《物理与人工智能》

8.神经网络与反向传播

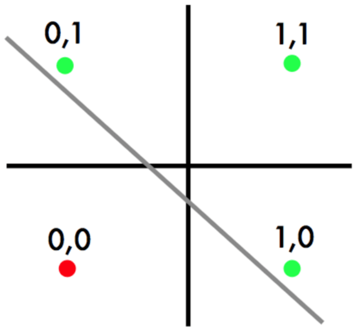
授课教师：马滟青

2025/10/13（第五周）

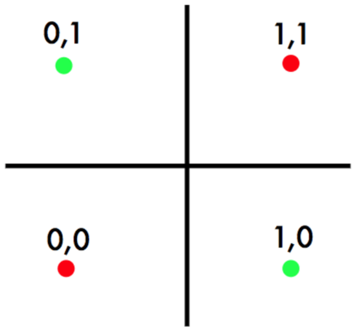
鸣谢：基于计算机学院《人工智能引论》课程组幻灯片

线性模型的局限

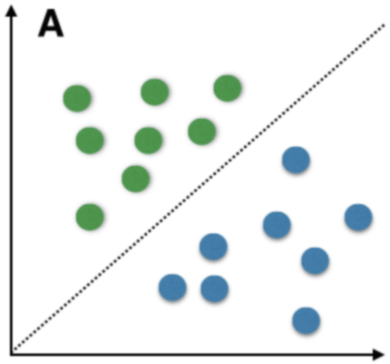
- 对分类问题，存在线性不可分 (linearly non-separable) 情况
  - 比如异或 (xor) 操作，不存在任何线性分类器可以拟合亦或
    - 导致1970年代第一次 AI 寒冬的主因之一
  - 对于较难的分类任务，如图像分类，简单的线性模型一般也无能为力
  - 树模型是一种引入非线性的方式



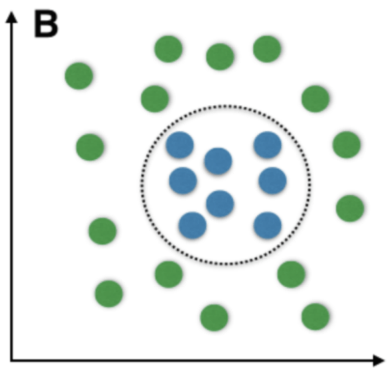
OR



XOR



线性可分



线性不可分

线性模型的局限

图像分类中，我们看到的图片如下

而计算机看到的图片如下:

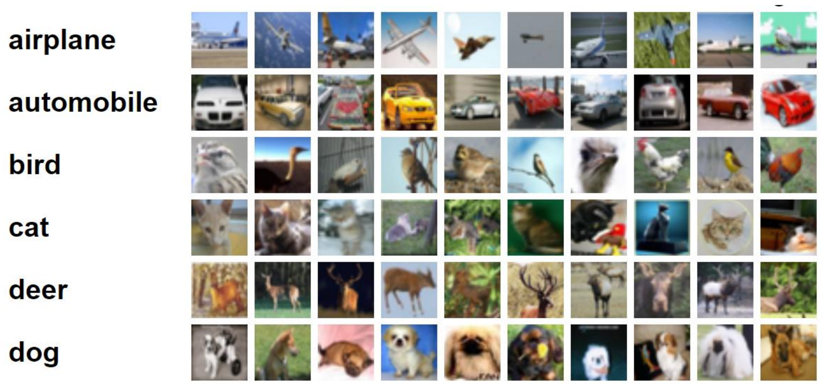
194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

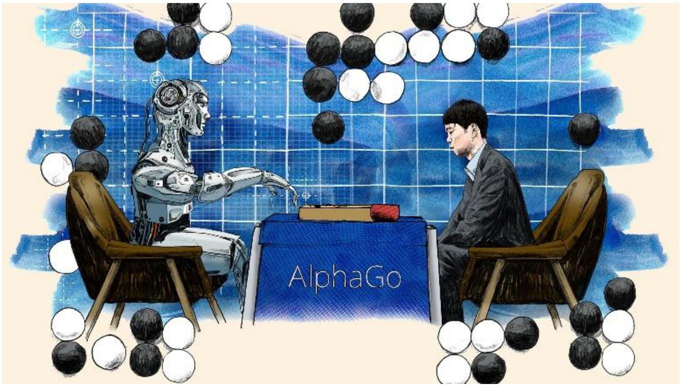
- 想要使用线性模型或树模型，需要提取一些高级特征，如门把手、轮胎、后视镜、车窗等
- 这些模型自身无法自动提取特征！

image from Andrew Ng

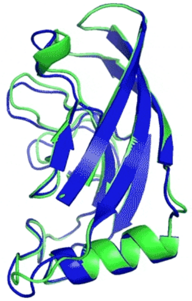
人工神经网络 (Artificial Neural Network)

- 想要模拟人脑神经网络的学习算法
- 80年代兴起，90年代没落，10年代重新兴起，为AI带来革命
- 也叫做：神经网络 (neural network)、深度学习 (deep learning)等

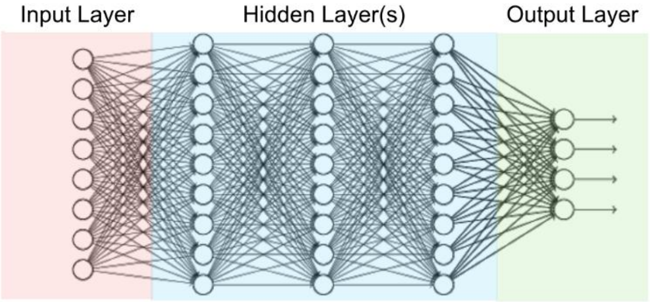




T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)

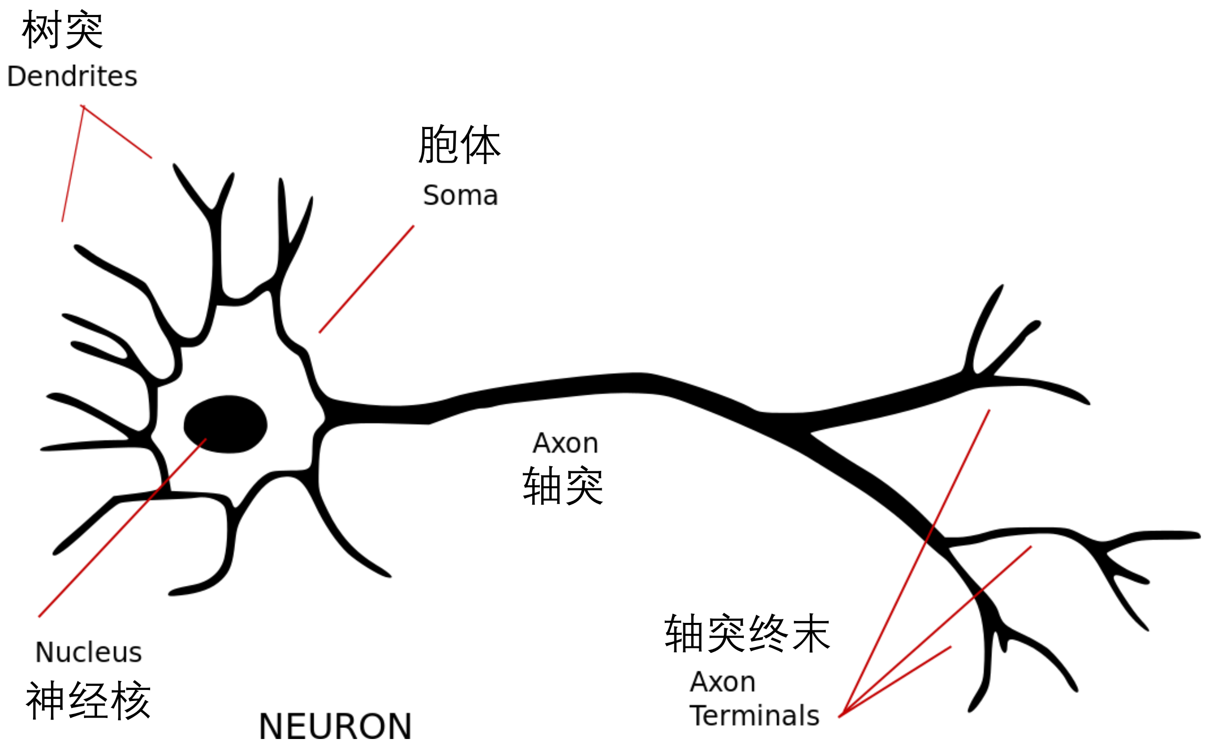


T1049 / 6y4f
93.3 GDT
(adhesin tip)

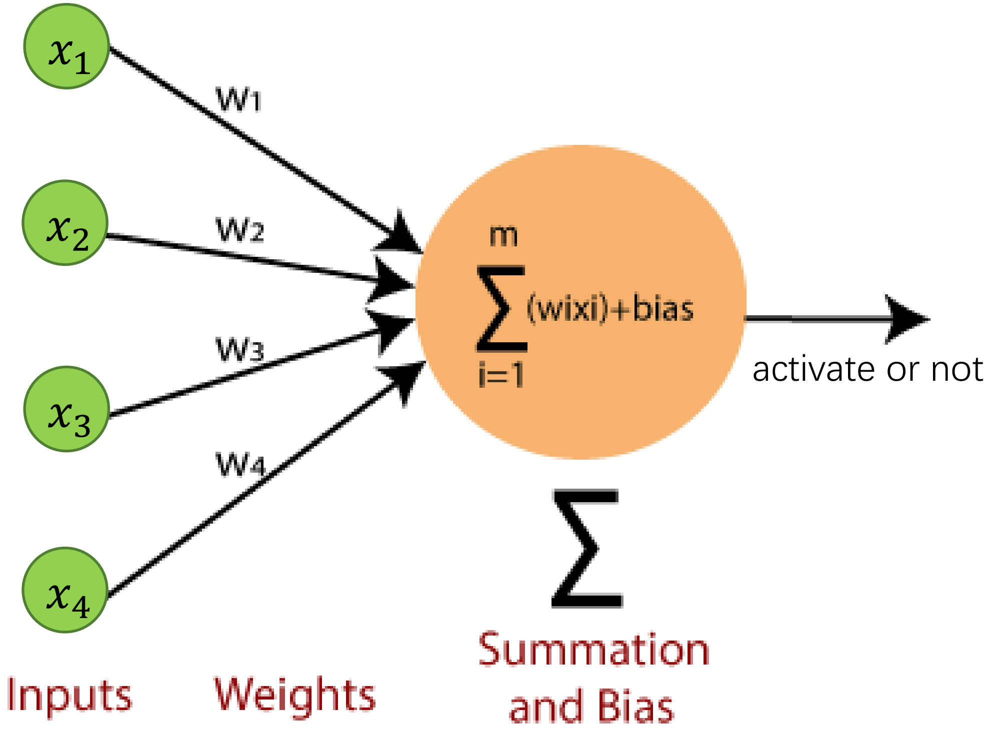


All powered by neural networks!

