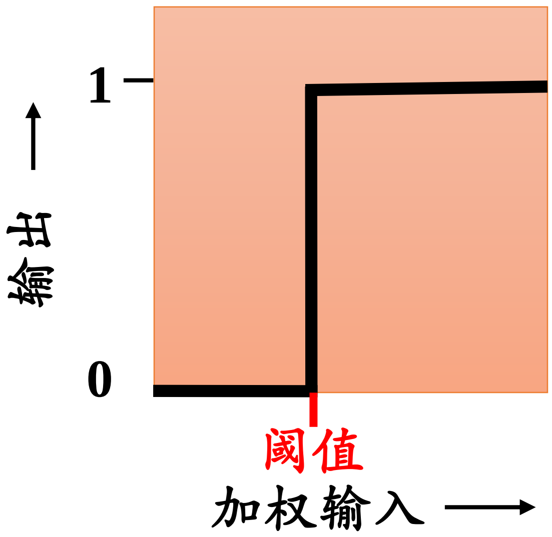
https://en.wikipedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png

二值神经元模型 McCulloch-Pitts [1943]

- 计算输入的加权和
- 如果加权和大于阈值, 则输出一个固定大小的神经冲动
- McCulloch 和 Pitts 认为,
  - 神经元 可以对一个命题进行真假判断
  - 多个神经元 整合各个命题真值 形成逻辑
- 该模型是实际情况的一种简化

  例: 实际中不同的神经元的激活强度可能

  是不一样的, 在这里用 **0-1** 近似了



神经元模型数学表达

- 二值神经元模型 可以用 线性方式表达
  - 不同的神经元的权值和阈值都可以不同

$$z = \sum_i x_i \mathbf{w}_i \qquad \xrightarrow{\theta = -b} \qquad z = b + \sum_i x_i \mathbf{w}_i$$

$$y = \begin{cases} 1, \text{ if } z \geq \theta \\ 0, \text{ otherwise} \end{cases} \qquad\qquad y = \begin{cases} 1, \text{ if } z \geq 0 \\ 0, \text{ otherwise} \end{cases}$$

- 实质为 线性函数 结果的符号判断

阈值

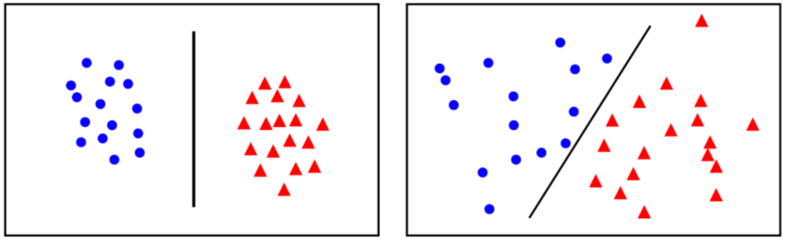
$$z = b + \sum_i x_i \mathbf{w}_i$$

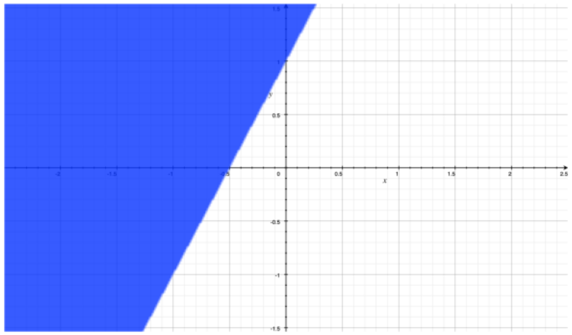
输出

对每个神经输入
进行加权求和

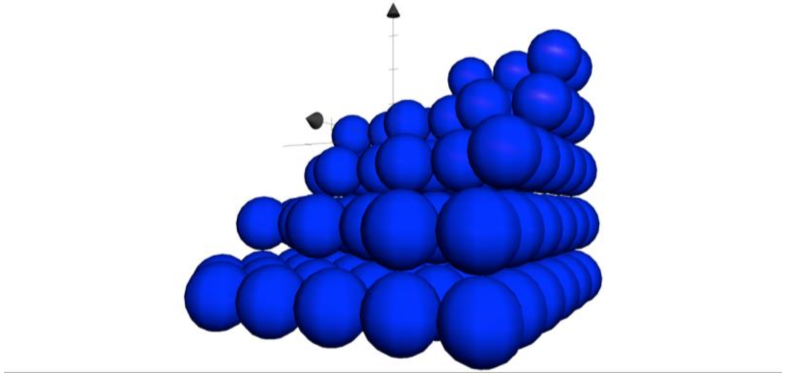
线性神经元 → 分类超平面

- 神经元 → 线性函数结果的正负号
  - 形式: $f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathbf{b})$
- 线性函数 → 分类超平面
  - 可分类**线性可分的**数据





$$2x - y + 1 < 0$$



$$2x - y + 3z + 1 < 0$$

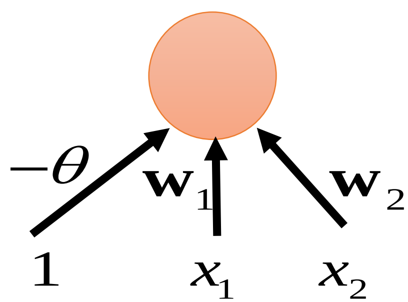
线性阈值神经元的局限

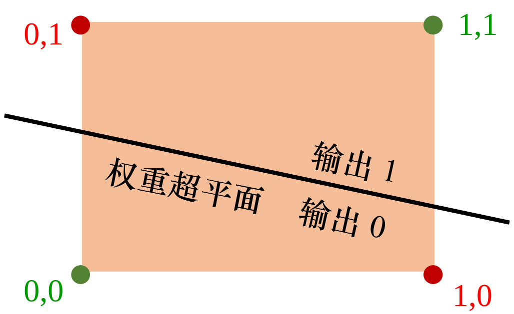
- 线性阈值神经元无法实现**异或**运算
- 异或运算真值表: (1,1) → 1; (0,0) → 1

  (1,0) → 0; (0,1) → 0
- 将真值表代入神经元表达式, 得到的方程无解

$$\mathbf{w}_1 + \mathbf{w}_2 \geq \theta, \quad 0 \geq \theta$$

$$\mathbf{w}_1 < \theta, \quad \mathbf{w}_2 < \theta$$





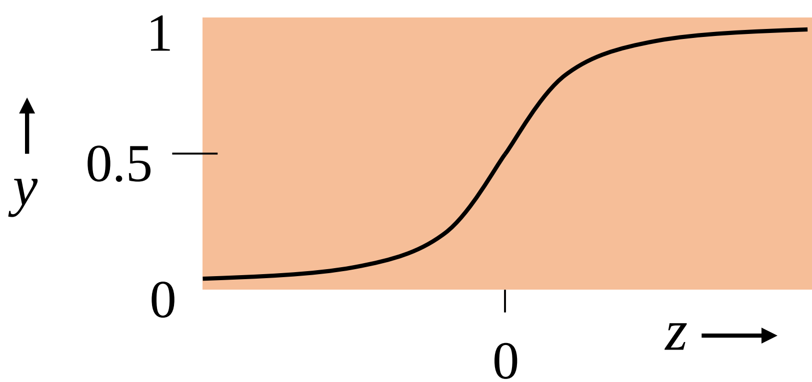
- 由于多个线性矩阵运算总是可以合并为一个矩阵运算, 多个神经元的组合无法得到高于单个神经元的表达能力
- **解决方案: 非线性激活函数和隐藏层**

非线性激活函数: Sigmoid

- 单调函数: 不同的神经元可以有不同的响应强度
- 可导: 可以使用梯度方法进行训练
- 非线性: 为模型提供非线性建模能力
- 响应曲线与二值模型和生物模型相似

$$z = b + \sum_i x_i \mathbf{w}_i$$

$$y = \frac{1}{1 + e^{-z}}$$

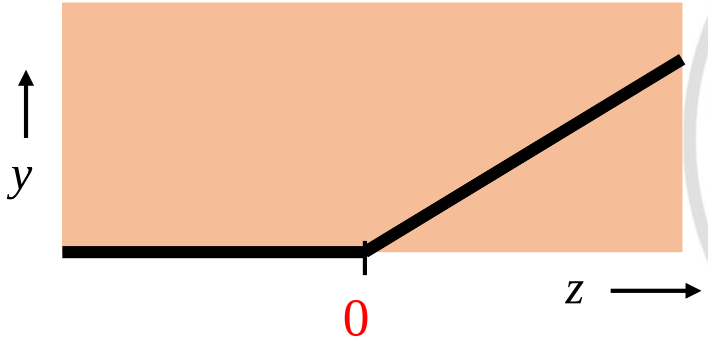


非线性激活函数：整流线性模型 ReLU

- 更简单的非线性神经元设计
  - 生物神经元存在“激活”和“抑制”
  - 数学模型中，负值与正值对称，无法表达“抑制”
  - 在线性神经元中引入整流机制，使得负响应被“抑制”
  - 满足激活函数的要求，并且计算更加简单

$$z = b + \sum_i x_i \mathbf{w}_i$$

$$y = \begin{cases} z, \text{if } z > 0 \\ 0, \textit{otherwise} \end{cases}$$

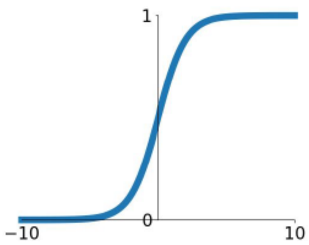


非线性激活函数

ReLU在大多数问题中
都是较好的默认选择

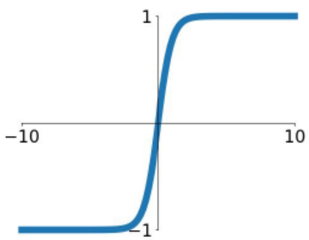
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



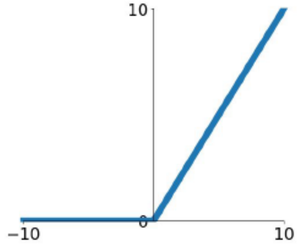
tanh

$\tanh(x)$



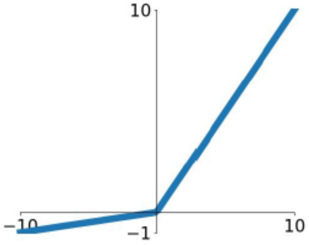
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

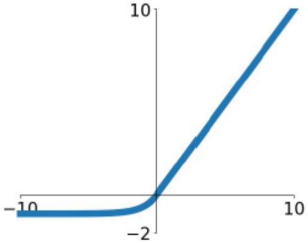


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

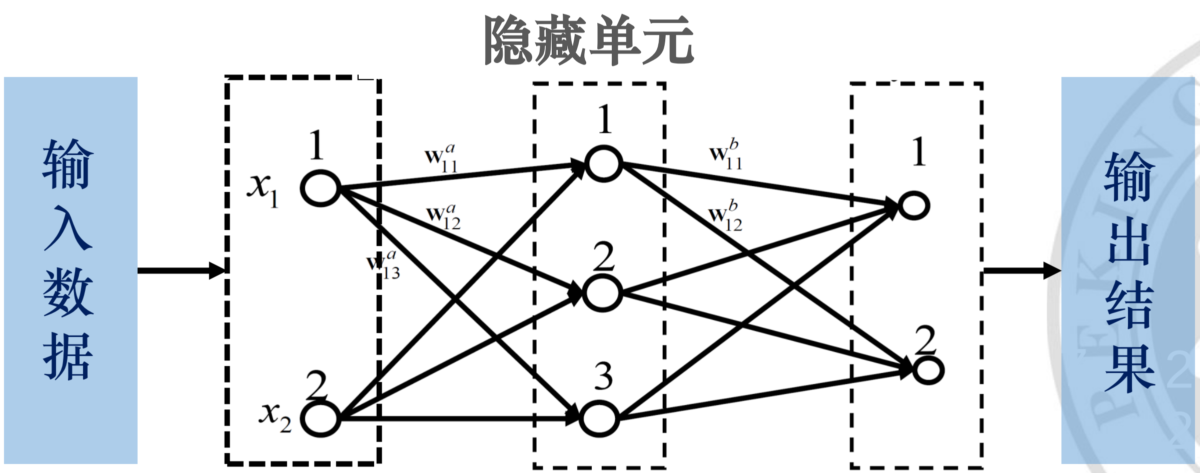
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