神经元 (Neuron) 与感知器 (Perceptron)



神经元



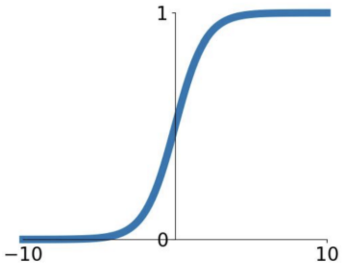
感知器

- 感知器是一个早期的线性分类模型，是对生物神经元的数学抽象
- 感知器可以理解为单层的人工神经网络
- 同理，线性回归、逻辑回归等也可以理解为单层的人工神经网络

常见激活函数

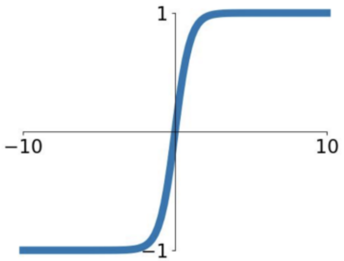
Sigmoid

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



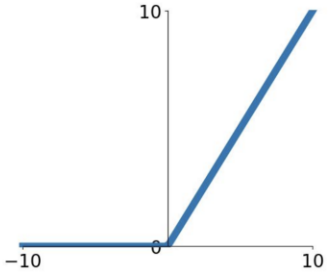
tanh

$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$$



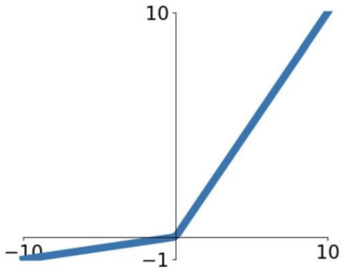
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

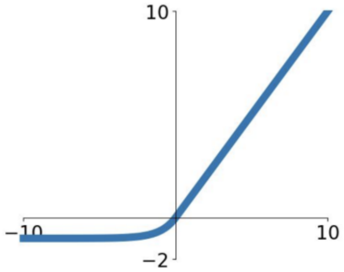


Maxout

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

ELU

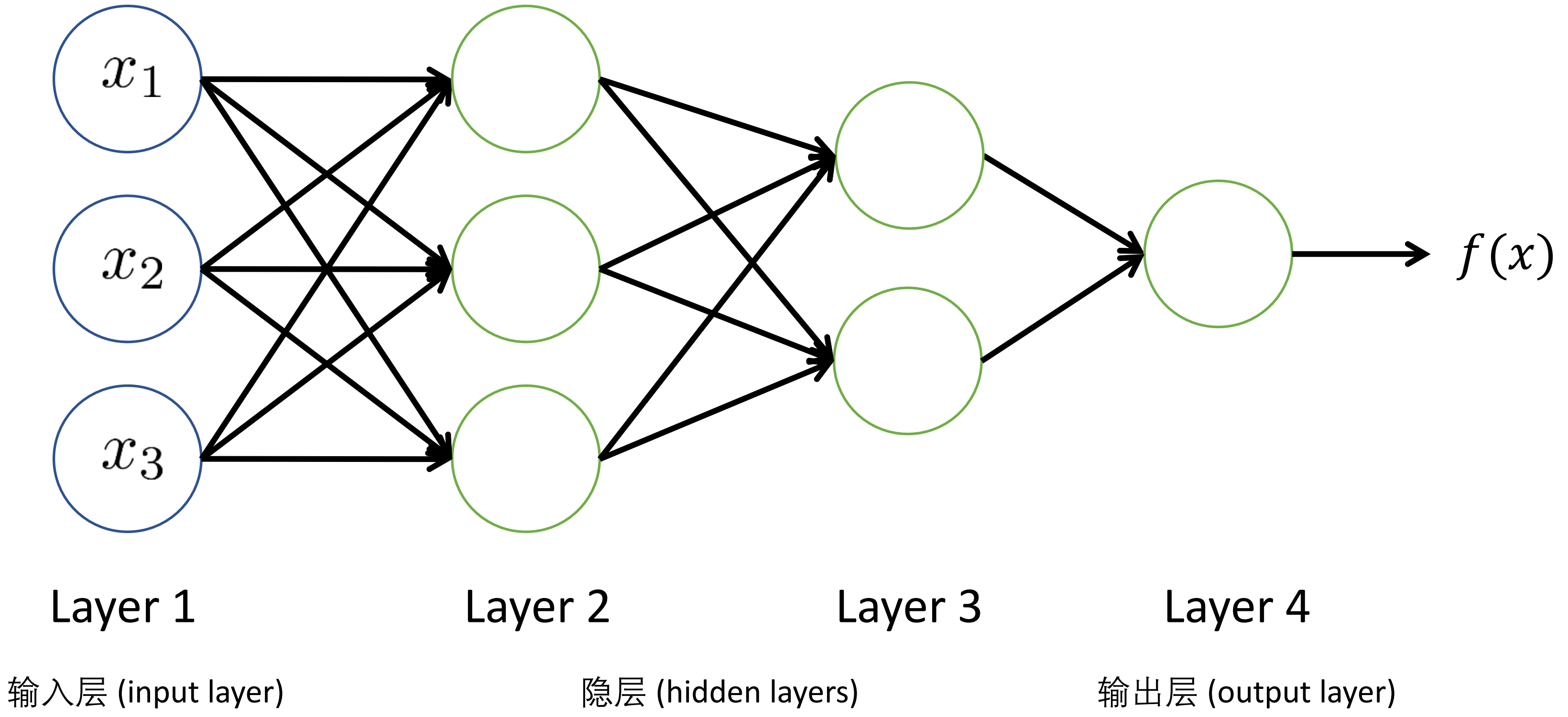
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



ReLU是很好的默认选择！在大部分常见问题上表现良好！

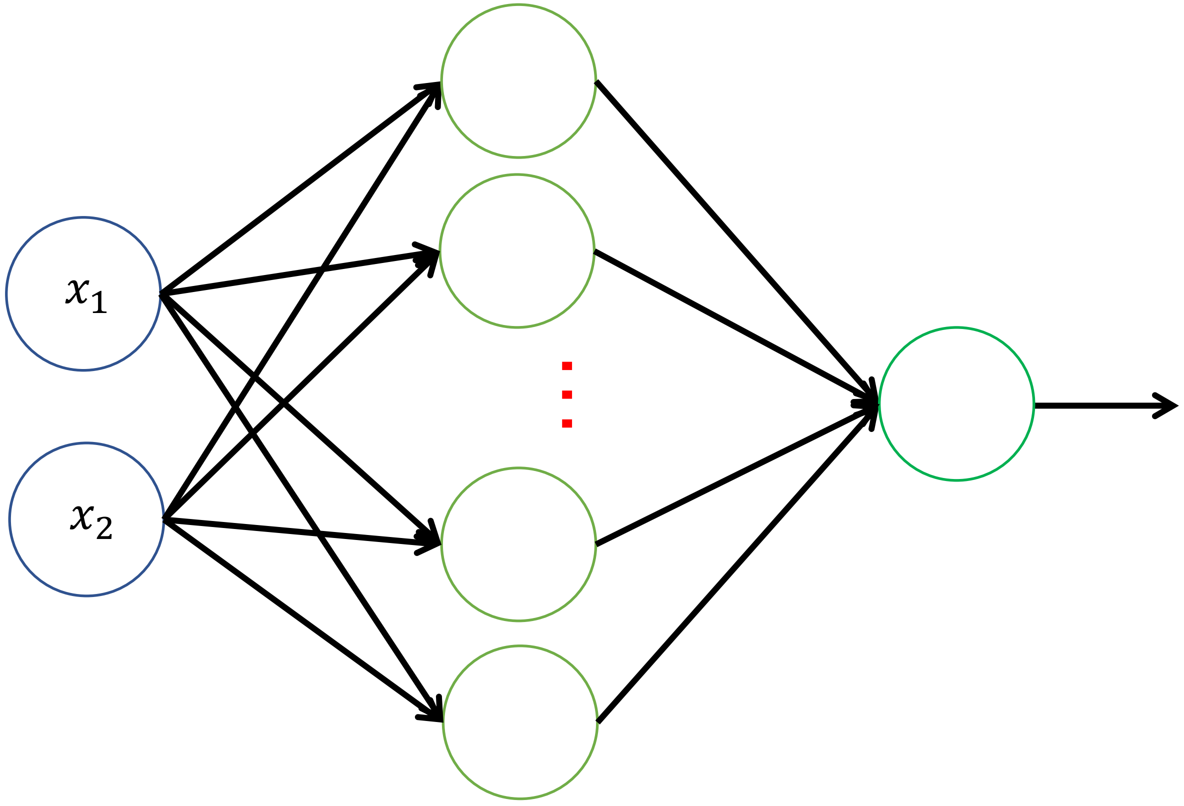
多层感知器 (Multi-Layer Perceptron)

- 多层感知器 (MLP)，也称为全连接神经网络 (fully connected neural network)，是一种结构最简单的多层神经网络
- 隐层层数、每层的神经元个数都可自由设计；一般随着任务的复杂程度或数据量提高，增加隐层的维度（即神经元个数）和神经网络的深度（层数）



万能逼近定理

定理：一个具有至少一个隐藏层的前馈神经网络，只要隐藏层中的神经元足够多，就可以逼近任何一个定义在有限维空间上的连续函数，精度可以任意接近。这个结果适用于使用非线性激活函数（如Sigmoid或ReLU）。

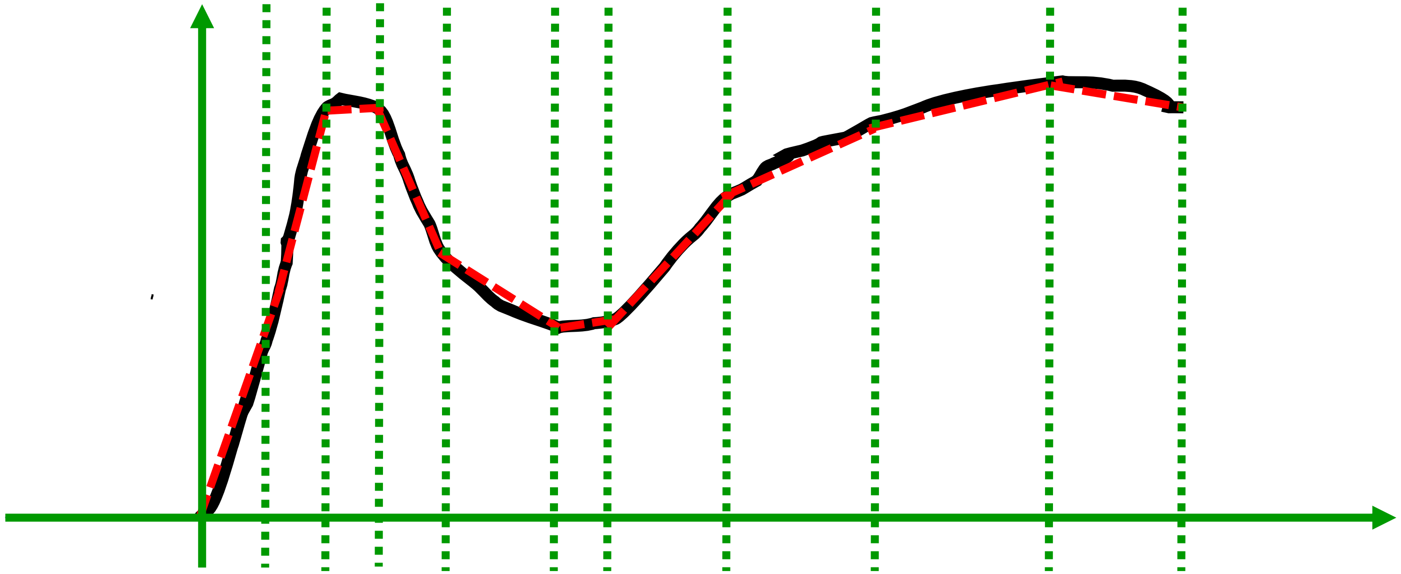


$$f_\theta(x) = \phi\left(\sum_{i=1}^{n} c_i\, \phi\left(w_i^T\, x + b_i\right) + c_0\right)$$

ϕ为激活函数

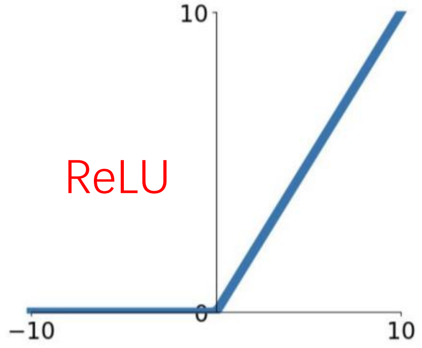
万能逼近定理

举例：对于单变量



- 折线可以无限逼近任意函数
- ReLU激活函数的单隐藏层网络可描述任意连续折线:

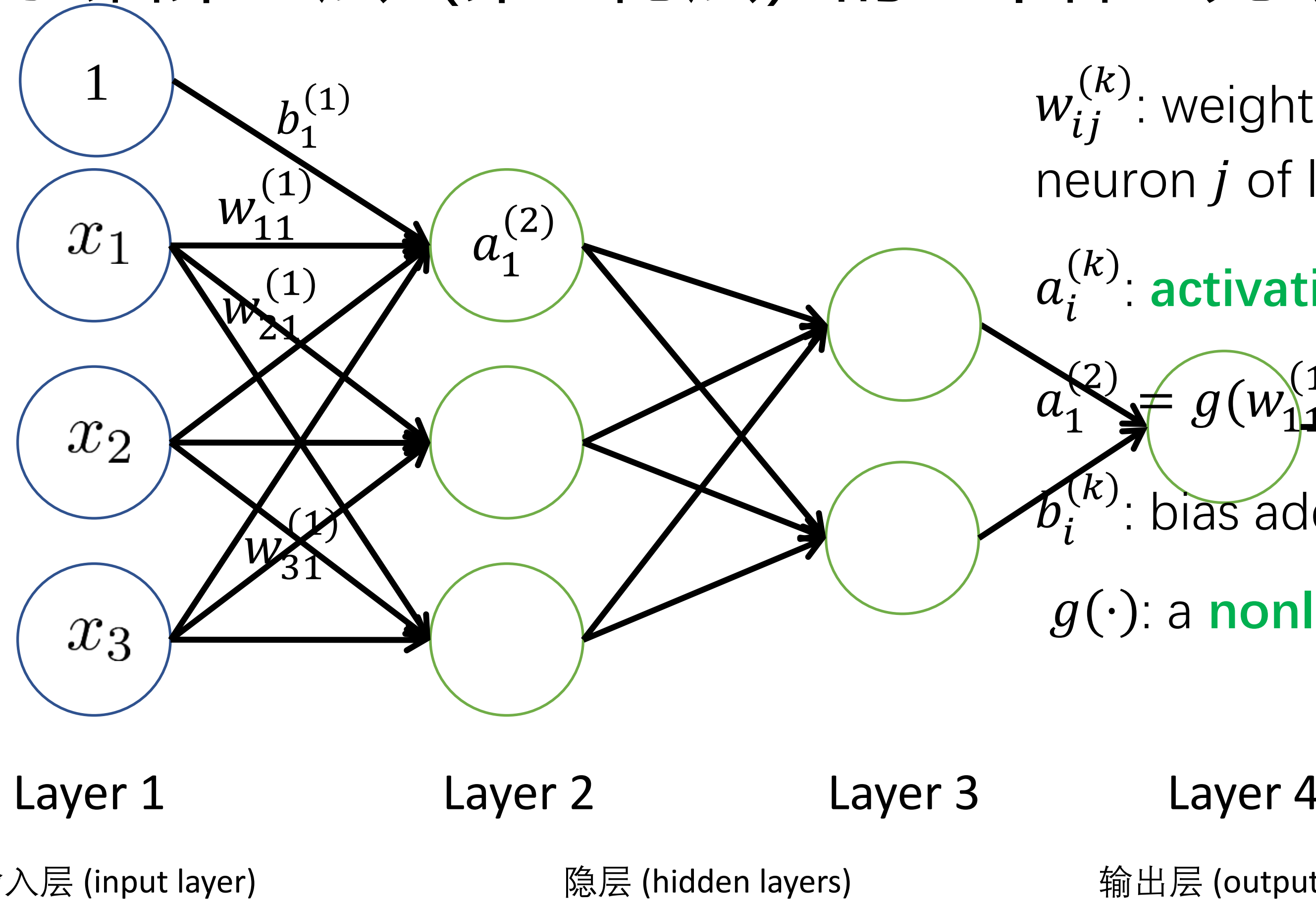
$$\phi(c\,\phi(x) + \cdots)$$



- 其它激活函数可以近似ReLU

正向传播

- 单独看第二层（第一隐层）的一个神经元 (neuron)



$w_{ij}^{(k)}$: weight connecting neuron i of layer k and neuron j of layer $k+1$

$a_i^{(k)}$: **activation (激活)** of neuron i at layer k

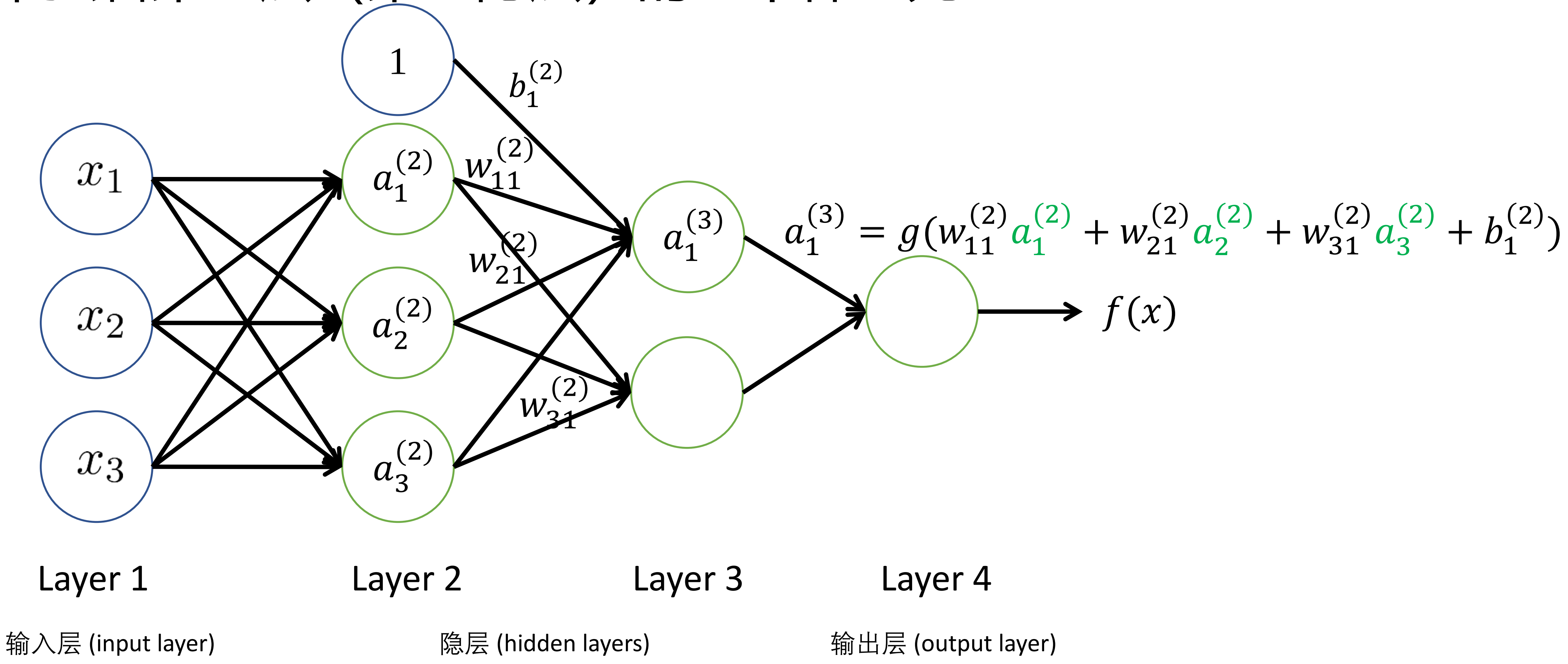
$$a_1^{(2)} = g(w_{11}^{(1)}x_1 + w_{21}^{(1)}x_2 + w_{31}^{(1)}x_3 + b_1^{(1)})$$

$b_i^{(k)}$: bias added to neuron i at layer $k+1$

$g(\cdot)$: a **nonlinear activation function (激活函数)**

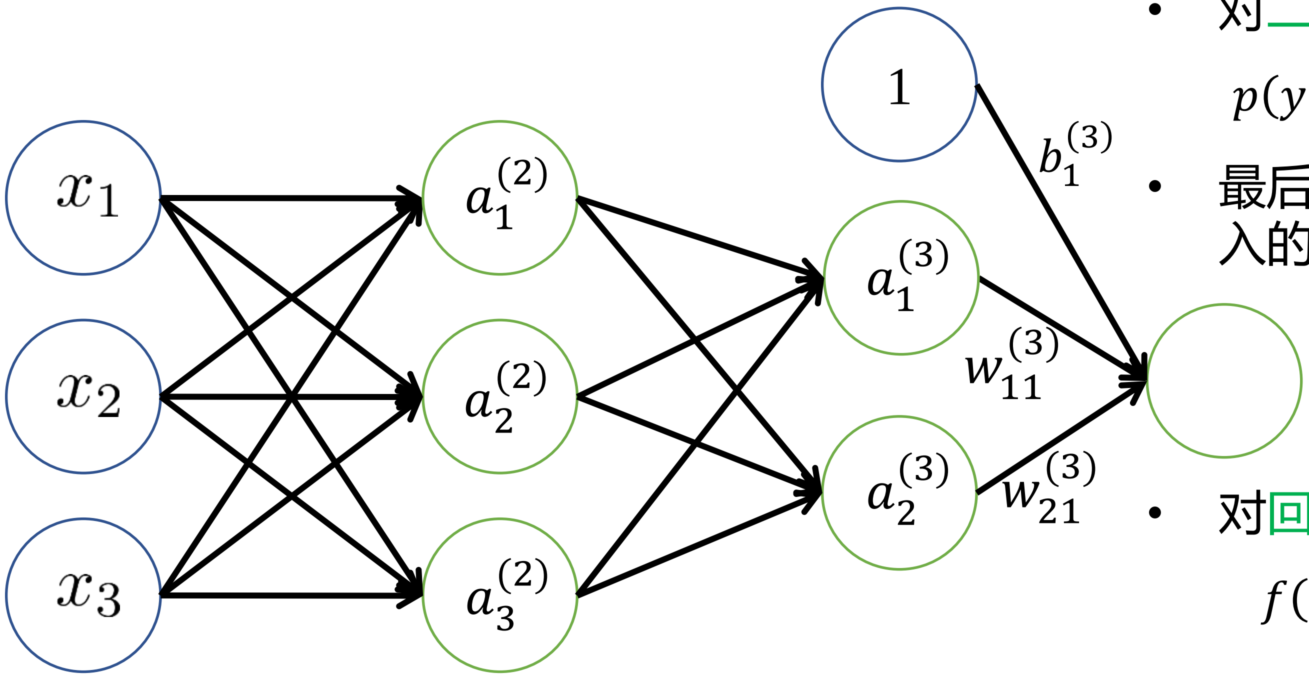
正向传播

- 单独看第三层（第二隐层）的一个神经元



正向传播

- 单独看输出层的一个神经元



$p(y=1|x)$ or $f(x)$

Layer 1   Layer 2   Layer 3   Layer 4

输入层 (input layer)   隐层 (hidden layers)   输出层 (output layer)

- 对二分类问题，用sigmoid函数输出正类概率

$$p(y=1|x)=\sigma(w_{11}^{(3)}a_1^{(3)}+w_{21}^{(3)}a_2^{(3)}+b_1^{(3)})$$

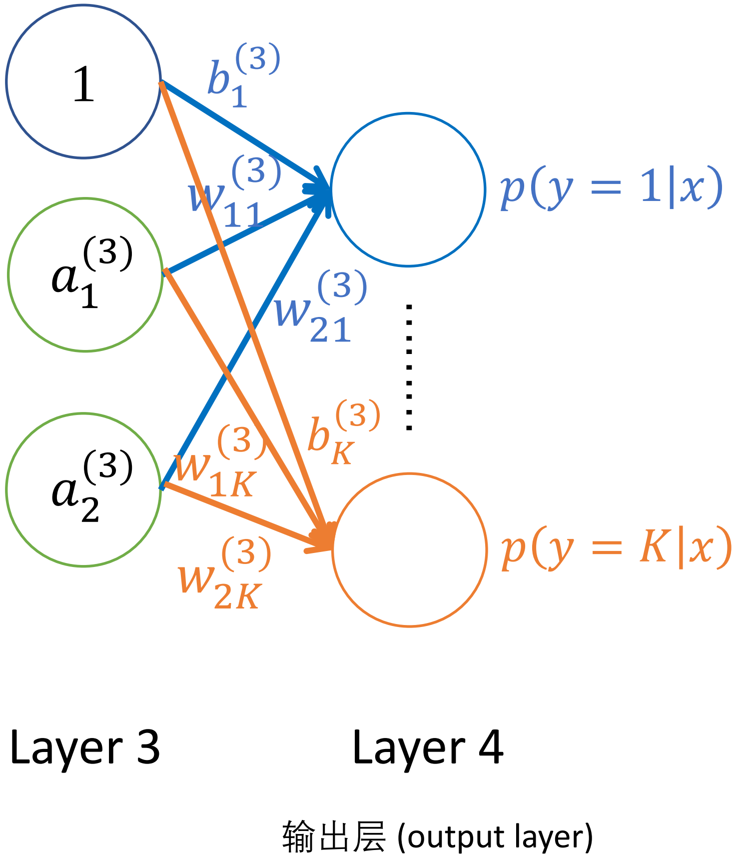
- 最后一层等价于一个以倒数第二层的激活作为输入的逻辑回归

- 对回归问题，直接输出

$$f(x)=w_{11}^{(3)}a_1^{(3)}+w_{21}^{(3)}a_2^{(3)}+b_1^{(3)}$$

正向传播

- 对多分类问题



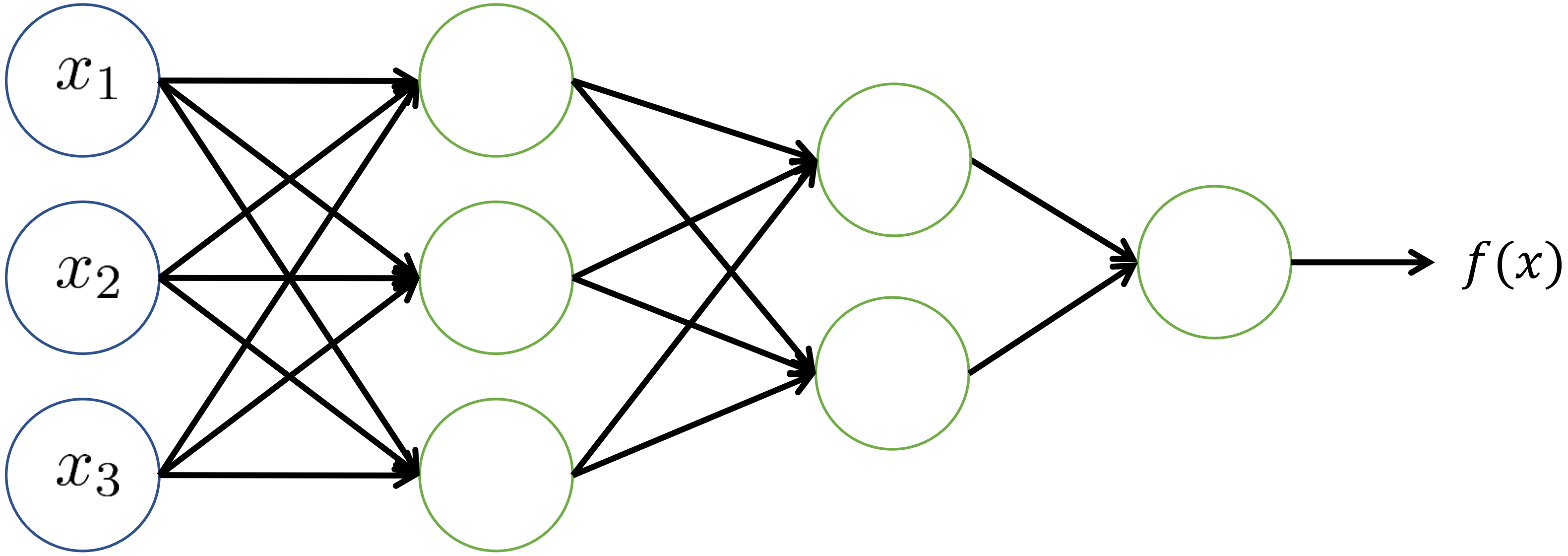
- 对 K 分类，输出层使用 K 个神经元
- 使用softmax函数，第 k 个输出层神经元输出

$$p(y=k|x)=\frac{\exp(w_{1k}^{(3)}a_1^{(3)}+w_{2k}^{(3)}a_2^{(3)}+b_k^{(3)})}{\sum_{j\in[K]}\exp(w_{1j}^{(3)}a_1^{(3)}+w_{2j}^{(3)}a_2^{(3)}+b_j^{(3)})}$$

- 即，最后一层等价于一个softmax回归

神经网络训练

- 问题：如何训练所有参数 $\{w_{ij}^{(l)}, b_j^{(l)}\}$?
- 梯度下降！如何求梯度?



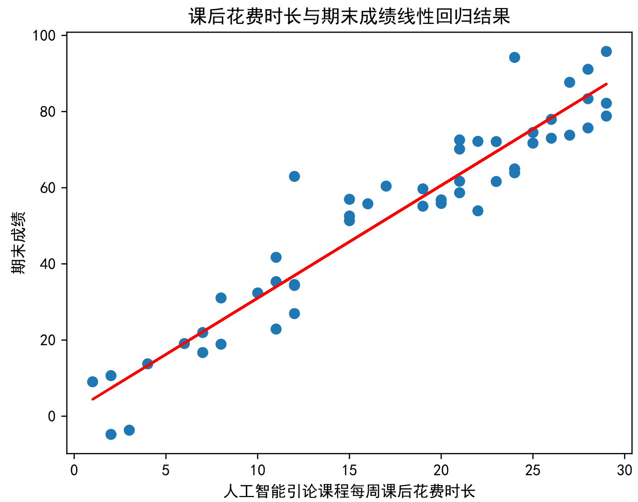
Layer 1　　Layer 2　　Layer 3　　Layer 4

输入层 (input layer)　　隐层 (hidden layers)　　输出层 (output layer)

神经网络训练——回顾线性回归模型

- 给定训练数据 $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$，学习率$\alpha$
- 初始化模型参数 w, b
- 训练模型：
  - a. 计算模型预测值：$\hat{y}_i = w^T x_i + b,\ \forall i \in [n]$
  - b. 计算平方损失函数：$J(w,b) = \frac{1}{n}\sum_{i\in[n]}(\hat{y}_i - y_i)^2$
  - c. 计算梯度
    - $\frac{\partial J(w,b)}{\partial w} = \frac{2}{n}\sum_{i\in[n]}(\hat{y}_i - y_i)\, x_i,\quad \frac{\partial J(w,b)}{\partial b} = \frac{2}{n}\sum_{i\in[n]}(\hat{y}_i - y_i)$
  - d. 梯度下降更新w, b
    - $w \leftarrow w - \alpha \cdot \frac{\partial J(w,b)}{\partial w},\ b \leftarrow b - \alpha \cdot \frac{\partial J(w,b)}{\partial b}$
  - e. 重复以上步骤，直到损失函数不再下降或达到预设迭代次数
- 给定新数据 x, 使用训练好的模型预测其标签 $\hat{y} = w^T x + b$

- 线性模型：$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$



反向传播

反向传播 (Backpropagation) 的作用

直观理解：

（1）**正向传播**：十个人在玩你画我猜的游戏，然后第一个人给第二个人描述，再将信息传递给第三个人，最后由第十个人说出画的内容。

（2）**反向传播**：第十个人得知自己说的和真实答案之间的误差后，发现他们在传递时的问题差在哪里，向前面一个人说下次描述的时候怎样可以更加准确的传递信息。就这样一直向前一个人告知。

（3）**权重更新**：十个人之间的的默契一直在磨合，然后描述的更加准确。

偏导数与链式法则

复合函数的偏导与链式法则：

复合函数偏导：

以二元函数 $z=f(u,v)$ 为例，z 是 u，v 的函数，但若 u 和 v 又都是 x 和 y 的函数，则 z 最终是 x 和 y 的函数，即

$$z(x,y)=f[u(x,y),v(x,y)].$$

由 $z=f(u,v)$ 得

$$dz=\frac{\partial f}{\partial u}du+\frac{\partial f}{\partial v}dv.$$

又由于 $du=\frac{\partial u}{\partial x}dx+\frac{\partial u}{\partial y}dy$，$dv=\frac{\partial v}{\partial x}dx+\frac{\partial v}{\partial y}dv$，代入上式得

$$dz=\frac{\partial f}{\partial u}\left(\frac{\partial u}{\partial x}dx+\frac{\partial u}{\partial y}dy\right)+\frac{\partial f}{\partial v}\left(\frac{\partial v}{\partial x}dx+\frac{\partial v}{\partial y}dy\right)$$

偏导数与链式法则

$$z(x,y)=f[u(x,y),v(x,y)]$$

$$\begin{aligned} dz &= \frac{\partial f}{\partial u}\left(\frac{\partial u}{\partial x}dx+\frac{\partial u}{\partial y}dy\right)+\frac{\partial f}{\partial v}\left(\frac{\partial v}{\partial x}dx+\frac{\partial v}{\partial y}dy\right) \\ &= \left(\frac{\partial f}{\partial u}\frac{\partial u}{\partial x}+\frac{\partial f}{\partial v}\frac{\partial v}{\partial x}\right)dx+\left(\frac{\partial f}{\partial u}\frac{\partial u}{\partial y}+\frac{\partial f}{\partial v}\frac{\partial v}{\partial y}\right)dy \end{aligned}$$

这就是 z 关于 x 和 y 的全微分关系，根据偏导数的定义

$$\frac{\partial z}{\partial x}=\frac{\partial f}{\partial u}\frac{\partial u}{\partial x}+\frac{\partial f}{\partial v}\frac{\partial v}{\partial x}, \qquad \frac{\partial z}{\partial y}=\frac{\partial f}{\partial u}\frac{\partial u}{\partial y}+\frac{\partial f}{\partial v}\frac{\partial v}{\partial y}.$$

这也叫偏导的**链式法则**。

偏导数与链式法则

例:

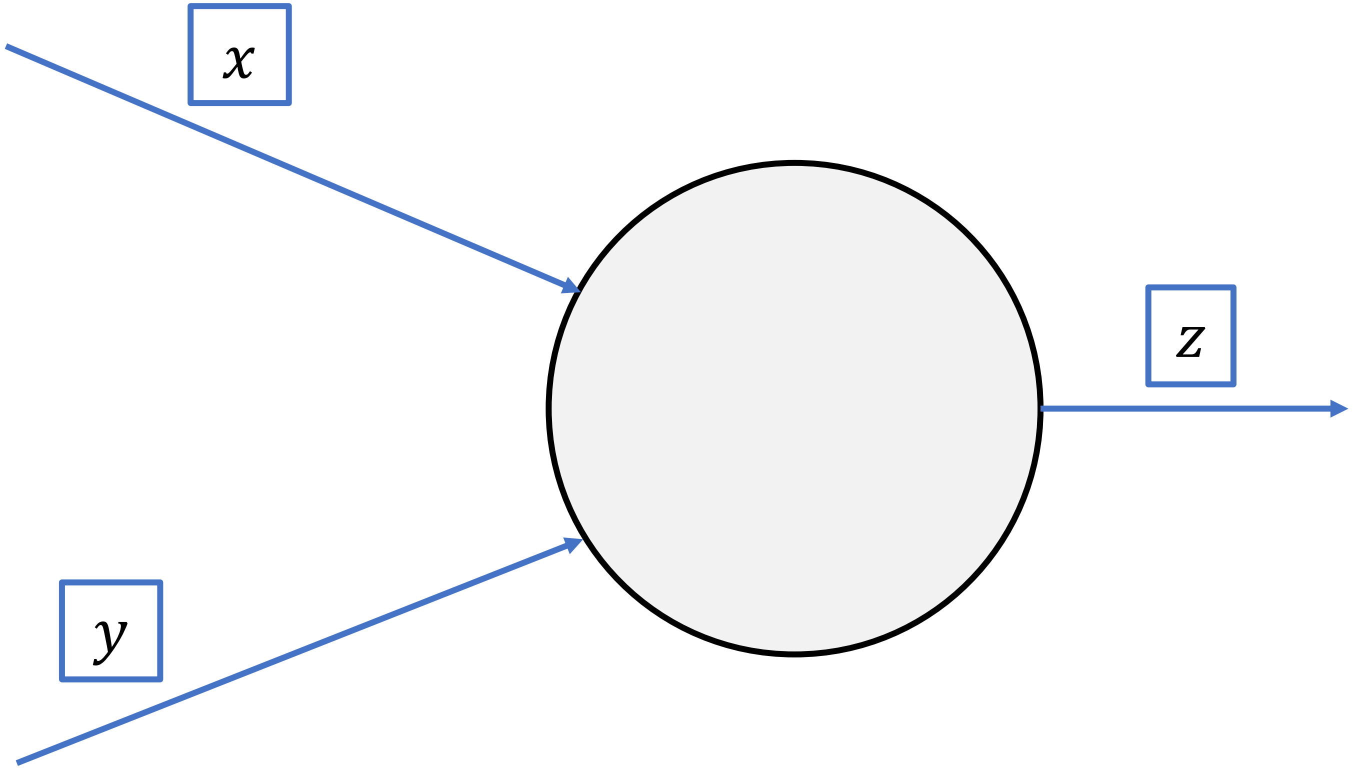
设 $z = f(u,v) = u^2 + v^2$，$u = xy$，$v = \frac{x}{y}$，求 $\frac{\partial z}{\partial x}$ 和 $\frac{\partial z}{\partial y}$.