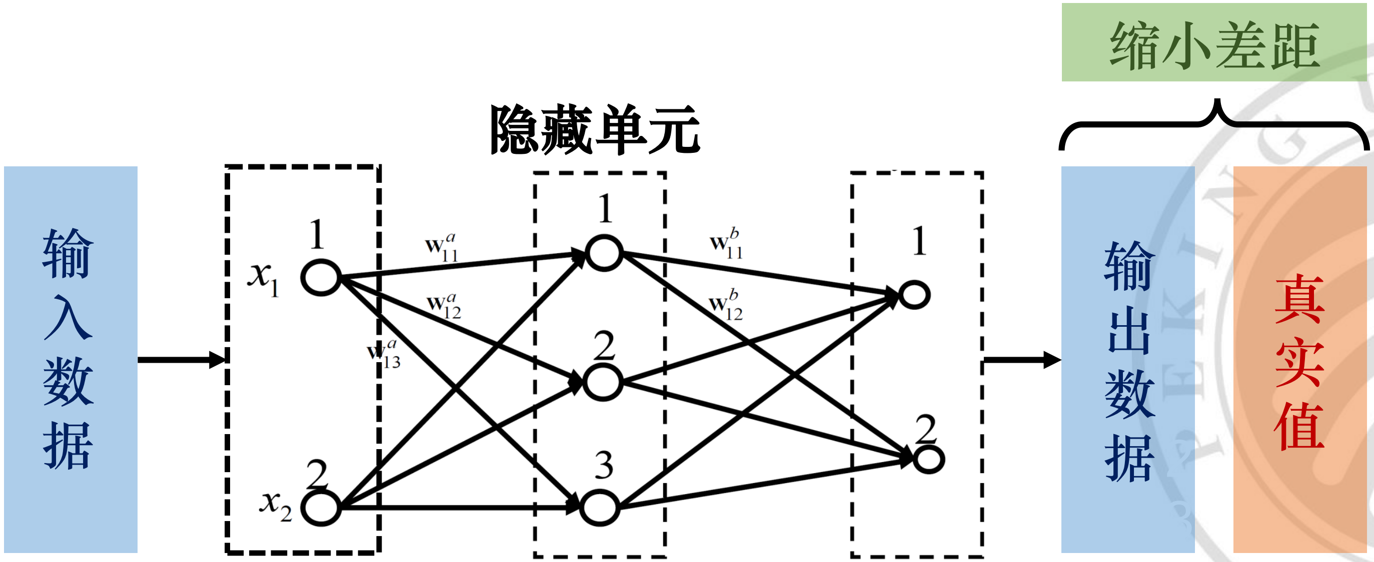


多层结构与隐藏单元

- 包含**隐藏单元**的**多组神经元**组成 **神经元网络**
- 输入与输出神经元之外的神经元被称为**隐藏单元**
- 模拟大脑的多层神经元结构能够带来增强的建模能力
- 线性神经元计算结果经过**激活函数**产生输出



- 但是需要解决以下问题：
  - 单个神经元可以使用梯度优化的算法进行训练
  - 隐藏神经元没有明确的输出目标，无法直接训练
  - 多层神经元需要有效的训练算法



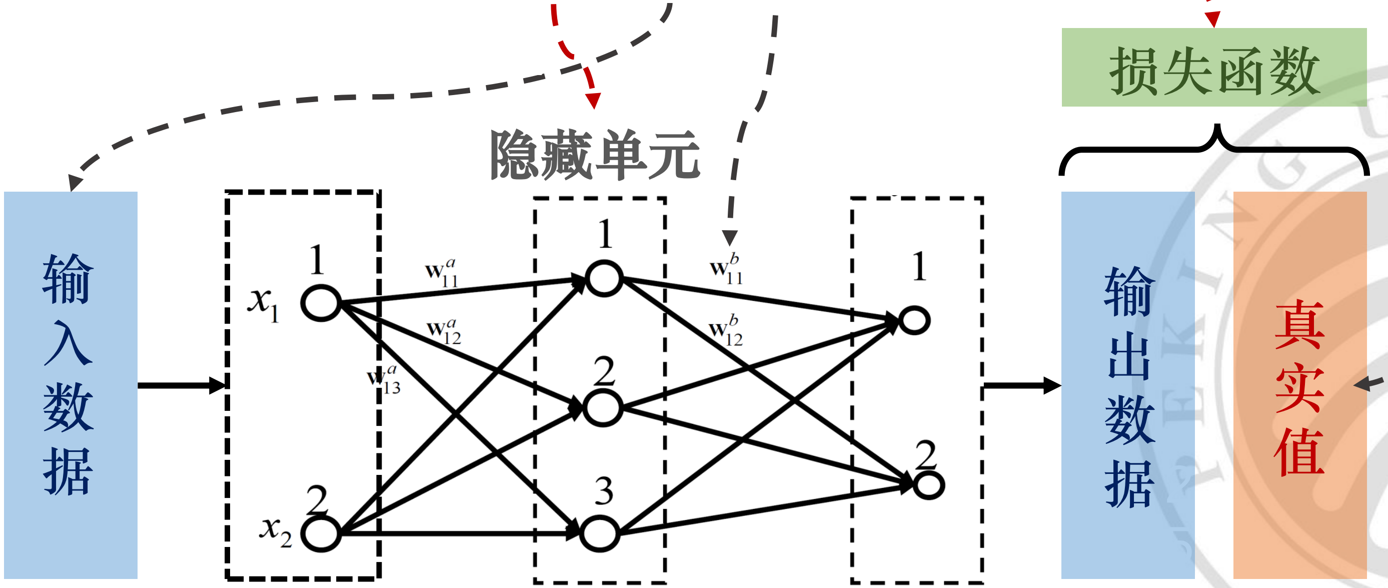
训练与最优化

- 将问题表达为最优化形式
  - 给定一组数据作为训练数据
  - 神经元网络中神经元的权值和阈值为待确定的参数
- 最小化神经元网络预测的“误差”
  - 将“误差”表达为损失函数
  - 损失函数取值越小，网络的预测越准确

多层结构与隐藏单元

- 将**损失函数**表达成关于数据和参数的**复合函数**
- 调整参数 $\boldsymbol{w}$, 使损失函数值最小化, 即**最小化误差**

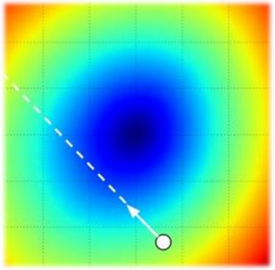
$$\mathcal{L} = f(g(x, w), y)$$



最优化: 梯度方法

- 梯度
  - 目标函数关于各参数偏导数构成的向量
  - 负梯度方向是函数值在该点下降最 "陡" 的方向

$$\boxed{\frac{df(x)}{dx} = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h}}$$



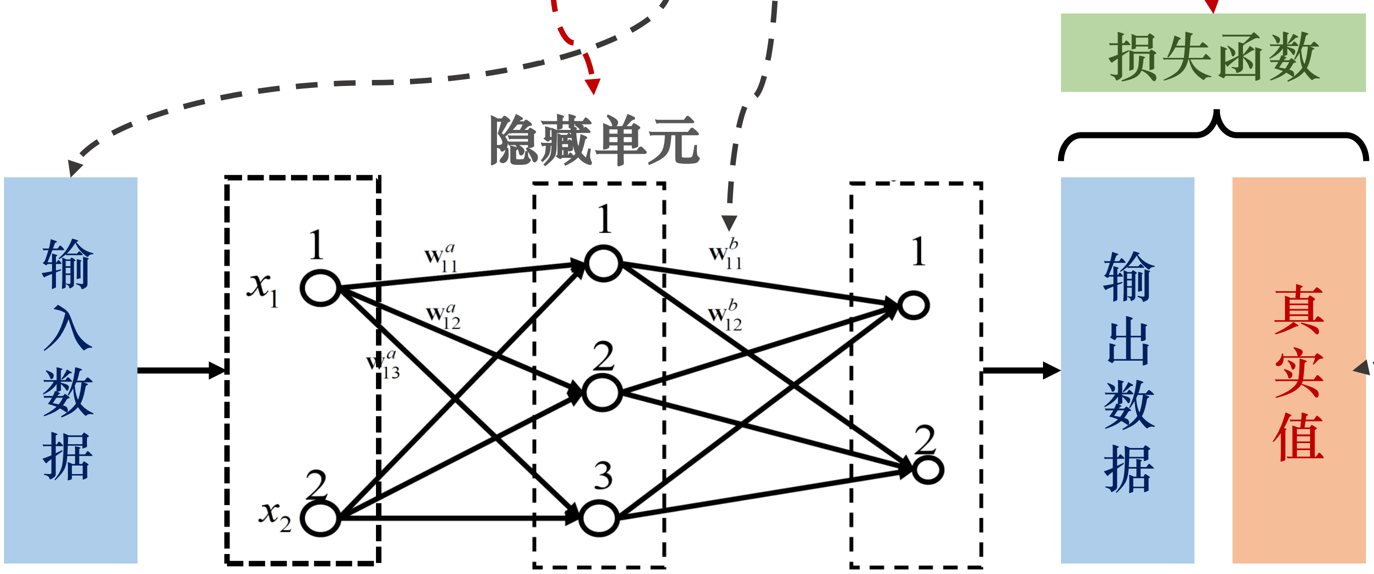
- 梯度下降
  - 参数量大, 无法得出闭式解, 因此用数值迭代方法
  - 沿着负梯度方向多次调整参数, 降低目标函数值
- 随机梯度下降
  - 求解时从训练数据中随机选取一定的样本进行迭代
  - 节约计算资源, 同时减少陷入局部最优的风险

多层结构与隐藏单元

- 计算损失函数在输入为某一 x, y, w 取值时关于 w 的梯度

$$\left.\left(\frac{\partial \mathcal{L}}{\partial w}\right)\right|_{x=x_0, w=w_0, y=y_0}$$

$$\mathcal{L} = f(g(x, w), y)$$



复合函数的求导

- 假设一个复合函数

$$f(x, y, w) = (x + y)w$$

- 现在为了训练其中的参数，需要求函数值关于其中各个参数的梯度

$$q = x + y \rightarrow \frac{\partial q}{\partial x} = 1, \ \frac{\partial q}{\partial y} = 1 \qquad f = qw \rightarrow \frac{\partial f}{\partial q} = w, \ \frac{\partial f}{\partial w} = q$$

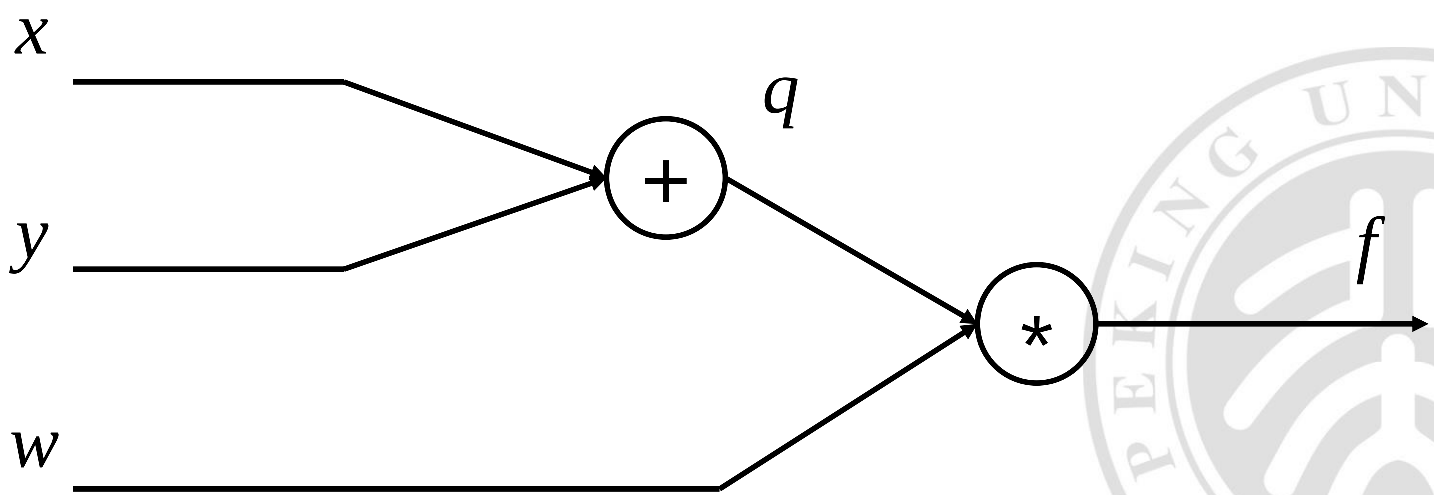
- 应用 **链式法则** 可得所需梯度

复合函数的求导

- 复合函数可以表示为一个有向无环图

$$f(x,y,w)=(x+y)w \qquad q=x+y \rightarrow \frac{\partial q}{\partial x}=1,\ \frac{\partial q}{\partial y}=1$$

$$x=-2,\ y=5,\ w=-4 \qquad f=qw \rightarrow \frac{\partial f}{\partial q}=w,\ \frac{\partial f}{\partial w}=q$$

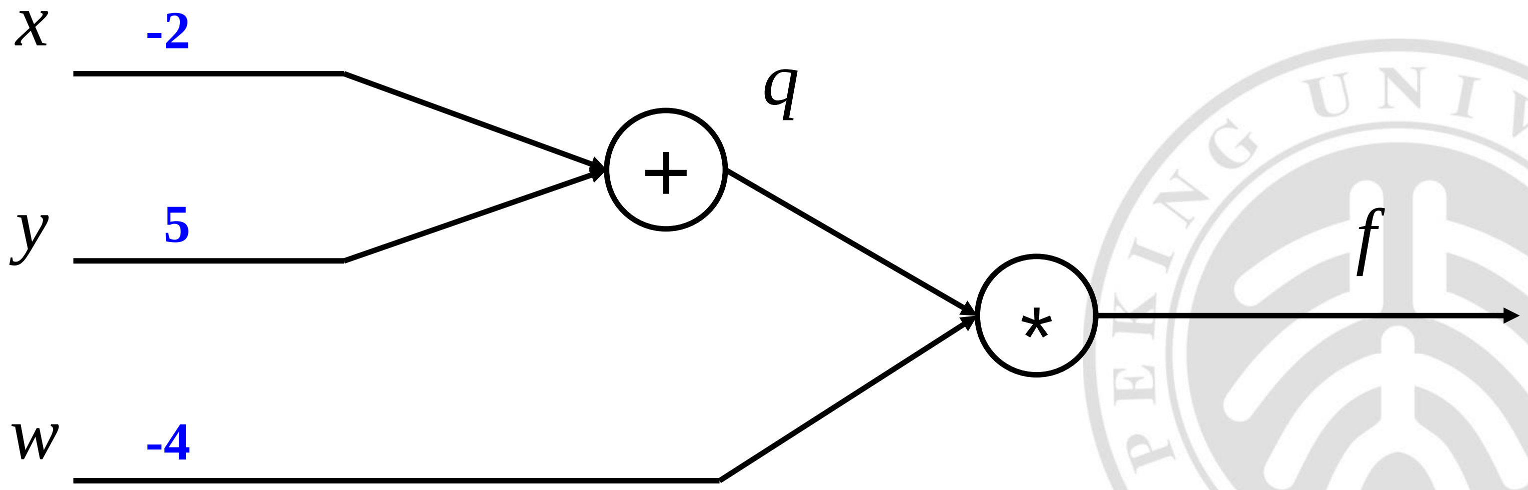


复合函数的求导

- 复合函数可以表示为一个有向无环图

$$f(x, y, w) = (x + y)w$$

$$q = x + y \rightarrow \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1 \qquad f = qw \rightarrow \frac{\partial f}{\partial q} = w, \frac{\partial f}{\partial w} = q$$