解:

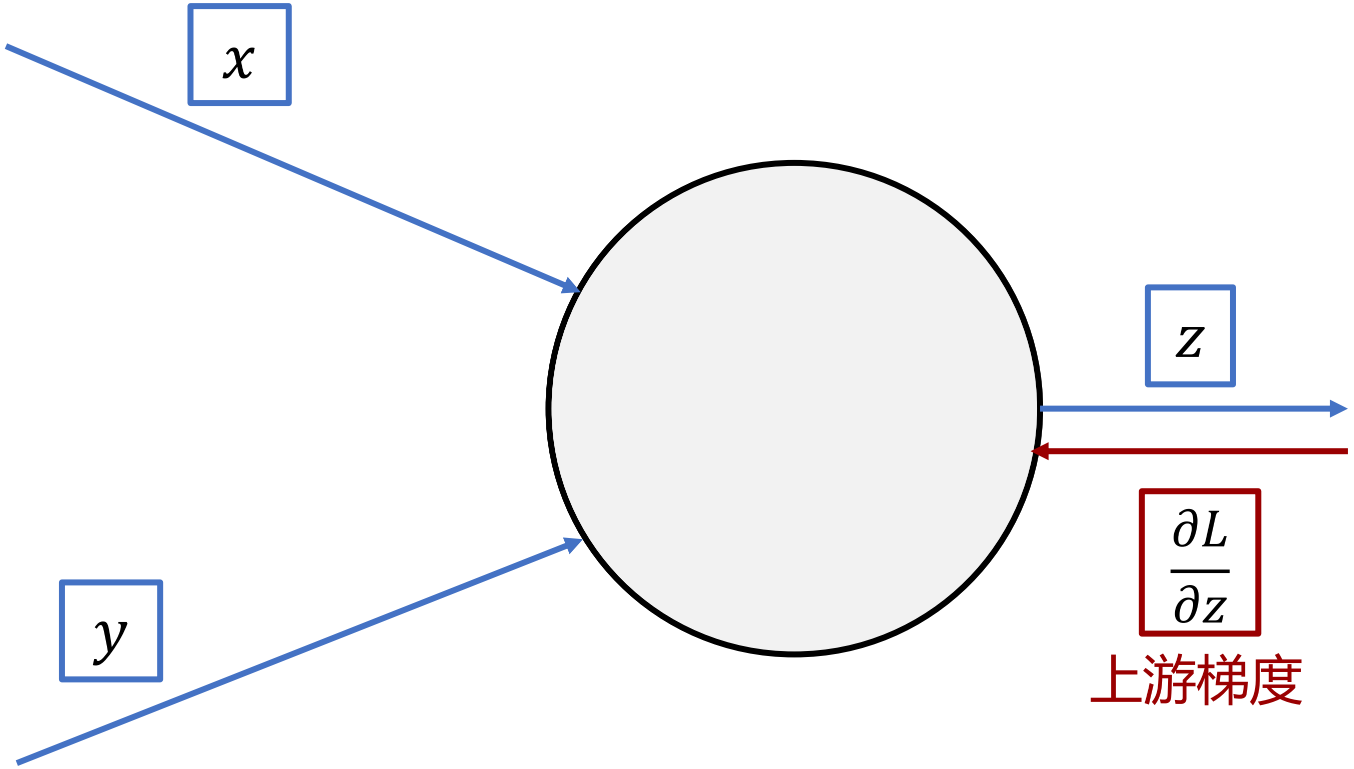
$$\frac{\partial z}{\partial x} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial f}{\partial v}\frac{\partial v}{\partial x} = 2u \cdot y + 2v \cdot \frac{1}{y} = 2xy^2 + \frac{2x}{y^2}.$$

$$\frac{\partial z}{\partial y} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial y} + \frac{\partial f}{\partial v}\frac{\partial v}{\partial y} = 2u \cdot x + 2v \cdot \left(-\frac{x}{y^2}\right) = 2x^2y - \frac{2x^2}{y^3}.$$