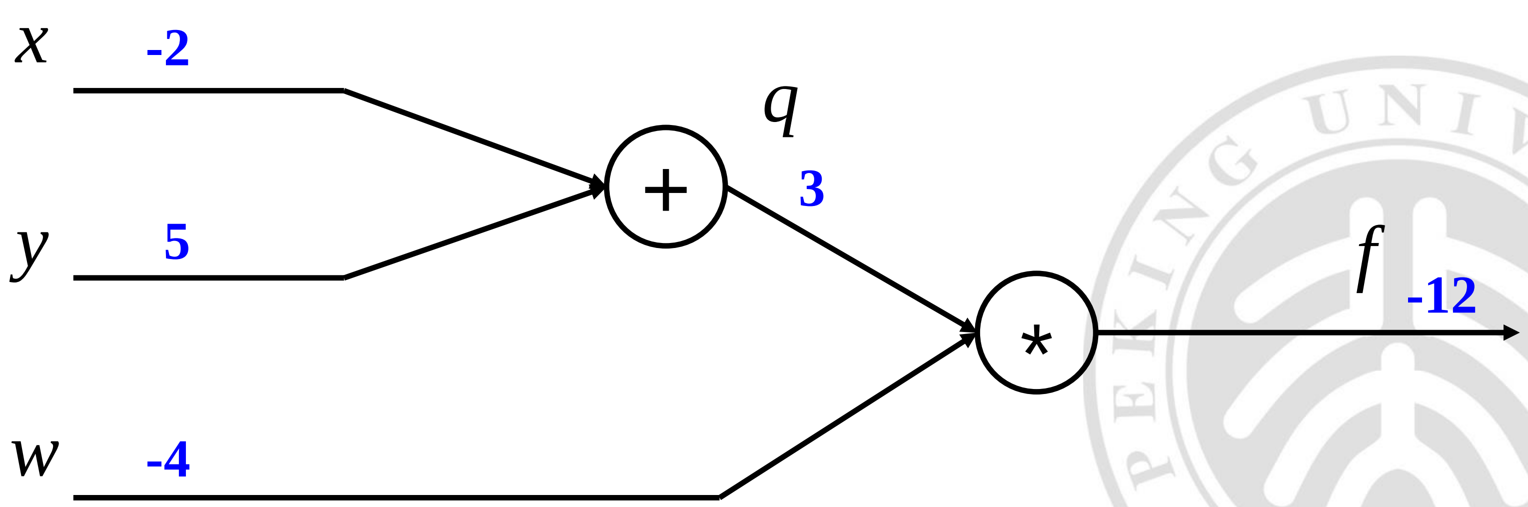


复合函数的求导

- 复合函数可以表示为一个有向无环图

$$f(x,y,w)=(x+y)w$$

$$q=x+y \rightarrow \frac{\partial q}{\partial x}=1,\ \frac{\partial q}{\partial y}=1 \qquad f=qw \rightarrow \frac{\partial f}{\partial q}=w,\ \frac{\partial f}{\partial w}=q$$

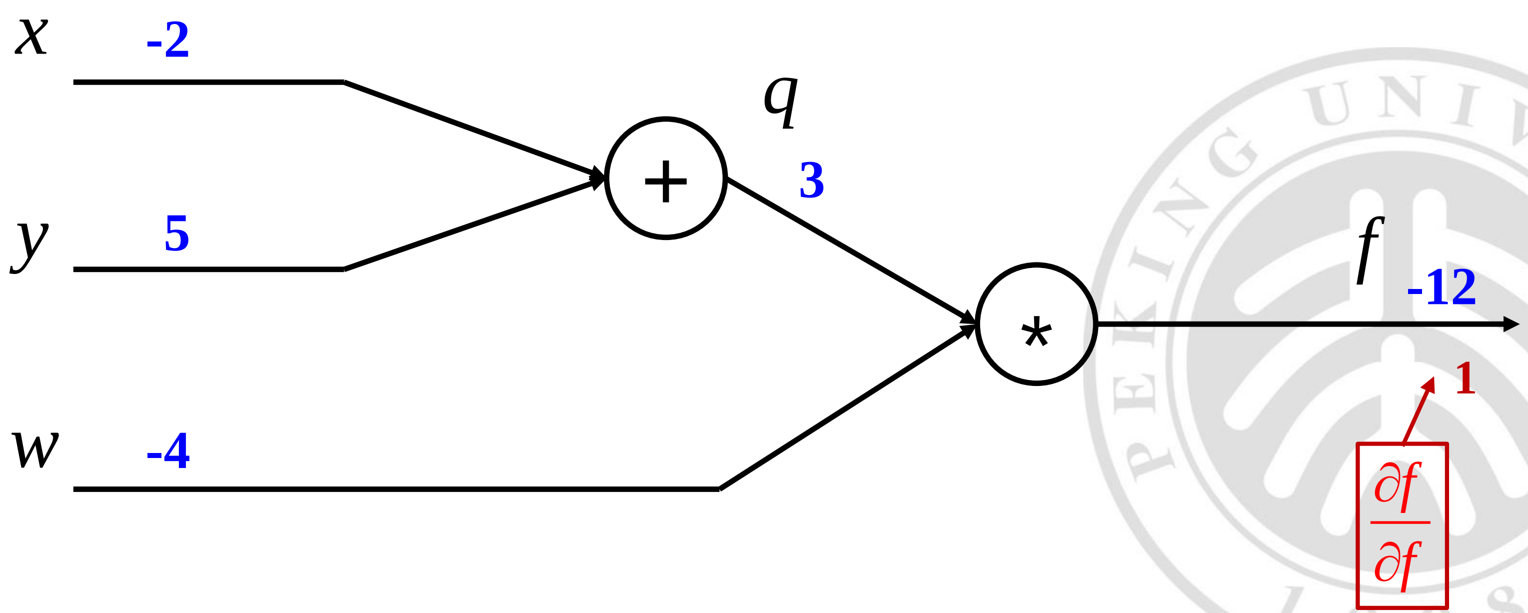


复合函数的求导

- 复合函数可以表示为一个有向无环图

$$f(x,y,w)=(x+y)w$$

$$q=x+y \rightarrow \frac{\partial q}{\partial x}=1, \frac{\partial q}{\partial y}=1 \qquad f=qw \rightarrow \frac{\partial f}{\partial q}=w, \frac{\partial f}{\partial w}=q$$

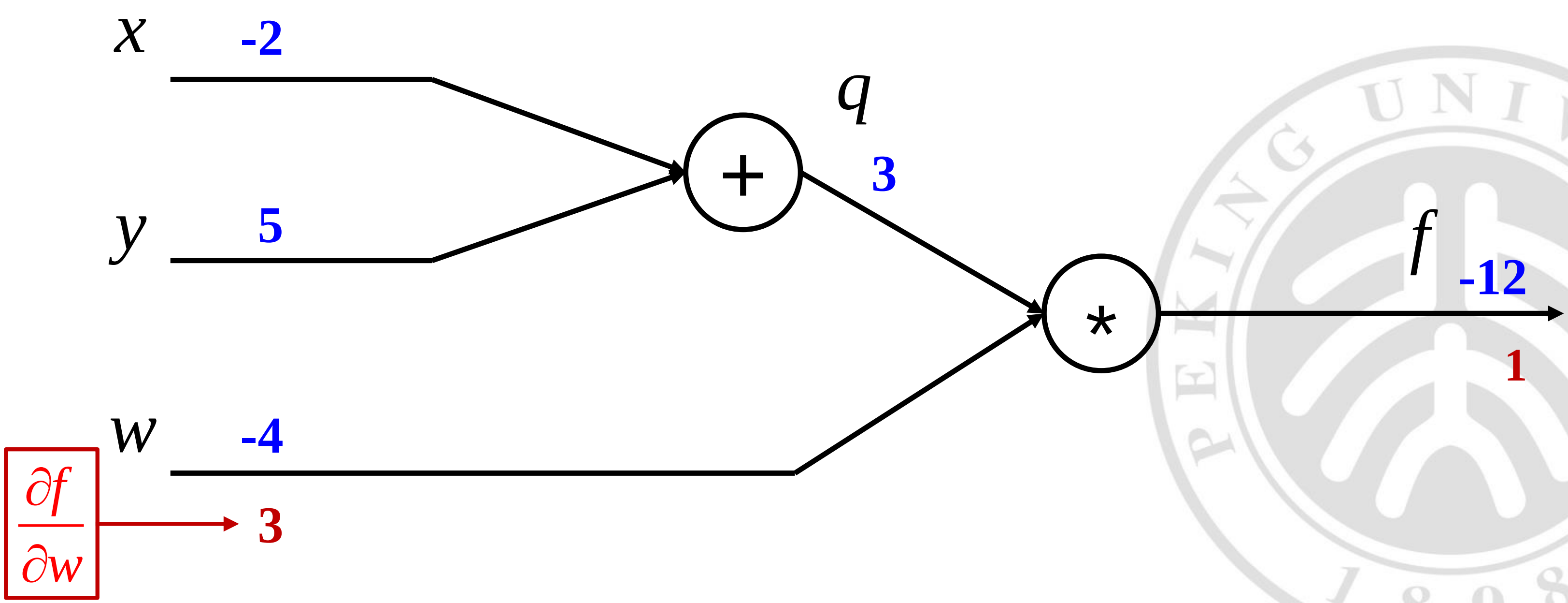


复合函数的求导

- 复合函数可以表示为一个有向无环图

$$f(x,y,w)=(x+y)w$$

$$q=x+y \rightarrow \frac{\partial q}{\partial x}=1,\ \frac{\partial q}{\partial y}=1 \qquad f=qw \rightarrow \frac{\partial f}{\partial q}=w,\ \frac{\partial f}{\partial w}=q$$

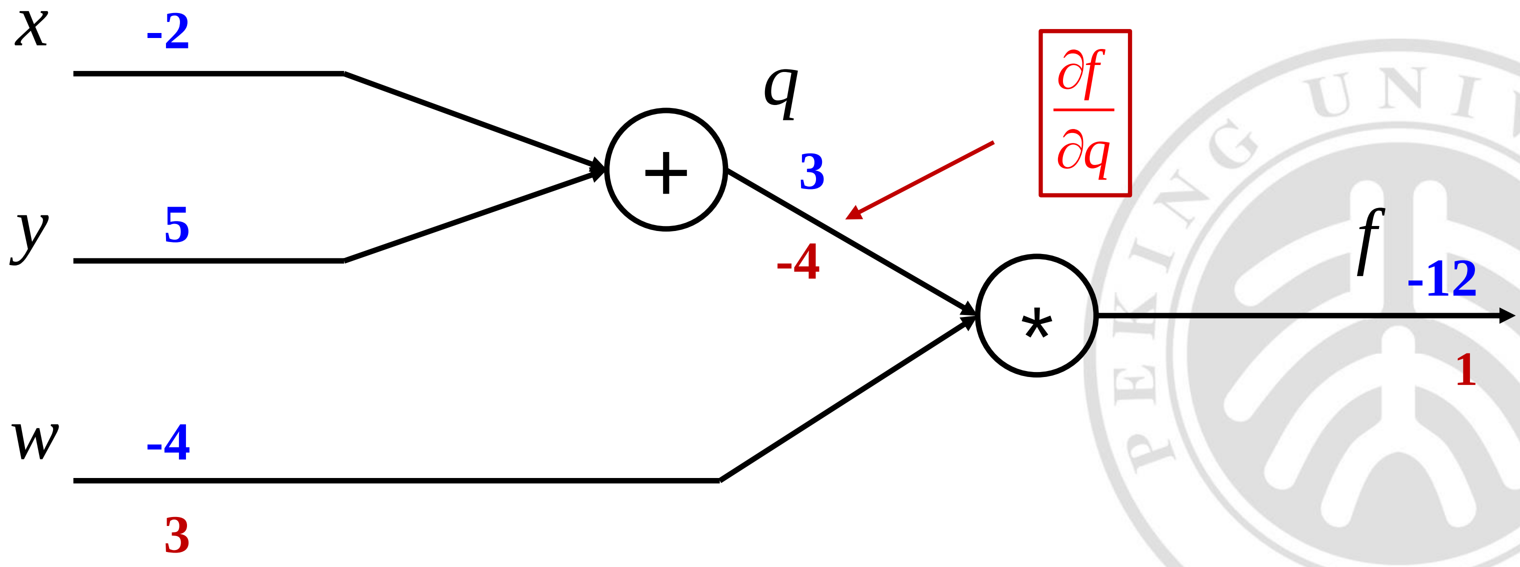


复合函数的求导

- 复合函数可以表示为一个有向无环图

$$f(x, y, w) = (x + y)w$$

$$q = x + y \rightarrow \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1 \qquad f = qw \rightarrow \frac{\partial f}{\partial q} = w, \frac{\partial f}{\partial w} = q$$



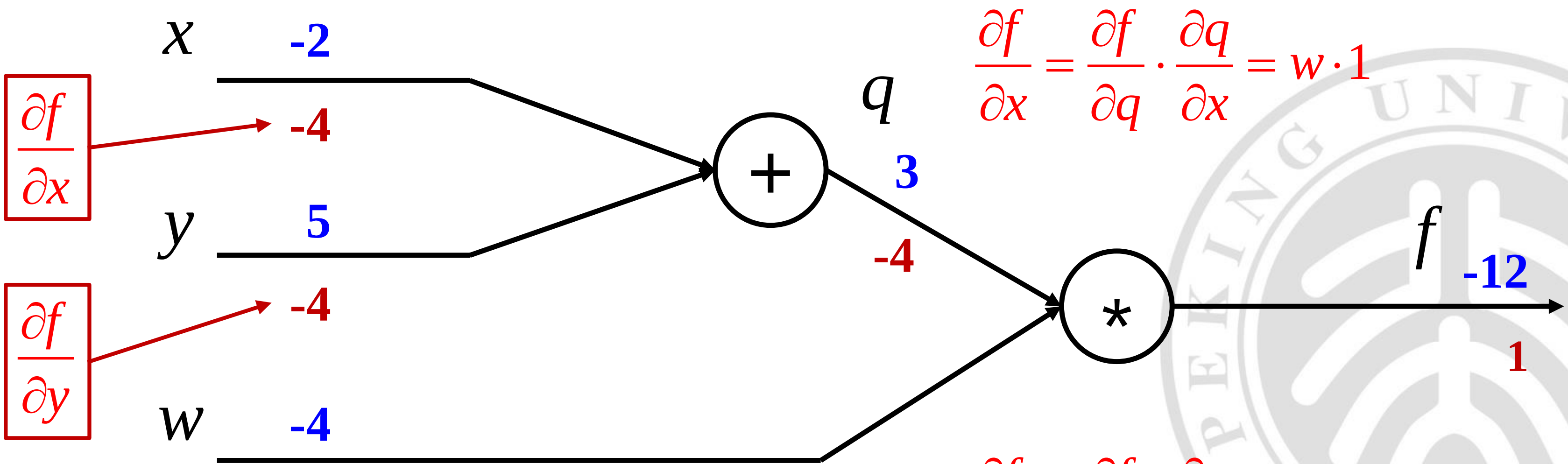
复合函数的求导

- 复合函数可以表示为一个有向无环图

$$f(x, y, w) = (x + y)w$$

$$q = x + y \;\rightarrow\; \frac{\partial q}{\partial x} = 1,\ \frac{\partial q}{\partial y} = 1 \qquad\qquad f = qw \rightarrow \frac{\partial f}{\partial q} = w,\ \frac{\partial f}{\partial w} = q$$

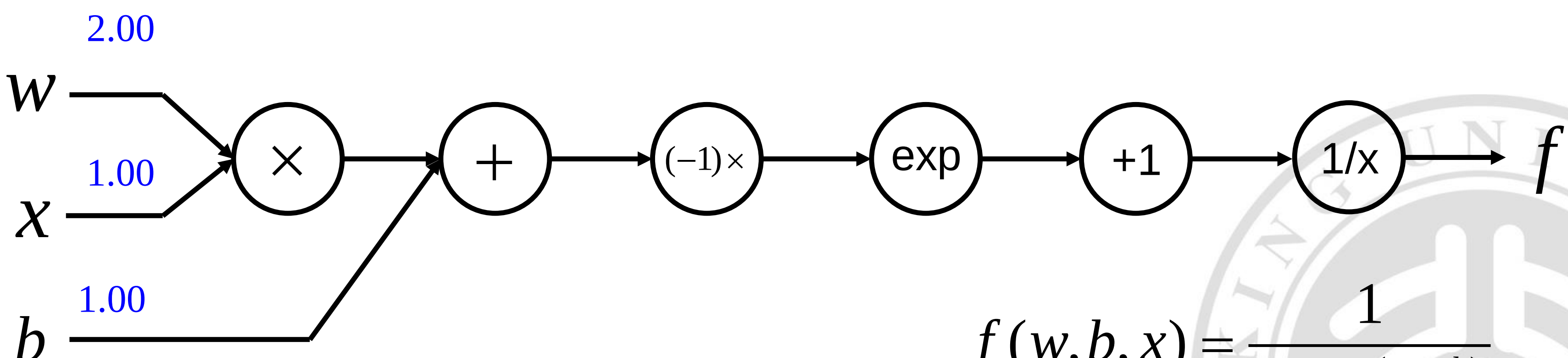
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \cdot \frac{\partial q}{\partial x} = w \cdot 1$$



$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \cdot \frac{\partial q}{\partial y} = w \cdot 1$$

复合函数的求导

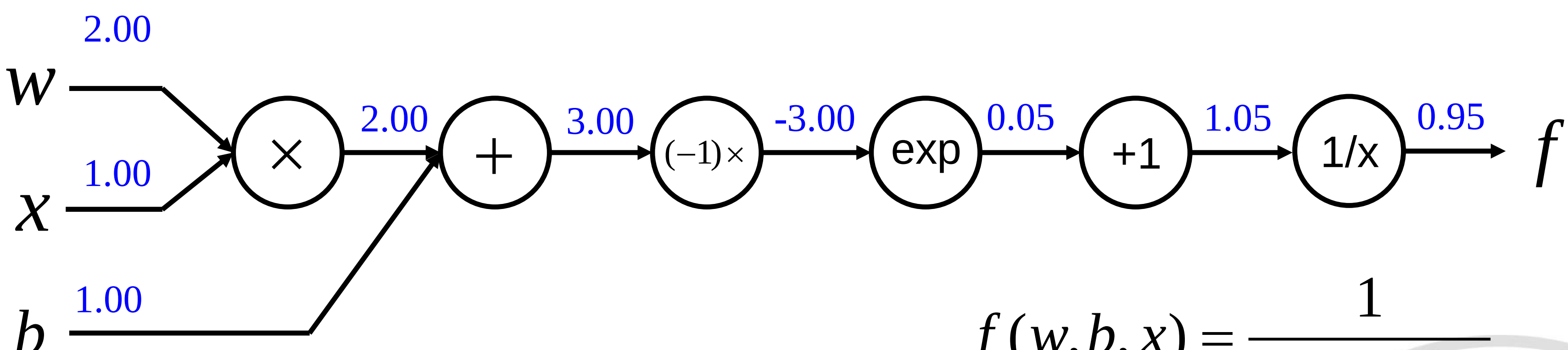
- 一个稍微复杂的例子
  - 假设关于输入 x 和参数 w, b 的函数
  - 求函数关于 x, w, b 的偏导数 $\left.\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial w}, \frac{\partial f}{\partial b}\right)\right|_{x=1,w=2,b=1}$



$$f(w,b,x) = \frac{1}{1+e^{-(wx+b)}}$$

复合函数的求导

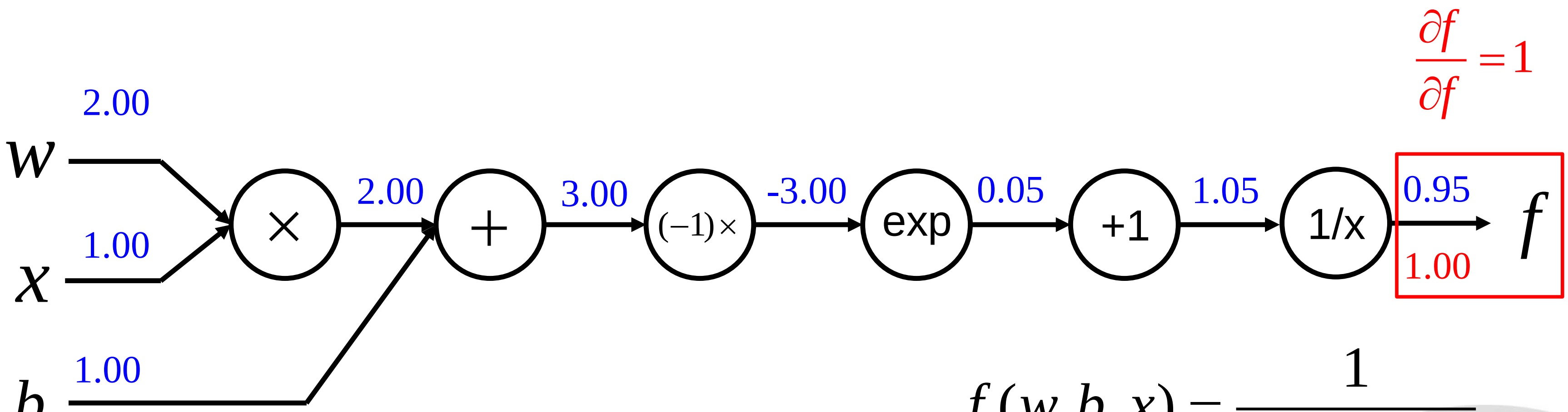
- 给定输入，计算各节点输出



$$f(w,b,x)=\frac{1}{1+e^{-(wx+b)}}$$

复合函数的求导

- 从后往前，最后一步偏导始终为1



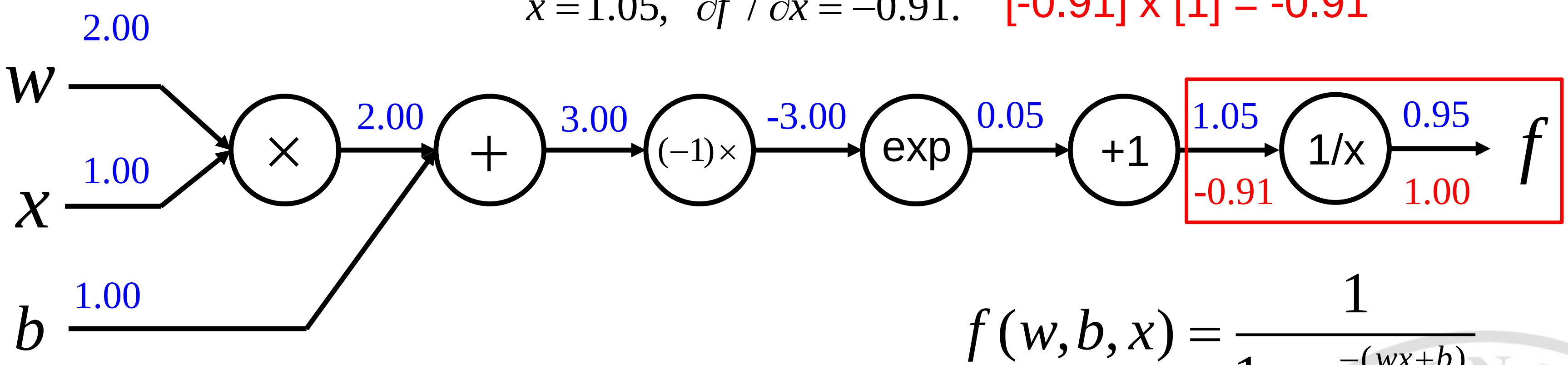
$$f(w,b,x)=\frac{1}{1+e^{-(wx+b)}}$$

复合函数的求导

- 沿着计算路径往前计算

$f = 1/x,\quad \partial f / \partial x = -1/x^2,$ [局部操作梯度] x [反传梯度]

$x = 1.05,\quad \partial f / \partial x = -0.91.$ [-0.91] x [1] = -0.91



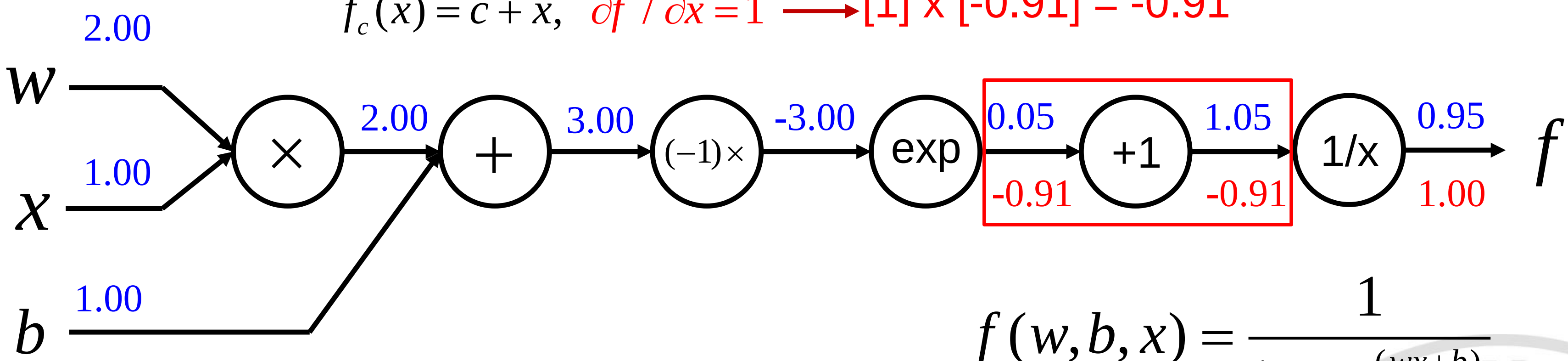
$$f(w,b,x) = \frac{1}{1+e^{-(wx+b)}}$$

复合函数的求导

- 沿着计算路径往前计算

[局部操作梯度] x [反传梯度]