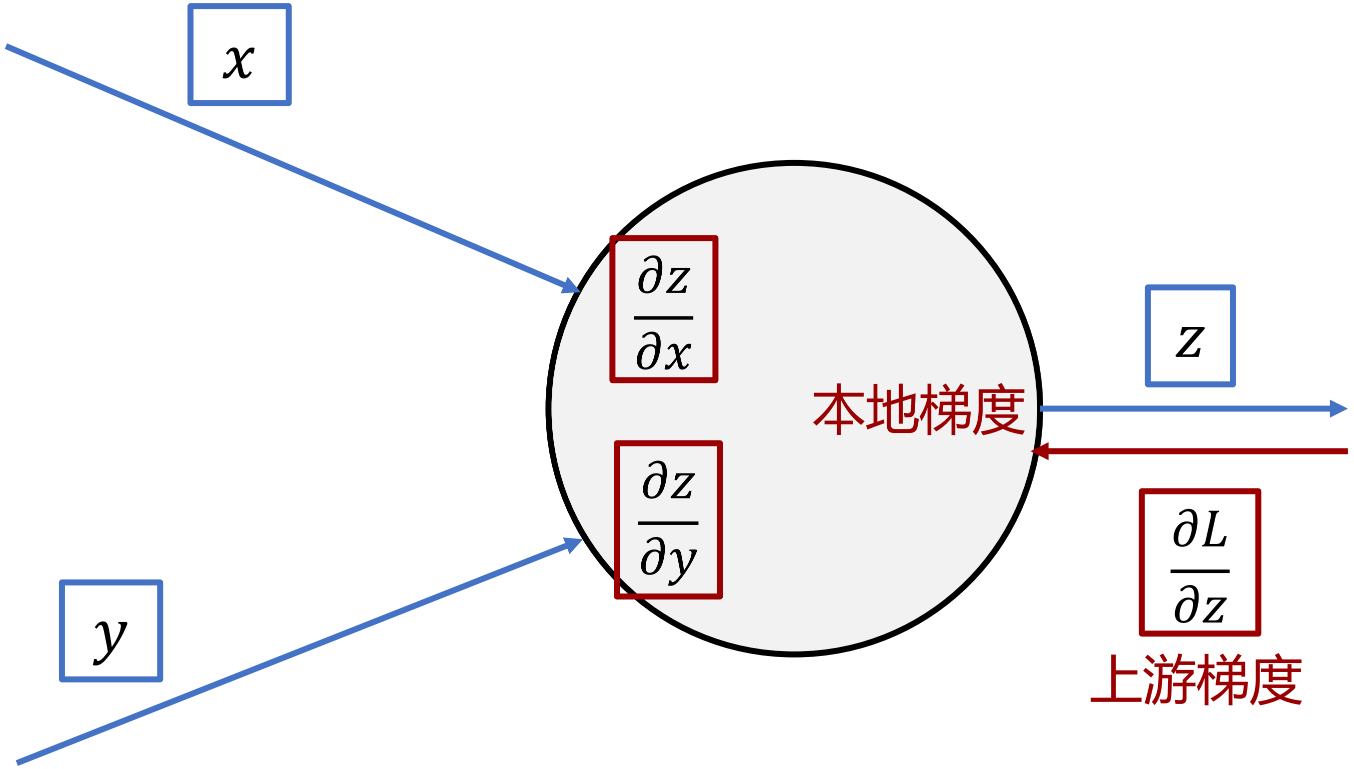
反向传播



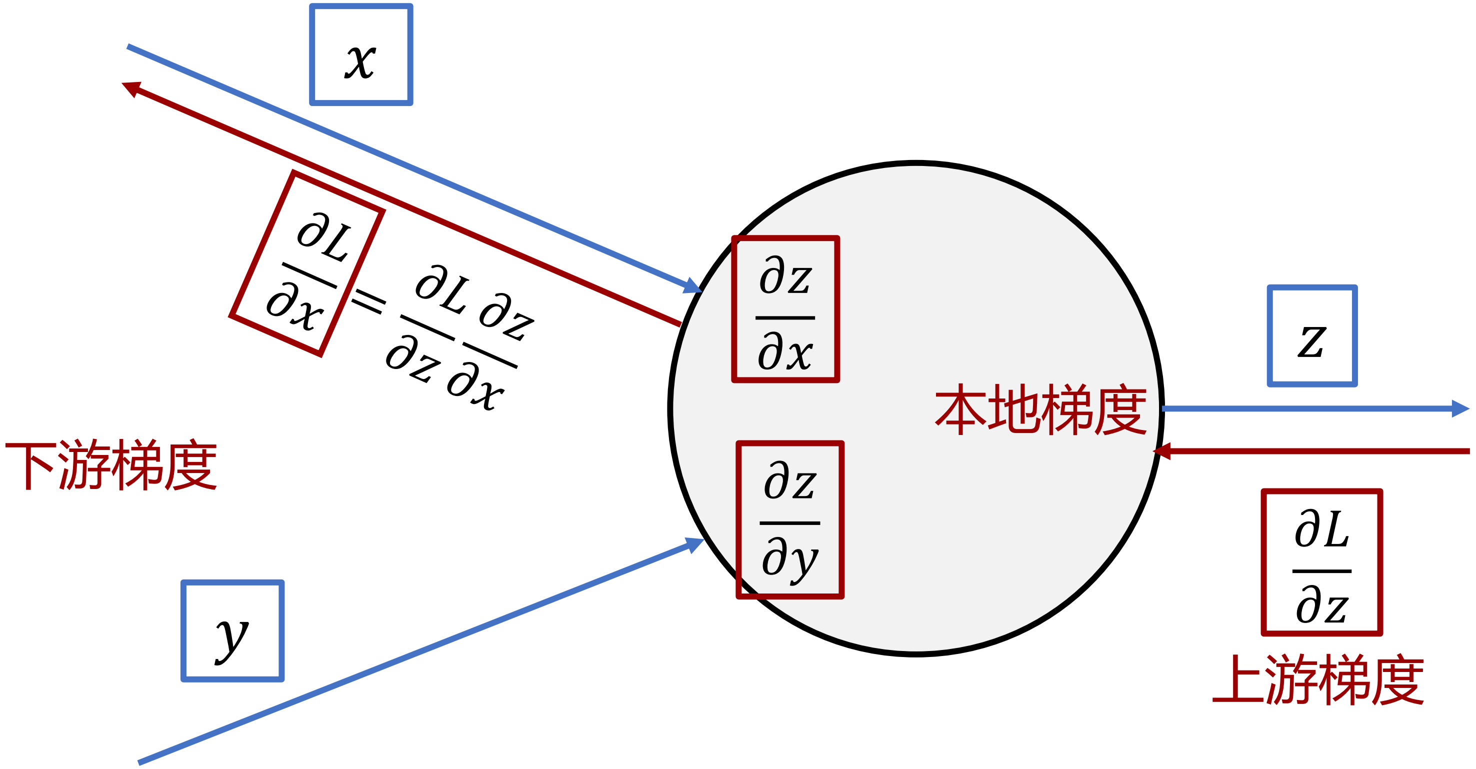
反向传播



反向传播

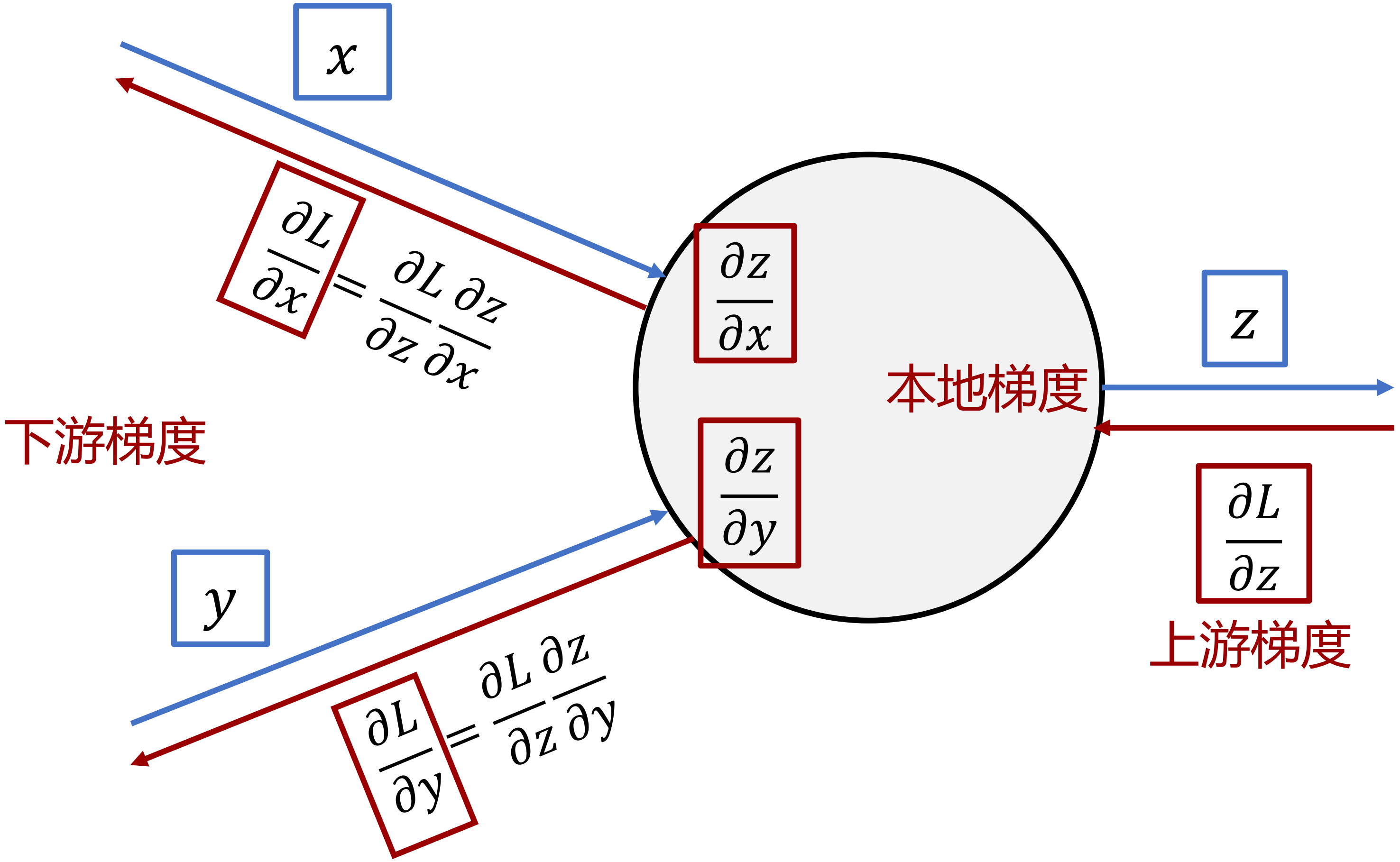

x
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
本地梯度
z
y
$\frac{\partial L}{\partial z}$
上游梯度

反向传播



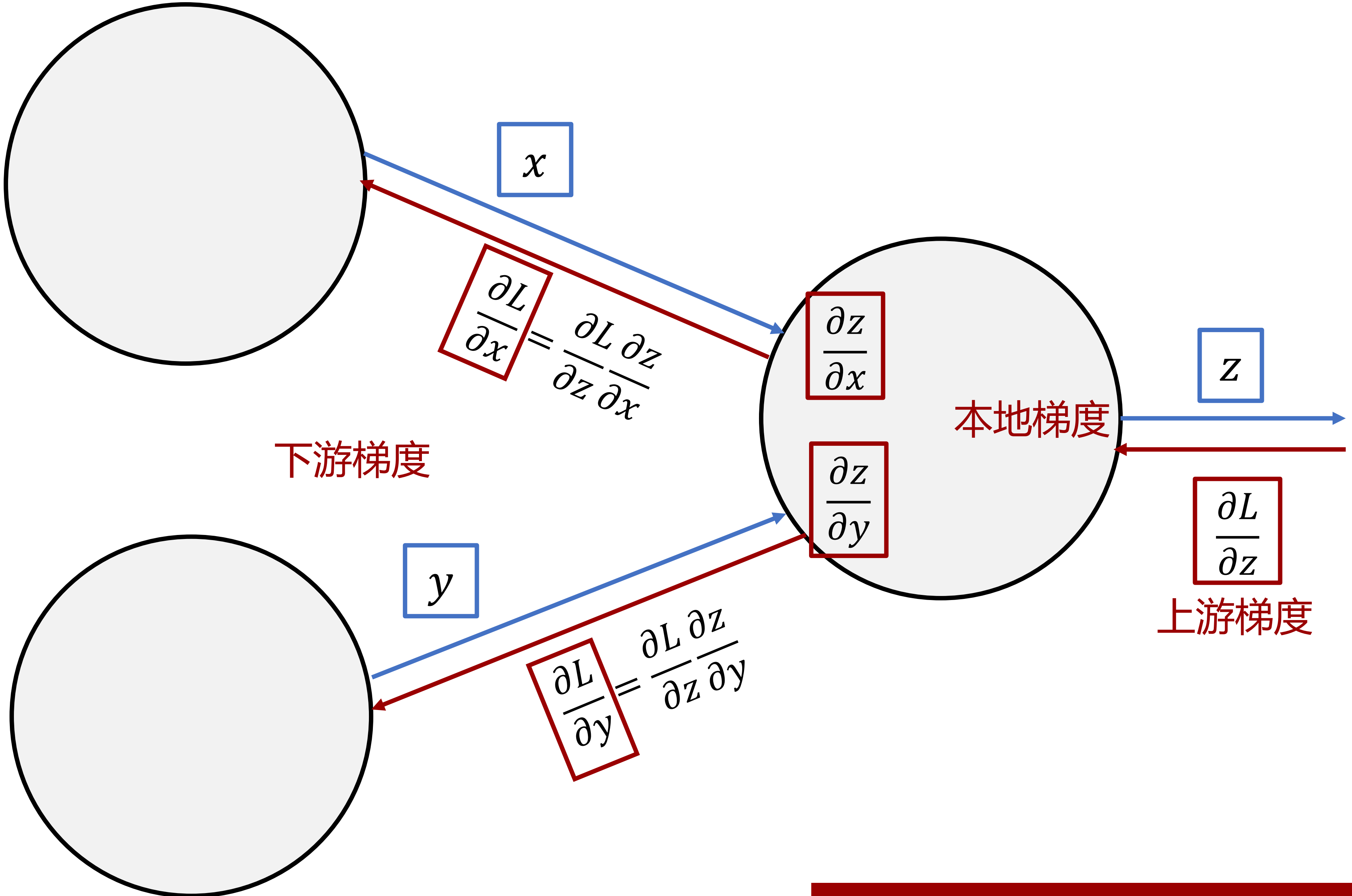
x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
下游梯度
y
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
本地梯度
z
$\frac{\partial L}{\partial z}$
上游梯度

反向传播



x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
下游梯度
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
本地梯度
z
$\frac{\partial L}{\partial z}$
上游梯度

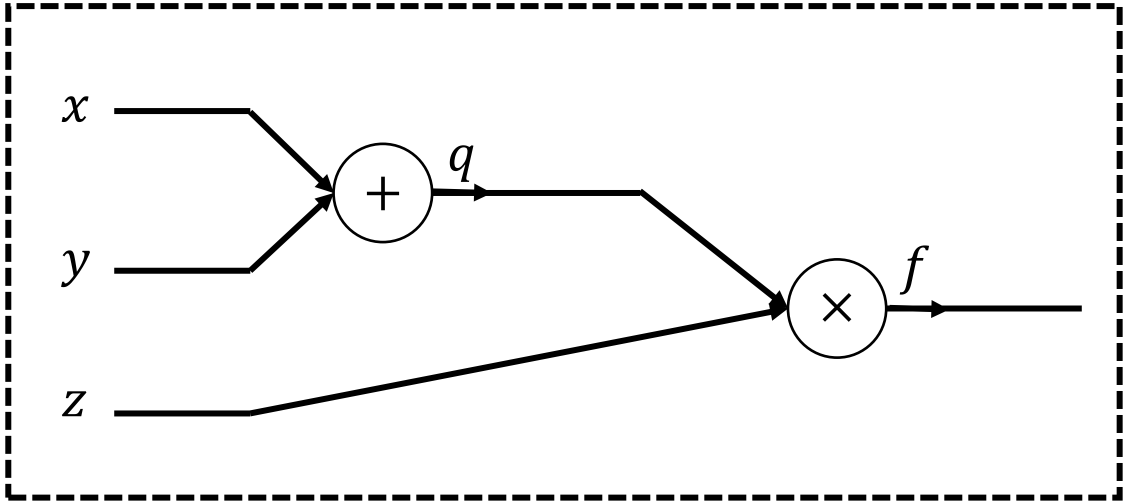
反向传播


x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial x}$
本地梯度
z
$\frac{\partial z}{\partial y}$
下游梯度
$\frac{\partial L}{\partial z}$
上游梯度
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$

反向传播

例1：

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

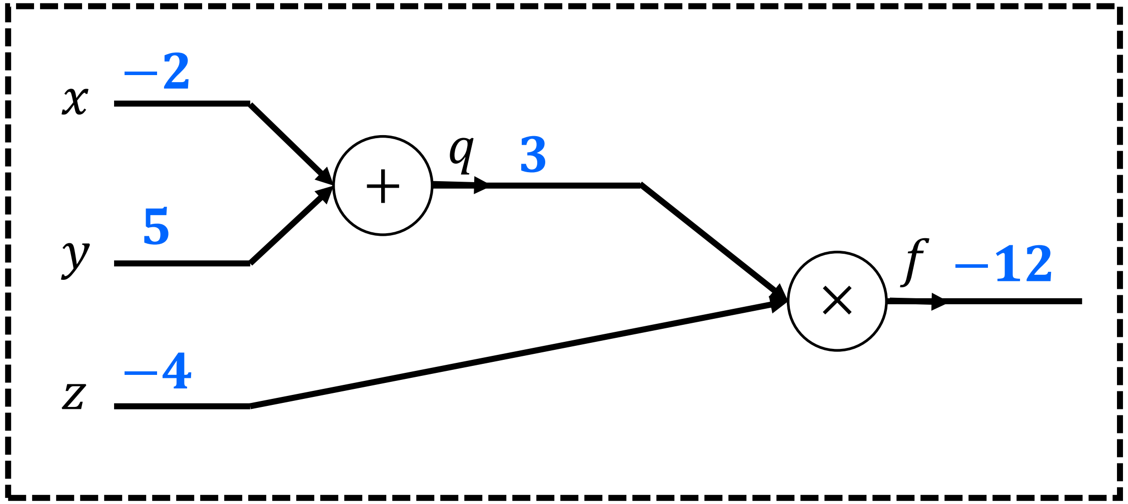


反向传播

例1：

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

以 $\boldsymbol{x = -2}$，$\boldsymbol{y = 5}$，$\boldsymbol{z = -4}$ 时为例，



反向传播

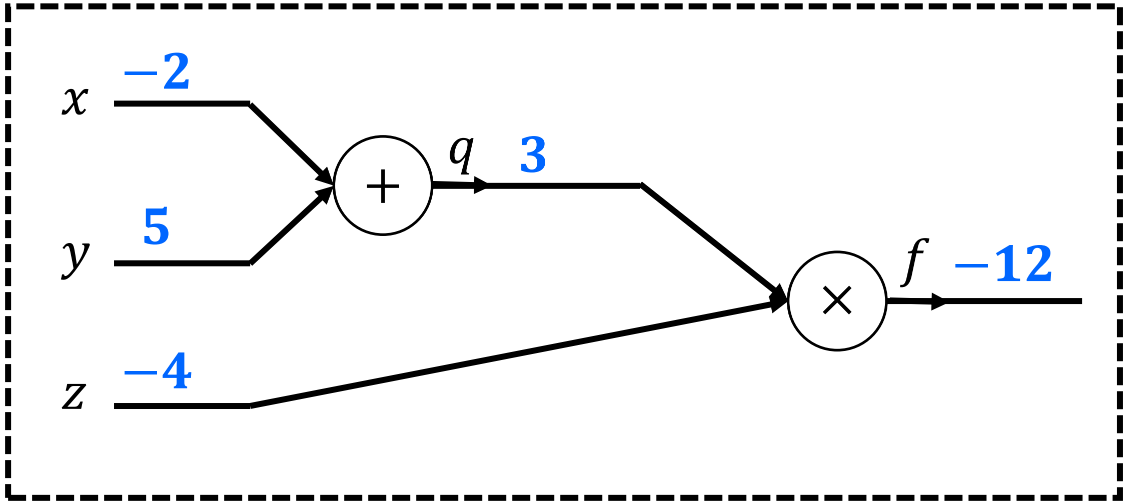
例1:

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

以 $\boldsymbol{x = -2}$，$\boldsymbol{y = 5}$，$\boldsymbol{z = -4}$ 时为例，

$q = x + y$，$\dfrac{\partial q}{\partial x} = 1$，$\dfrac{\partial q}{\partial y} = 1$；$f = qz$，$\dfrac{\partial f}{\partial q} = z = -4$，$\dfrac{\partial f}{\partial z} = q = 3$.