$f_c(x) = c + x$, $\partial f / \partial x = 1$ → [1] x [-0.91] = -0.91

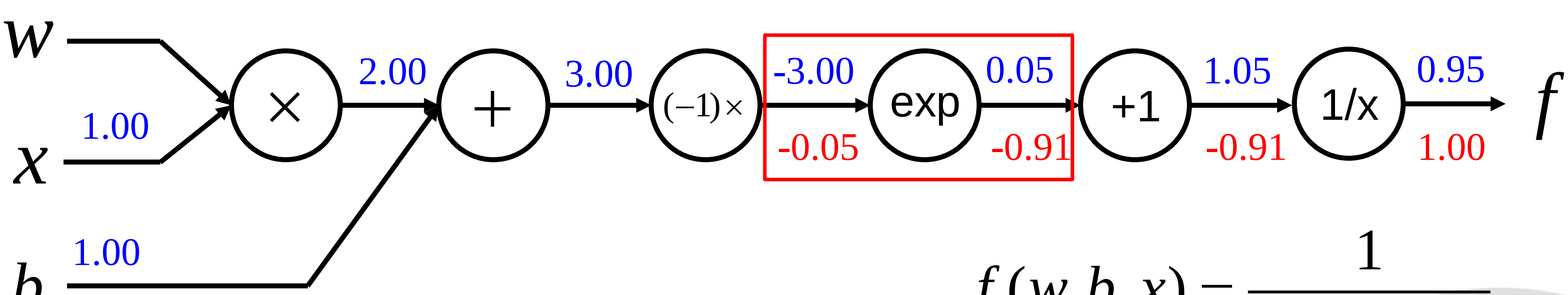


$$f(w,b,x) = \frac{1}{1+e^{-(wx+b)}}$$

- 沿着计算路径往前计算

[局部操作梯度] x [反传梯度]

$f = e^x, \ \partial f / \partial x = e^x$ ⟶ [e^(-3)] x [-0.91] = -0.05

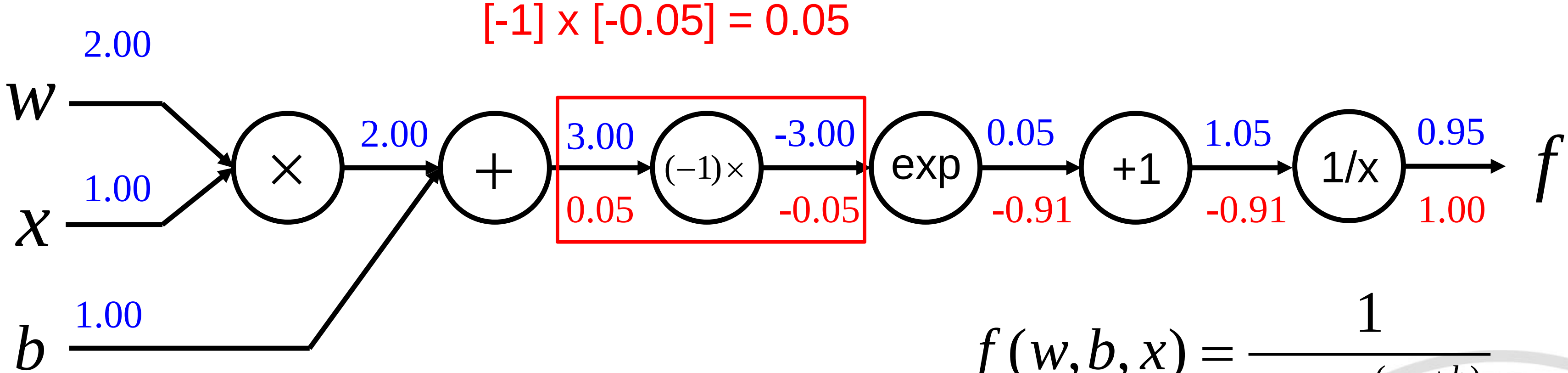


$$f(w,b,x) = \frac{1}{1+e^{-(wx+b)}}$$

复合函数的求导

- 沿着计算路径往前计算

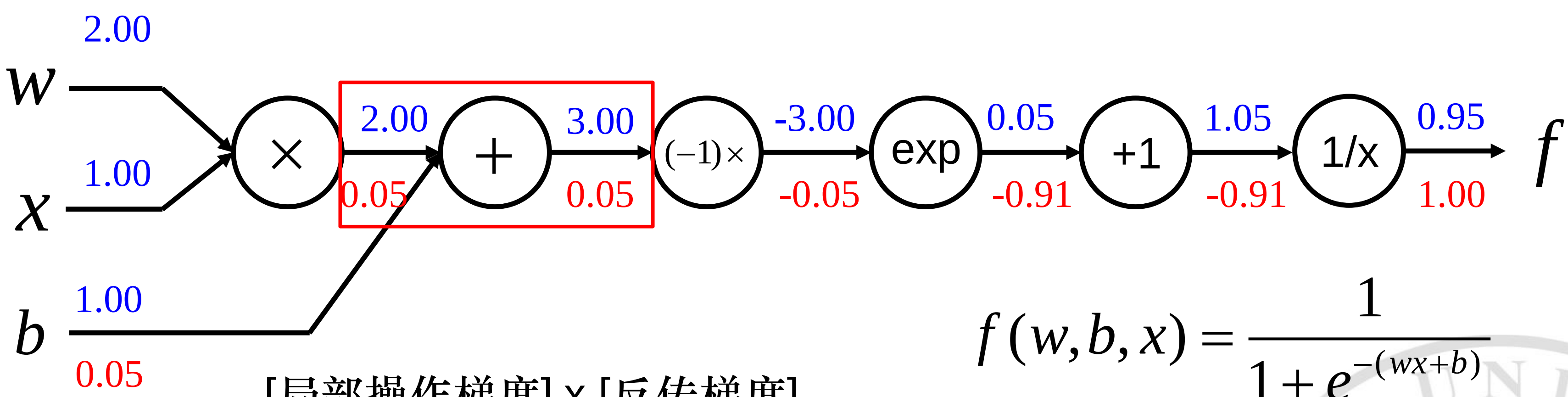
[局部操作梯度] x [反传梯度]
[-1] x [-0.05] = 0.05



$$f(w,b,x)=\frac{1}{1+e^{-(wx+b)}}$$

复合函数的求导

- 沿着计算路径往前计算



$$f(w,b,x)=\frac{1}{1+e^{-(wx+b)}}$$

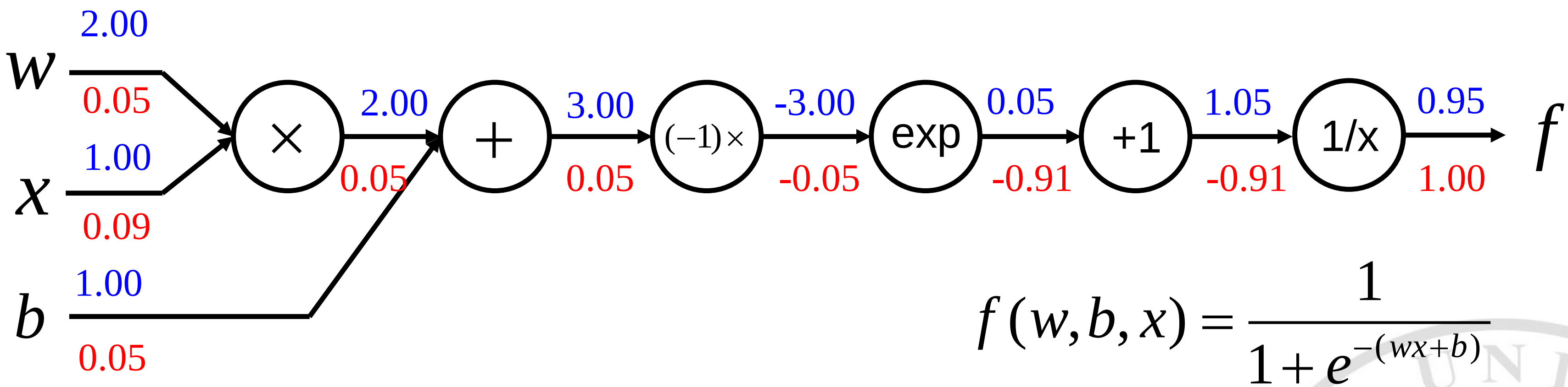
[局部操作梯度] x [反传梯度]

[1] x [0.05] = 0.05

[1] x [0.05] = 0.05

复合函数的求导

- 最终到达输入节点，获得梯度



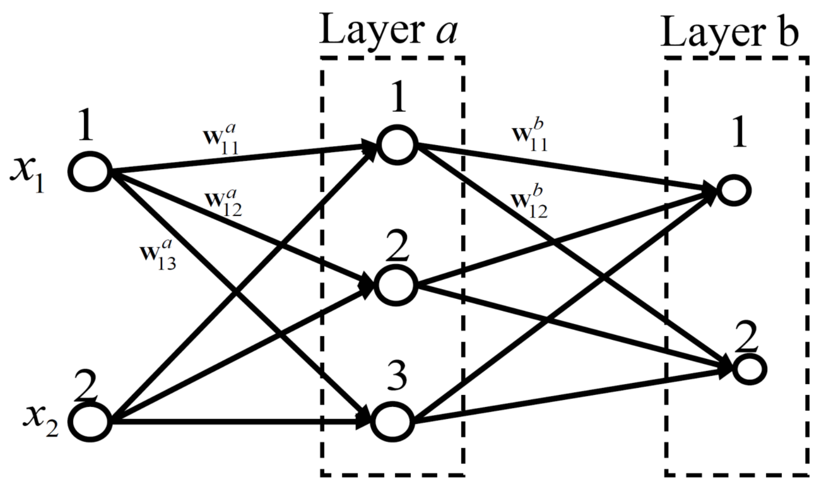
$$f(w,b,x)=\frac{1}{1+e^{-(wx+b)}}$$

多层神经元的训练

- 链式求导法则

$$\frac{\partial g(f(x,y))}{\partial x} = \frac{\partial g}{\partial f}\frac{\partial f}{\partial x}$$

- 神经元网络可以表达为一个复合函数



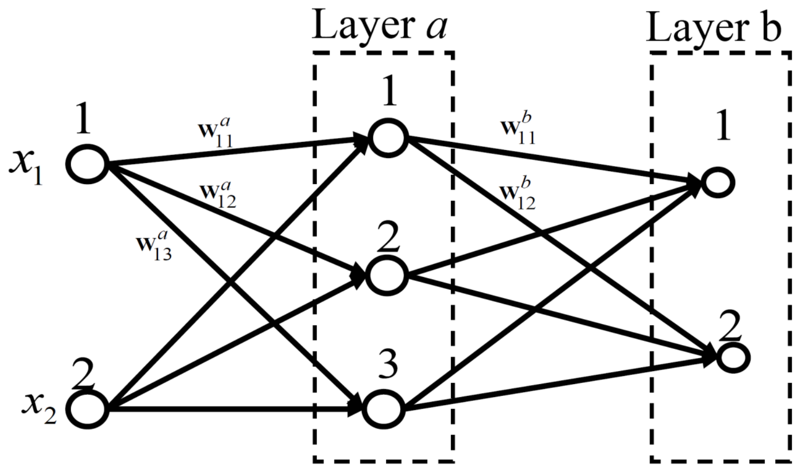
$$y_j^a = \sum_i \mathbf{W}_{ij}{}^a \cdot x_i + b_j^a, \quad j = 1,2,3$$

$$\sigma(x) = \frac{1}{1+e^{(-x)}}, \quad z_j^a = \sigma(y_j^a)$$

$$y_j^b = \sum_i \mathbf{W}_{ij}{}^b \cdot z_i^a + b_j^b, \quad j = 1,2$$

多层神经元的训练：例子

- 假设一个 **2层** 的神经元网络



$$y_j^a = \sum_i \mathbf{W}_{ij}{}^a \cdot x_i + b_j^a, \quad j = 1,2,3$$

$$\sigma(x) = \frac{1}{1+e^{(-x)}}, \quad z_j^a = \sigma(y_j^a)$$

$$y_j^b = \sum_i \mathbf{W}_{ij}{}^b \cdot z_i^a + b_j^b, \quad j = 1,2$$

- 假设输入 $x_1 = 1, x_2 = 1$ ，目标值 $Y_1 = 0, Y_2 = 1$
- 损失函数MSE $\mathcal{L} = \frac{1}{2}[(y_1^b - Y_1)^2 + (y_2{}^b - Y_2)^2]$
- 如何计算各个权重的梯度，实现网络的训练？

多层神经元的训练：例子

- 步骤一：随机给定权值参数的初始值

W_{ij}^{a}

	j=1	j=2	j=3
i=1	1	0.5	2
i=2	-1	0.5	3

W_{ij}^{b}

	j=1	j=2
i=1	1	2
i=2	1	3
i=3	-1	-2

b_j^a

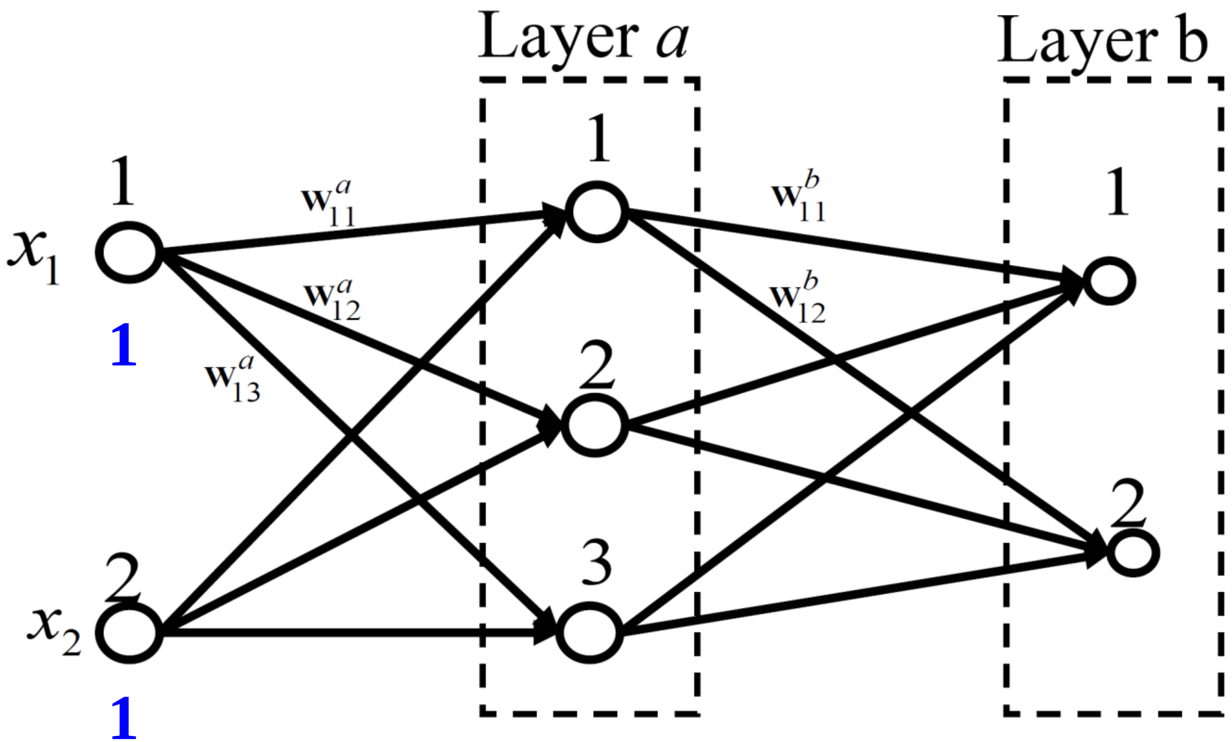
j = 1	j = 2	j = 3
0	2	-2

b_j^b

j = 1	j = 2
1	-1

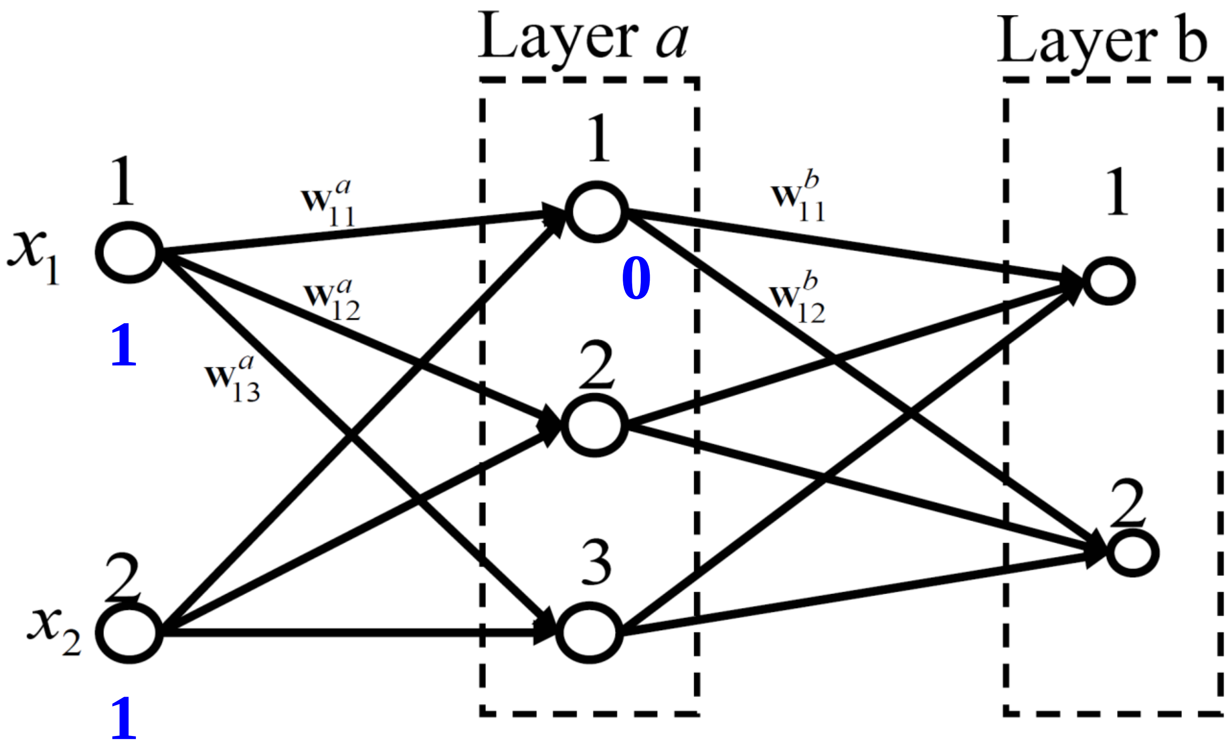
多层神经元的训练：例子

- 步骤二：计算当前网络的预测输出值



多层神经元的训练: 例子

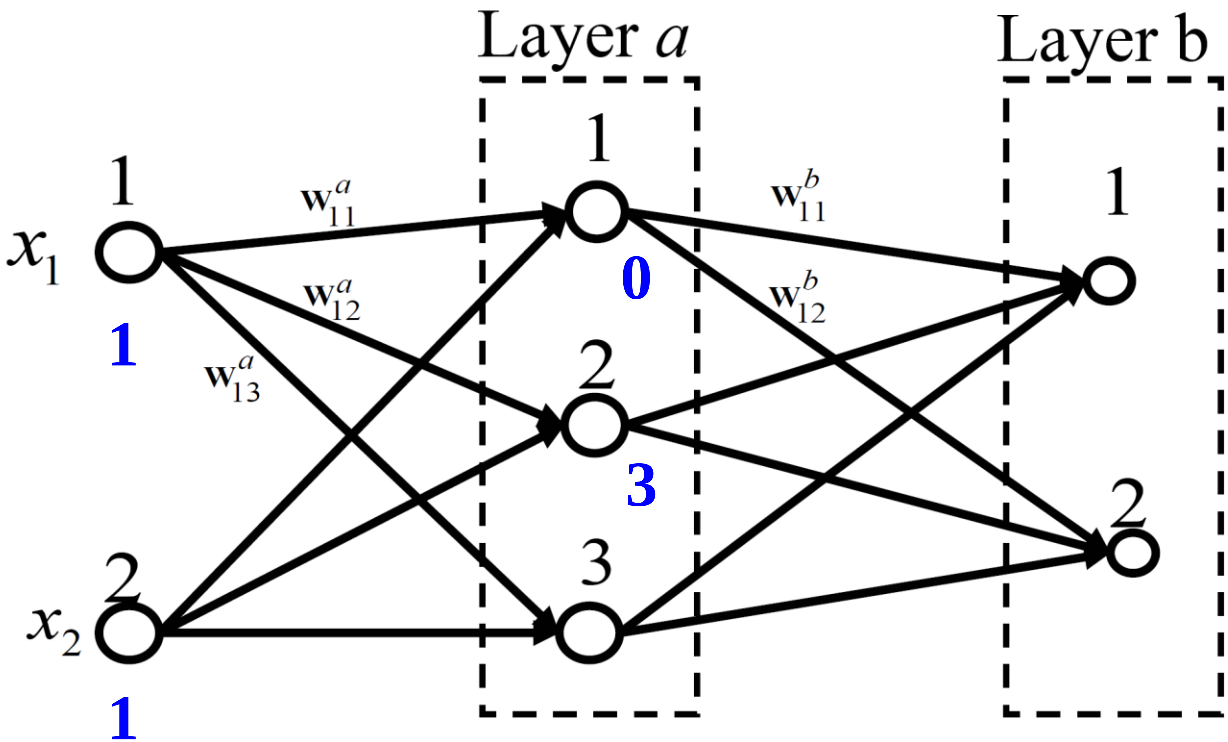
- 步骤二: 计算当前网络的预测输出值



$$(w_{11}, w_{2,1}) \cdot (x_1, x_2) + b_1$$
$$= (1,-1) \cdot (1,1) + 0 = 0$$

多层神经元的训练: 例子

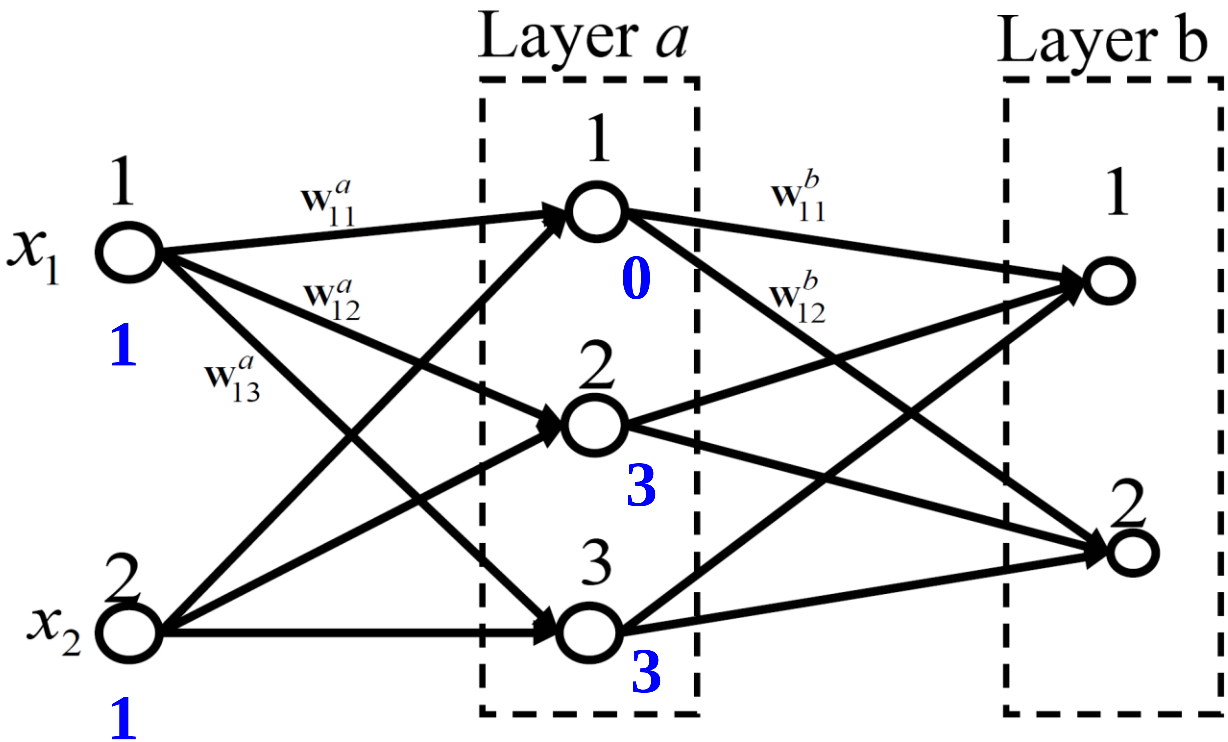
- 步骤二: 计算当前网络的预测输出值



$$(0.5, 0.5) \cdot (1, 1) + 2 = 3$$

多层神经元的训练：例子

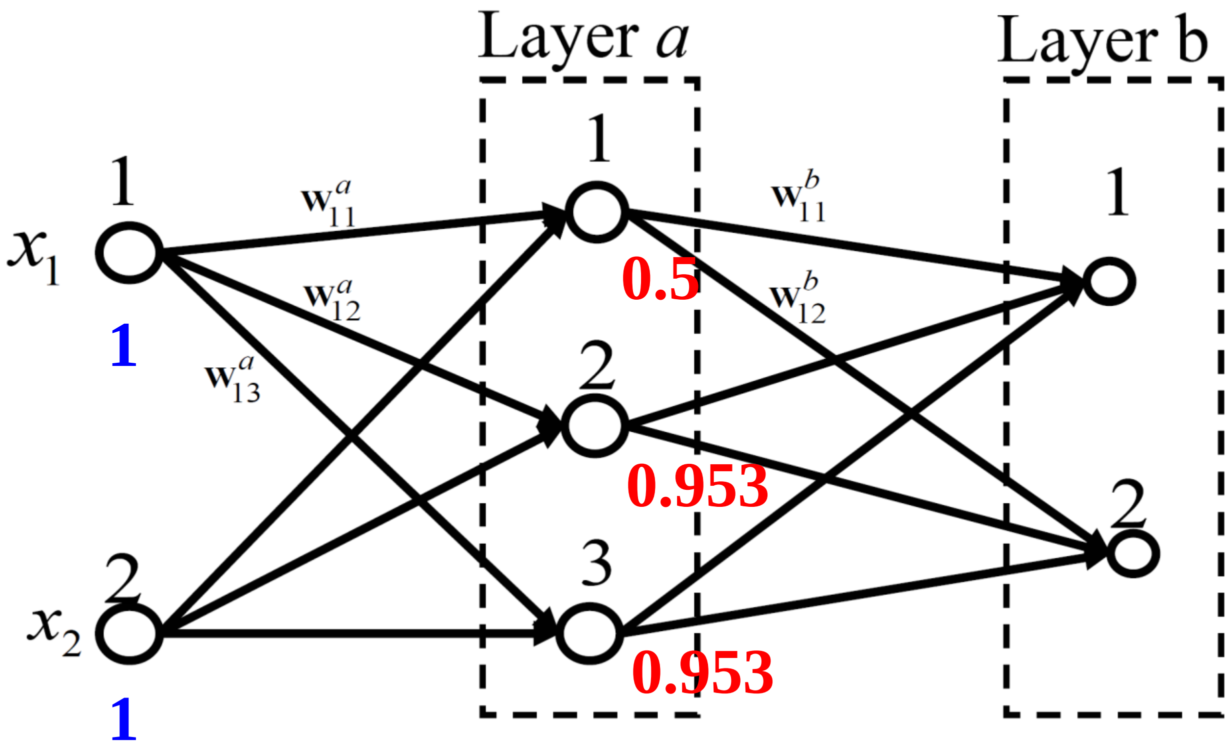
- 步骤二：计算当前网络的预测输出值



$$(2,3)\cdot(1,1)+(-2)=3$$