求 $\dfrac{\partial f}{\partial x}$，$\dfrac{\partial f}{\partial y}$，$\dfrac{\partial f}{\partial z}$.



反向传播

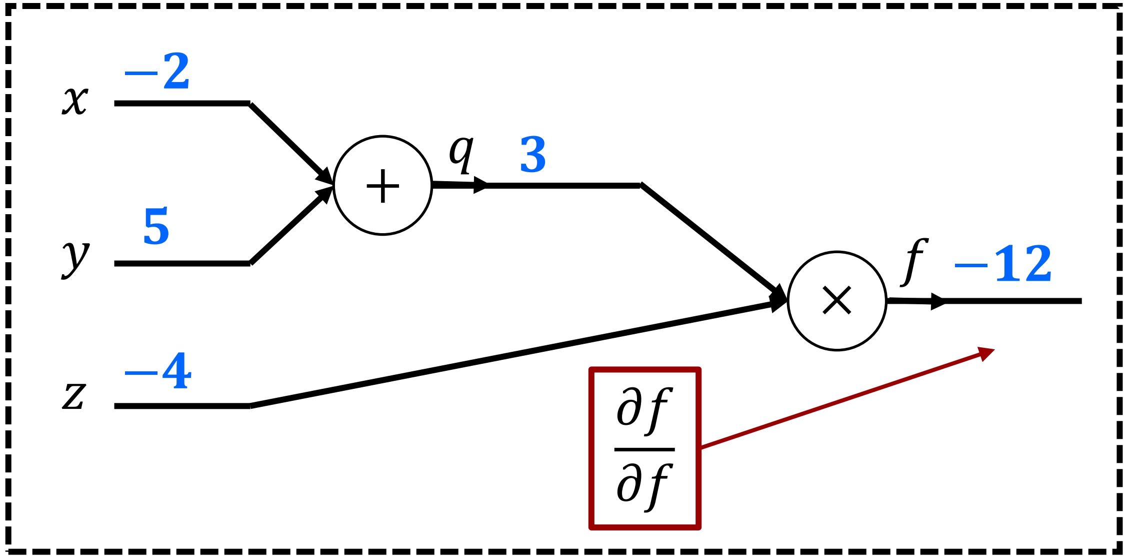
例1:

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

以 $\boldsymbol{x=-2}$, $\boldsymbol{y=5}$, $\boldsymbol{z=-4}$ 时为例,

$q = x + y$, $\frac{\partial q}{\partial x} = 1$, $\frac{\partial q}{\partial y} = 1$; $f = qz$, $\frac{\partial f}{\partial q} = z = -4$, $\frac{\partial f}{\partial z} = q = 3.$

求 $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$, $\frac{\partial f}{\partial z}$.



反向传播

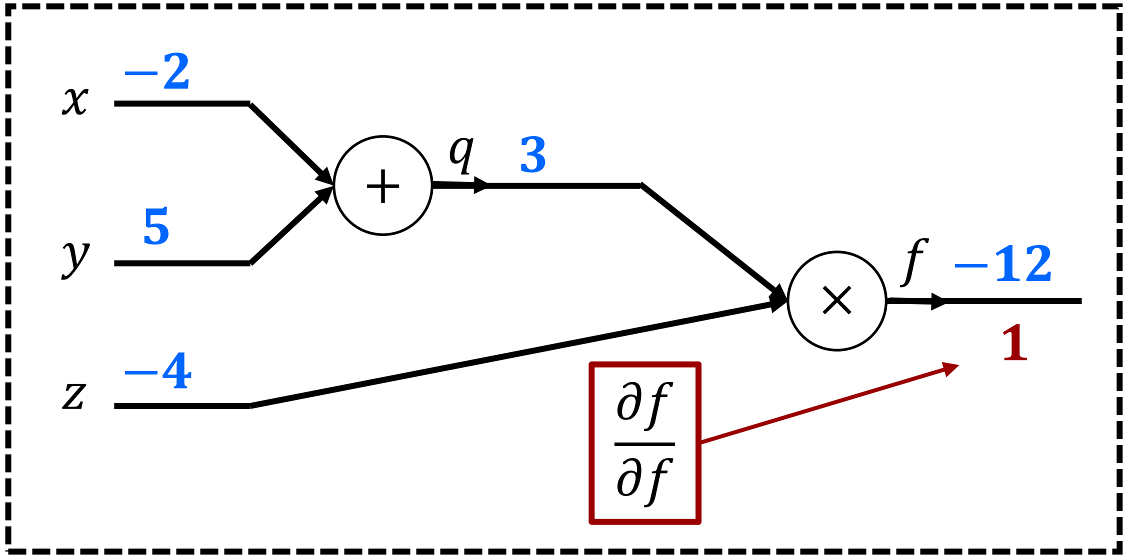
例1：

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

以 $x=-2$，$y=5$，$z=-4$ 时为例，

$q=x+y$，$\frac{\partial q}{\partial x}=1$，$\frac{\partial q}{\partial y}=1$；$f=qz$，$\frac{\partial f}{\partial q}=z=-4$，$\frac{\partial f}{\partial z}=q=3$.

求 $\frac{\partial f}{\partial x}$，$\frac{\partial f}{\partial y}$，$\frac{\partial f}{\partial z}$.



反向传播

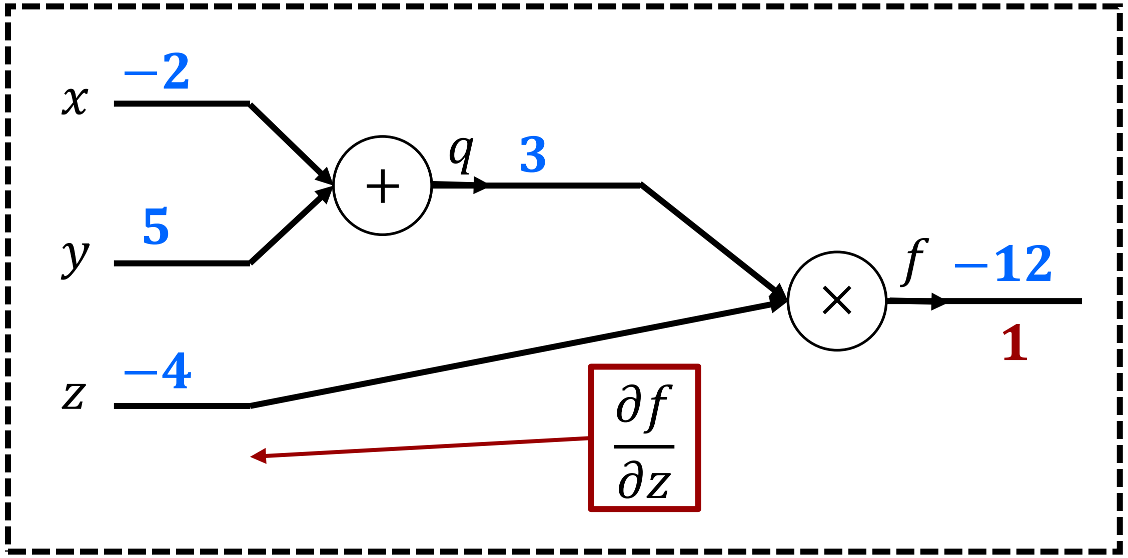
例1：

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

以 $\boldsymbol{x=-2}$，$\boldsymbol{y=5}$，$\boldsymbol{z=-4}$ 时为例，

$q = x + y$，$\dfrac{\partial q}{\partial x} = 1$，$\dfrac{\partial q}{\partial y} = 1$；$f = qz$，$\dfrac{\partial f}{\partial q} = z = -4$，$\dfrac{\partial f}{\partial z} = q = 3$.

求 $\dfrac{\partial f}{\partial x}$，$\dfrac{\partial f}{\partial y}$，$\dfrac{\partial f}{\partial z}$.



反向传播

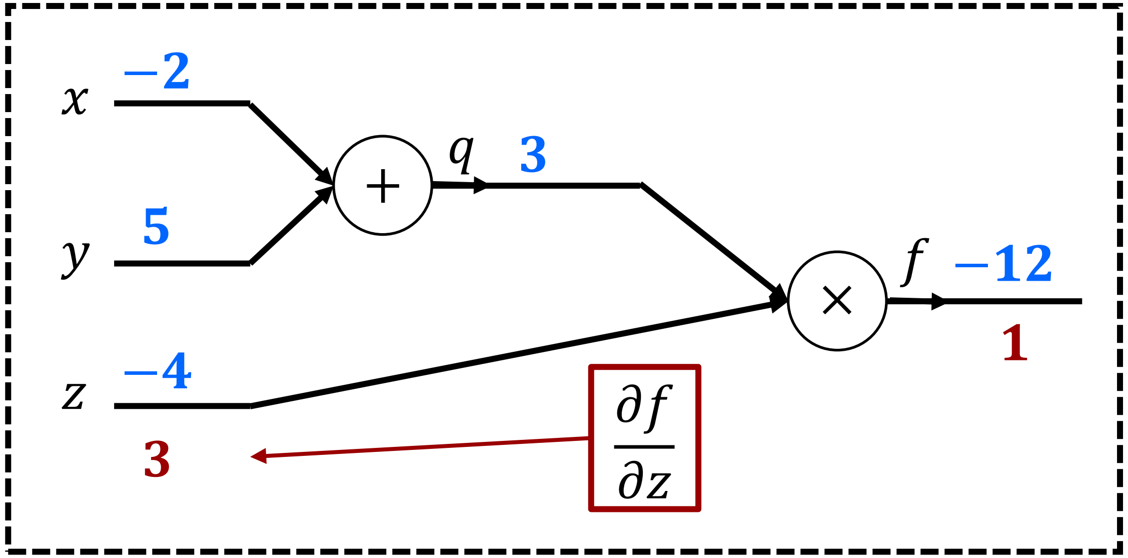
<u>例1</u>:

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

以 $\boldsymbol{x=-2}$，$\boldsymbol{y=5}$，$\boldsymbol{z=-4}$ 时为例，

$q=x+y$，$\frac{\partial q}{\partial x}=1$，$\frac{\partial q}{\partial y}=1$；$f=qz$，$\frac{\partial f}{\partial q}=z=-4$，$\frac{\partial f}{\partial z}=q=3$.

求 $\frac{\partial f}{\partial x}$，$\frac{\partial f}{\partial y}$，$\frac{\partial f}{\partial z}$.