多层神经元的训练: 例子

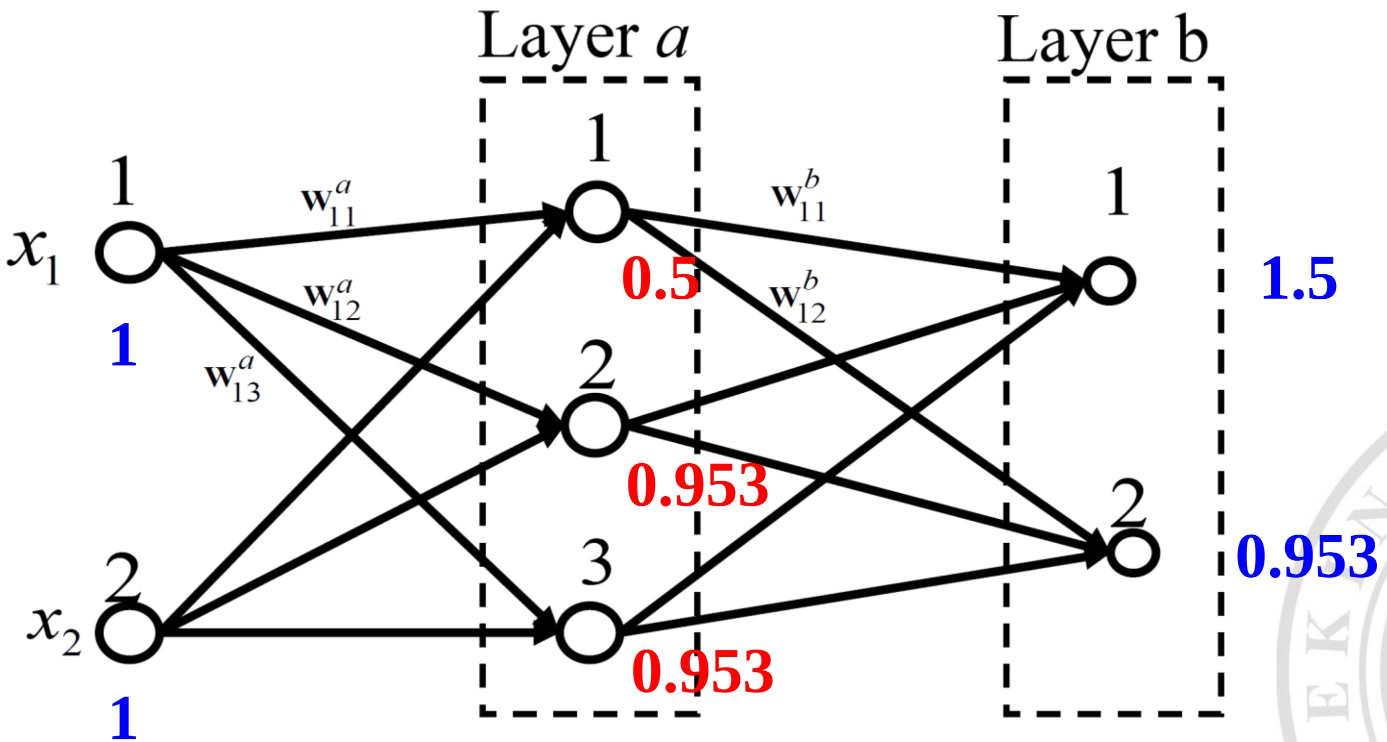
- 步骤二: 计算当前网络的预测输出值



$$\mathbf{z}^a = \sigma(\mathbf{y}^a) = (0.5, 0.953, 0.953)$$

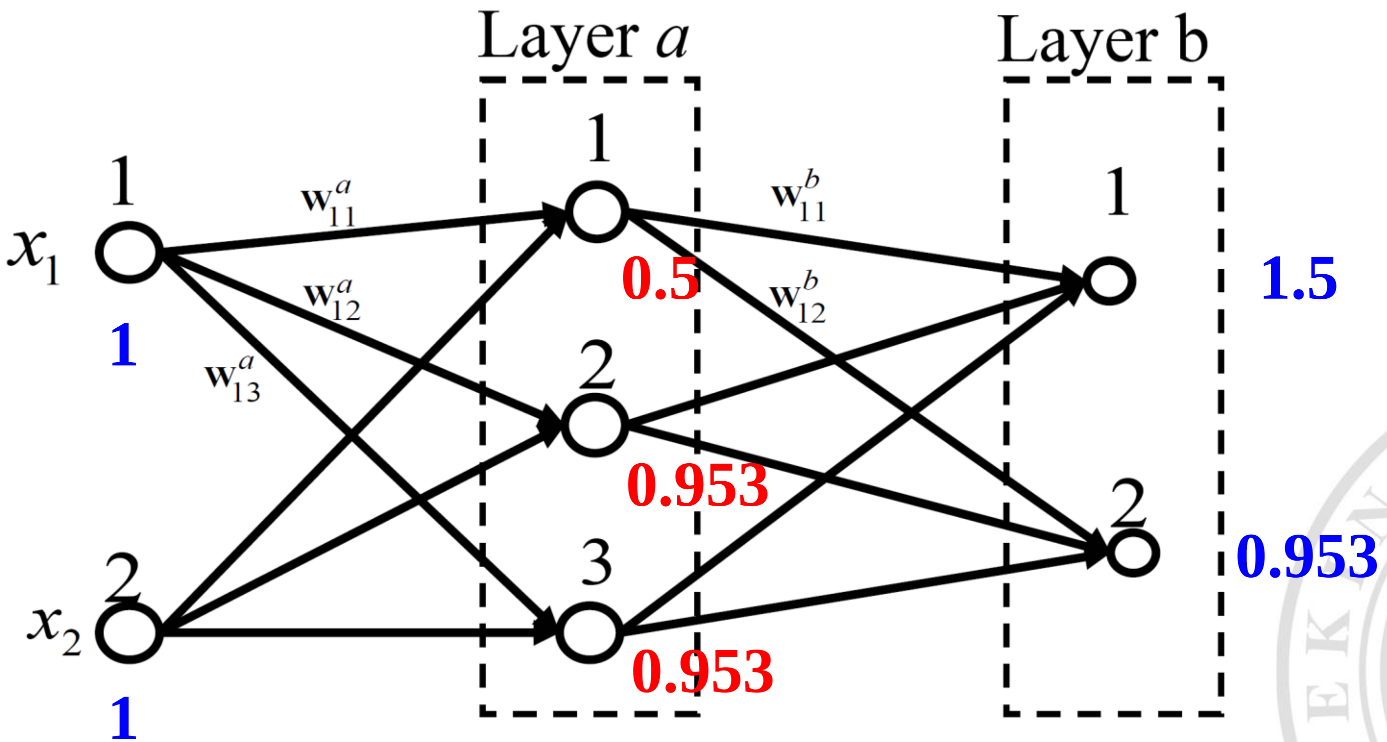
多层神经元的训练: 例子

- 步骤二: 计算当前网络的预测输出值



多层神经元的训练：例子

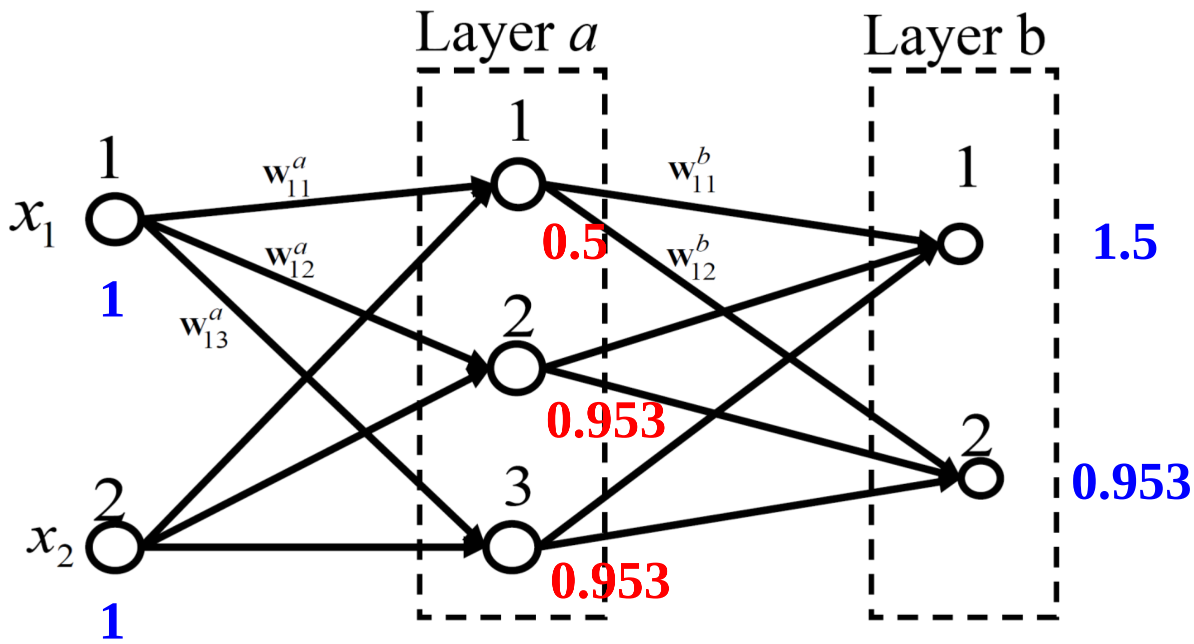
- 步骤三：计算损失函数值



$$\mathcal{L} = \frac{1}{2}[(y_1^b - Y_1)^2 + (y_2^b - Y_2)^2] = \frac{1}{2}[(1.5-0)^2 + (0.953-1)^2] = 1.126$$

多层神经元的训练：例子

- 步骤四：梯度计算和反向传播 (Back Propagation)



$$\mathcal{L} = \frac{1}{2}[(y_1^b - Y_1)^2 + (y_2^b - Y_2)^2]$$

$$\frac{\partial \mathcal{L}}{\partial y_1^{\,b}} = y_1^b - Y_1 = 1.5, \qquad \frac{\partial \mathcal{L}}{\partial y_2^{\,b}} = y_2^b - Y_2 = -0.0474$$

多层神经元的训练: 例子

- 步骤四: 梯度计算和反向传播 (Back Propagation)

$$y_1^b = w_{11}^b z_1^a + w_{21}^b z_2^a + w_{31}^b z_3^a + b_1^b$$

$$y_2^b = w_{12}^b z_1^a + w_{22}^b z_2^a + w_{32}^b z_3^a + b_2^b$$

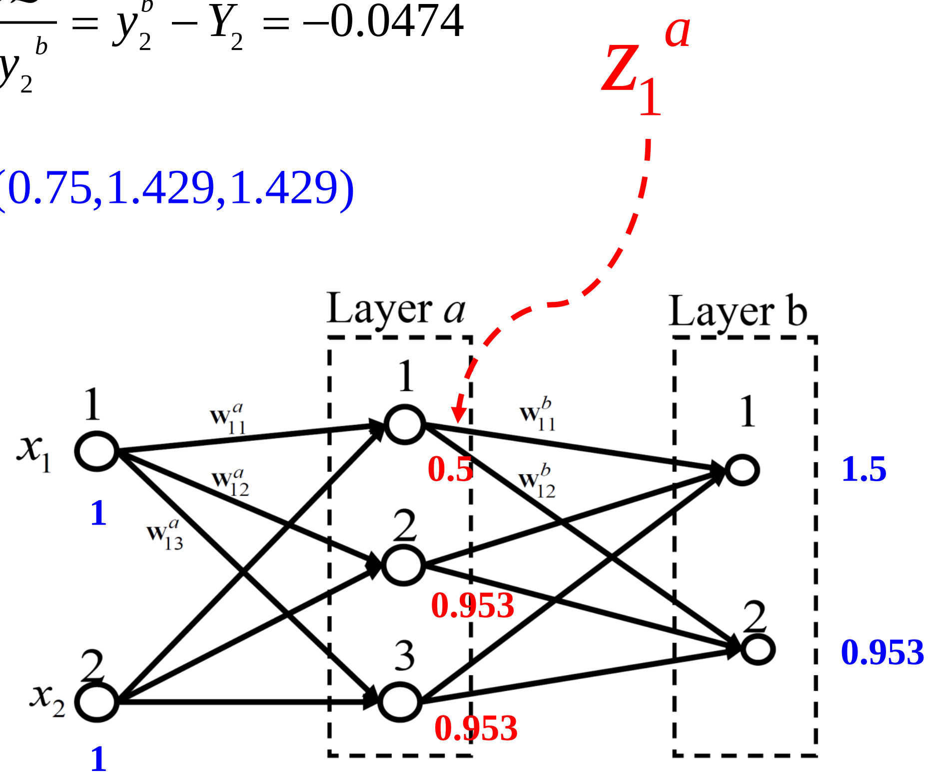
上一页可知: $\frac{\partial \mathcal{L}}{\partial y_1^b} = y_1^b - Y_1 = 1.5,\quad \frac{\partial \mathcal{L}}{\partial y_2^b} = y_2^b - Y_2 = -0.0474$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_{\_1}} = \frac{\partial \mathcal{L}}{\partial y_1^b}\frac{\partial y_1^b}{\partial \mathbf{w}_{\_1}} = (1.5)\cdot(z_1^a, z_2^a, z_3^a) = (0.75, 1.429, 1.429)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_{\_2}} = (-0.0237, -0.0452, -0.0452)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^b} = \left(\frac{\partial \mathcal{L}}{\partial y_1^b}, \frac{\partial \mathcal{L}}{\partial y_2^b}\right) = (1.5, -0.0474)$$

- 计算得到梯度 $\frac{\partial \mathcal{L}}{\partial \mathbf{w}_{\_1}}, \frac{\partial \mathcal{L}}{\partial \mathbf{w}_{\_2}}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}^b}$



多层神经元的训练: 例子

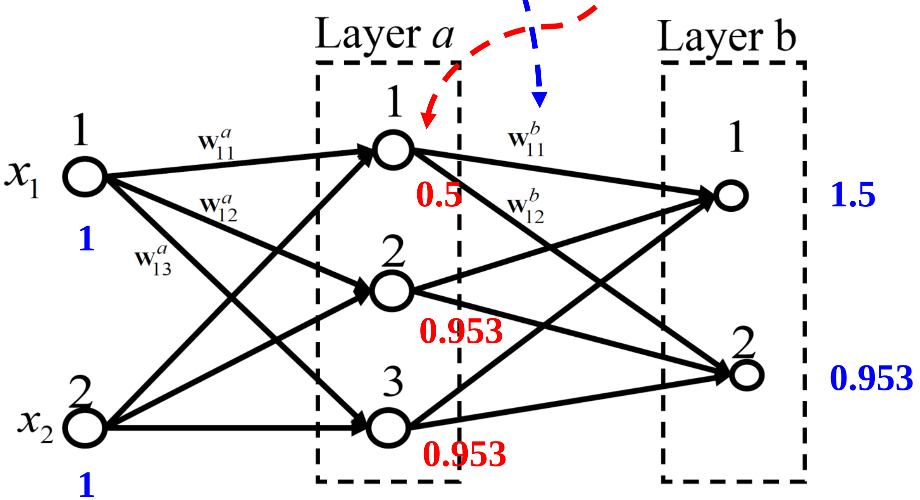
- 步骤四: 梯度计算和反向传播 (Back Propagation)
  - 我们现在得到了 Loss 关于第二层各个参数的梯度
  - 接下来计算关于 z^a 的梯度 → 以便继续反传
  - Loss 是 两个分别关于 y_1^b 和 y_2^b 的 复合函数
  - 根据多元复合函数求导法则

$$\frac{\partial \mathcal{L}}{\partial z_1^a} = \frac{\partial \mathcal{L}}{\partial y_1^b}\frac{\partial y_1^b}{\partial z_1^a} + \frac{\partial \mathcal{L}}{\partial y_2^b}\frac{\partial y_2^b}{\partial z_1^a}$$

$$= \frac{\partial \mathcal{L}}{\partial y_1^b} w_{11}^b + \frac{\partial \mathcal{L}}{\partial y_2^b} w_{12}^b$$

- 计算得到梯度 $\frac{\partial \mathcal{L}}{\partial z_1^a}$

z_1^a



多层神经元的训练：例子

- 步骤四：梯度计算和反向传播 (Back Propagation)
  - 现在获得了 *Loss* 关于第一层输出的梯度 $\dfrac{\partial \mathcal{L}}{\partial z_1^a}$
  - 接下来计算

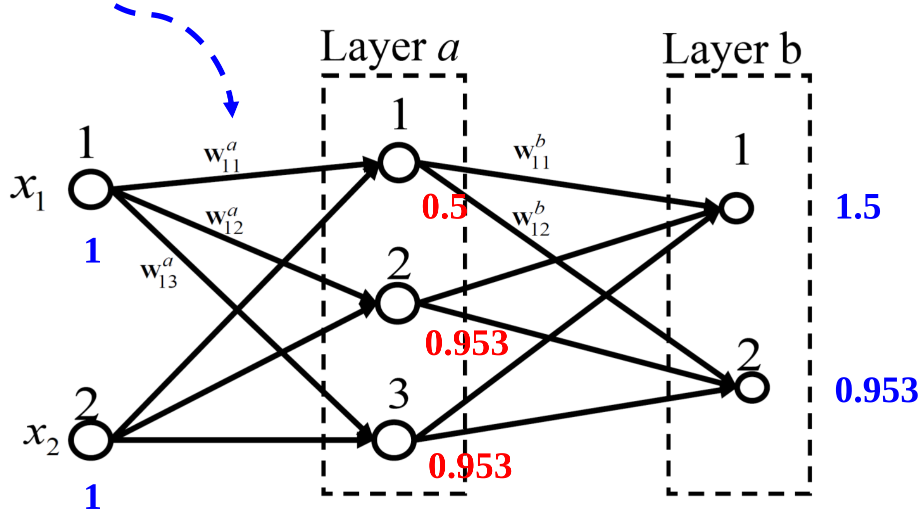
$$\frac{\partial \mathcal{L}}{\partial y_1^a} = \frac{\partial \mathcal{L}}{\partial z_1^a}\frac{\partial z_1^a}{\partial y_1^a}$$

$$\frac{\partial z_1^a}{\partial y_1^a} = \frac{\partial \dfrac{1}{1+e^{(-y_1^a)}}}{\partial y_1^a}$$

$$= -\frac{1}{[1+e^{(-y_1^a)}]^2}\cdot[-e^{(-y_1^a)}]$$

$$\boxed{= (z_1^a)^2 \cdot e^{(-y_1^a)}}$$

$$= (0.5)^2 \times 1$$

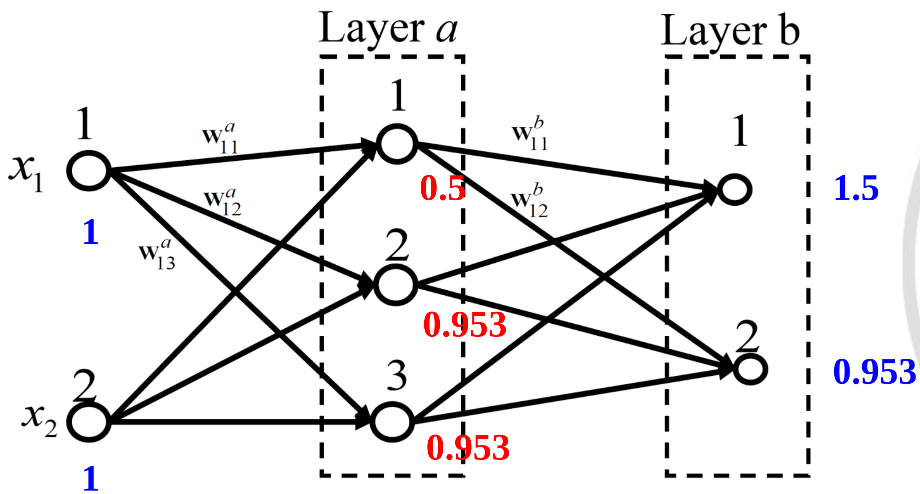


y_1^a 和 z_1^a 在前传过程中均可暂存下来

z_1^a 和 y_1^a 在前传过程中均可暂存下来

多层神经元的训练: 例子

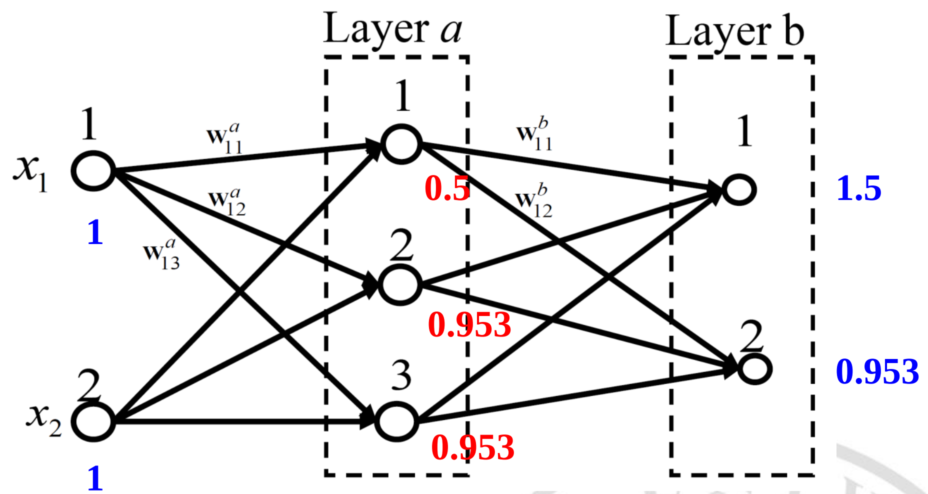
- 步骤四: 梯度计算和反向传播 (Back Propagation)
  - 现在得到了 $\dfrac{\partial \mathcal{L}}{\partial y_1^a}$
  - 与前一层 $\dfrac{\partial \mathcal{L}}{\partial y_1^b}$ 形式一致
  - 接下来可以采用同样的方式得到其它的梯度值



多层神经元的训练: 例子

- 步骤四: 梯度计算和反向传播 (Back Propagation)

$$\frac{\partial g(f(x,y))}{\partial x} = \frac{\partial g}{\partial f}\frac{\partial f}{\partial x}$$



网络总可以写成 复合函数

复合函数的每个部分均可导

故可对函数应用链式法则

实际中前向传播和反向传播均可写成矩阵形式

多层神经元的训练：例子

- 步骤四：梯度计算和反向传播 (Back Propagation)

$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{ij}^{a}}$

	j = 1	j = 2	j = 3
i = 1	0.3513	0.0613	-0.0635
i = 2	0.3513	0.0613	-0.0635

$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{ij}^{b}}$

	j = 1	j = 2
i = 1	0.75	-0.0237
i = 2	1.4289	-0.0452
i = 3	1.4289	-0.0452

$\frac{\partial \mathcal{L}}{\partial b_{j}^{a}}$

j = 1	j = 2	j = 3
0.3513	0.0613	-0.0634

$\frac{\partial \mathcal{L}}{\partial b_{j}^{b}}$

j = 1	j = 2
1.5	-0.0474

多层神经元的训练: 例子

- 步骤五: 按照一定的学习率 (步长) 更新参数

$$\mathbf{W} = \mathbf{W} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}}, \quad \text{assume } \eta = 0.1$$

$\hat{\mathbf{W}}_{ij}^{a}$

	j = 1	j = 2	j = 3
i = 1	0.965	0.494	2.006
i = 2	-1.035	0.494	3.006

$\hat{\mathbf{W}}_{ij}^{b}$

	j = 1	j = 2
i = 1	0.925	2.002
i = 2	0.857	3.005
i = 3	-1.143	-1.995

$\hat{b}_j^{a}$

j = 1	j = 2	j = 3
-0.035	1.994	-1.994

$\hat{b}_j^{b}$

j = 1	j = 2
0.85	-0.995

$$\hat{\mathcal{L}} = \frac{1}{2}[(y_1^b - Y_1)^2 + (y_2^b - Y_2)^2] = \frac{1}{2}[(1.014 - 0)^2 + (0.910 - 1)^2] = 0.518 \downarrow$$

神经网络的训练

- 随机**初始化**网络中的权值
- 计算当前网络的预测输出值 (**前向传播**)
- 计算 **损失函数** 值
- 计算损失函数值关于各权值的梯度 (**反向传播**)
- 按照一定的学习率**更新参数**

神经网络的训练

- 随机**初始化**网络中的权值
- 计算当前网络的预测输出值 (**前向传播**)
- 计算 **损失函数** 值
- 计算损失函数值关于各权值的梯度 (**反向传播**)
- 按照一定的学习率**更新参数**

为了使得网络能够满足需求，需要选择合适的损失函数

人工神经网络 — 小结

- 多层神经元组成神经元网络
- 能够拟合多种函数
- 损失函数表达神经网络训练的目标
- 随机梯度下降法求解关于参数的最优化问题
- 反向传播算法计算多层参数的梯度