反向传播

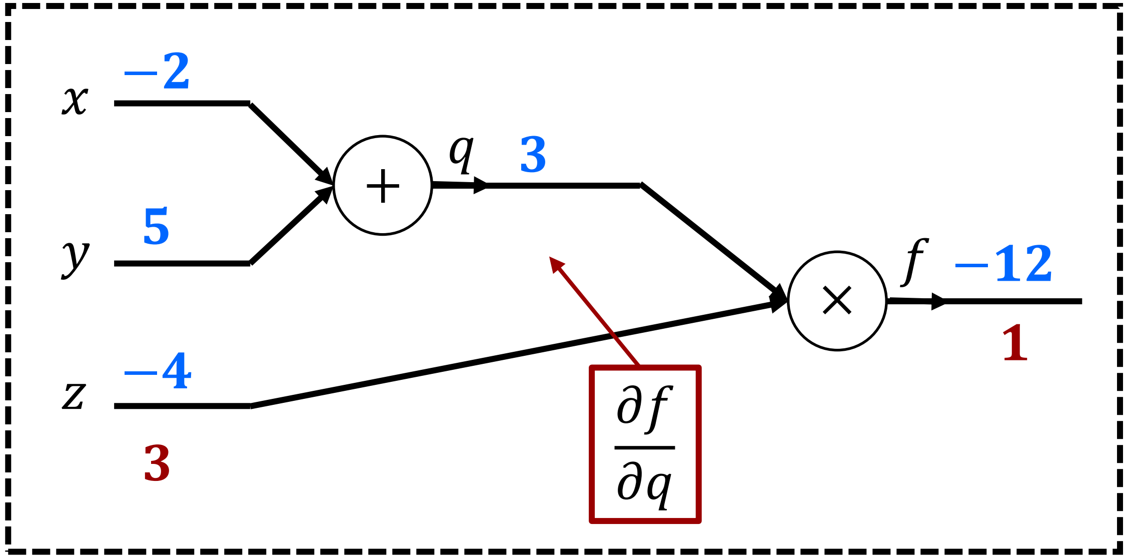
例1：

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

以 $\boldsymbol{x = -2}$，$\boldsymbol{y = 5}$，$\boldsymbol{z = -4}$ 时为例，

$q = x + y$，$\frac{\partial q}{\partial x} = 1$，$\frac{\partial q}{\partial y} = 1$；$f = qz$，$\frac{\partial f}{\partial q} = z = -4$，$\frac{\partial f}{\partial z} = q = 3$.

求 $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$.



反向传播

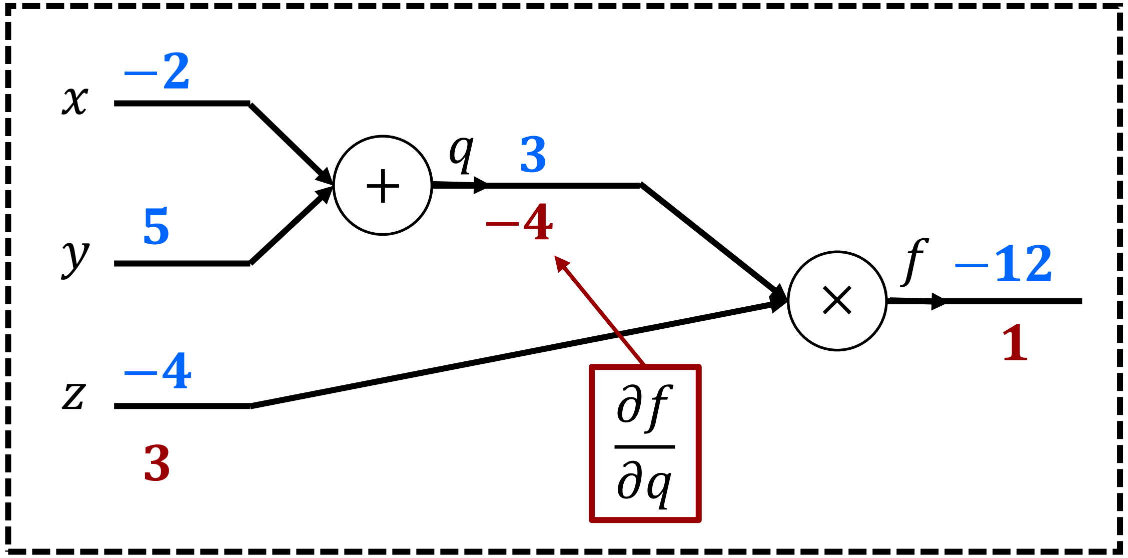
<u>例1</u>：

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

以 $x=-2$，$y=5$，$z=-4$ 时为例，

$q = x + y$，$\frac{\partial q}{\partial x} = 1$，$\frac{\partial q}{\partial y} = 1$；$f = qz$，$\frac{\partial f}{\partial q} = z = -4$，$\frac{\partial f}{\partial z} = q = 3.$

求 $\frac{\partial f}{\partial x}$，$\frac{\partial f}{\partial y}$，$\frac{\partial f}{\partial z}$.



反向传播

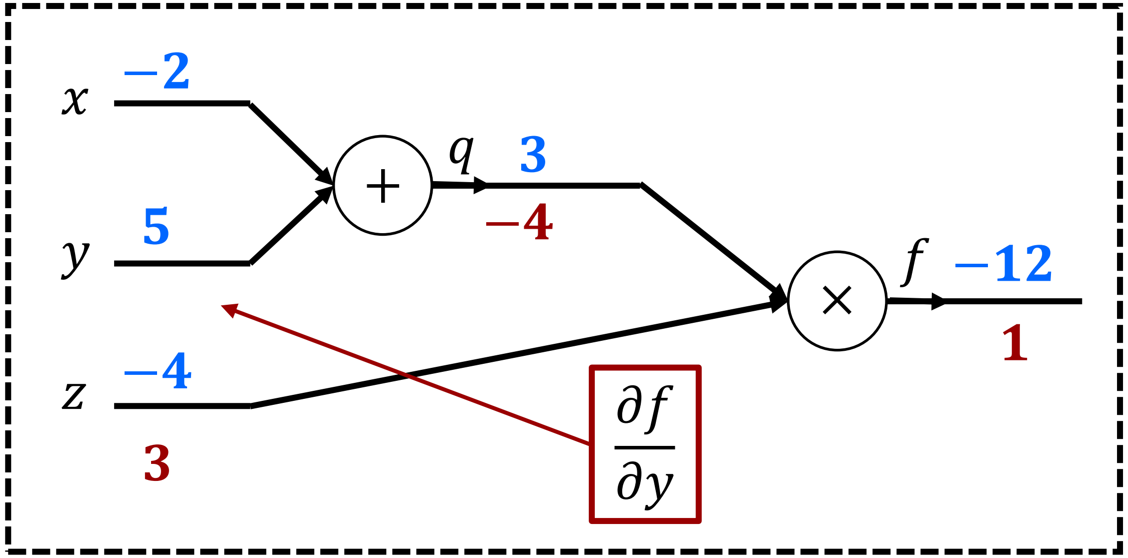
例1：

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

以 $\boldsymbol{x = -2}$，$\boldsymbol{y = 5}$，$\boldsymbol{z = -4}$ 时为例，

$q = x + y$，$\frac{\partial q}{\partial x} = 1$，$\frac{\partial q}{\partial y} = 1$；$f = qz$，$\frac{\partial f}{\partial q} = z = -4$，$\frac{\partial f}{\partial z} = q = 3$.

求 $\frac{\partial f}{\partial x}$，$\frac{\partial f}{\partial y}$，$\frac{\partial f}{\partial z}$.



链式法则：

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

上游梯度　　本地梯度

反向传播

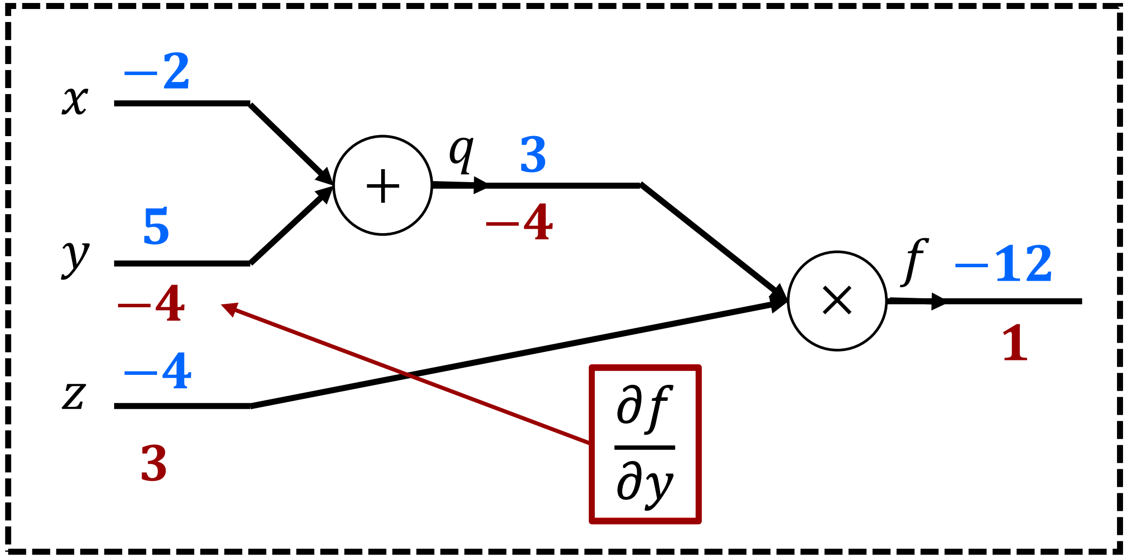
例1:

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

以 $\boldsymbol{x = -2}$，$\boldsymbol{y = 5}$，$\boldsymbol{z = -4}$ 时为例，

$q = x + y$，$\frac{\partial q}{\partial x} = 1$，$\frac{\partial q}{\partial y} = 1$；$f = qz$，$\frac{\partial f}{\partial q} = z = -4$，$\frac{\partial f}{\partial z} = q = 3$.

求 $\frac{\partial f}{\partial x}$，$\frac{\partial f}{\partial y}$，$\frac{\partial f}{\partial z}$.



链式法则:

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

上游梯度　　本地梯度

反向传播

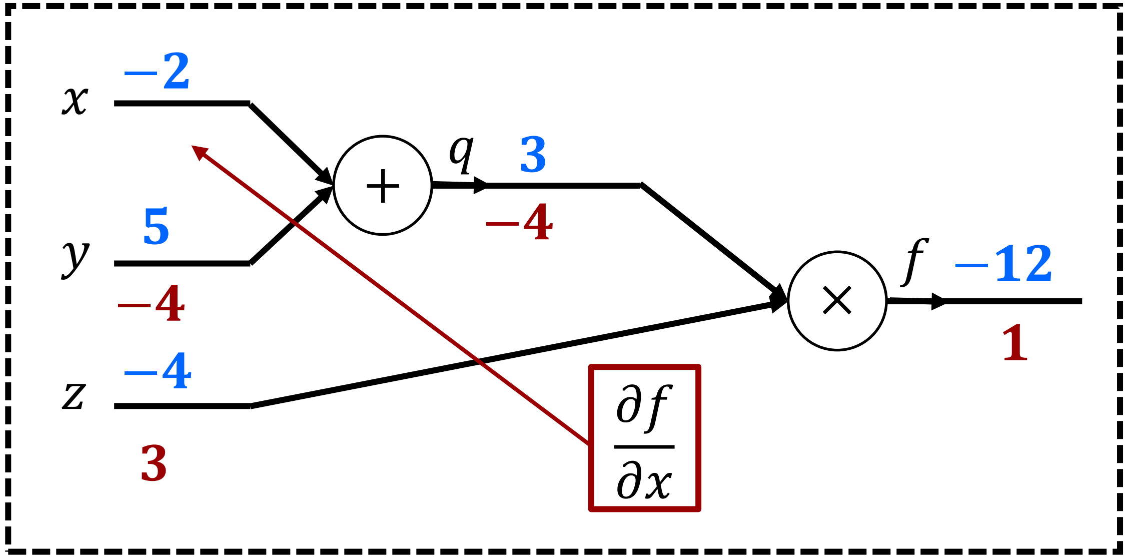
例1：

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

以 $\boldsymbol{x=-2}$，$\boldsymbol{y=5}$，$\boldsymbol{z=-4}$ 时为例，

$q=x+y$，$\frac{\partial q}{\partial x}=1$，$\frac{\partial q}{\partial y}=1$；$f=qz$，$\frac{\partial f}{\partial q}=z=-4$，$\frac{\partial f}{\partial z}=q=3$.

求 $\frac{\partial f}{\partial x}$，$\frac{\partial f}{\partial y}$，$\frac{\partial f}{\partial z}$.



链式法则：

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

上游梯度　　本地梯度

反向传播

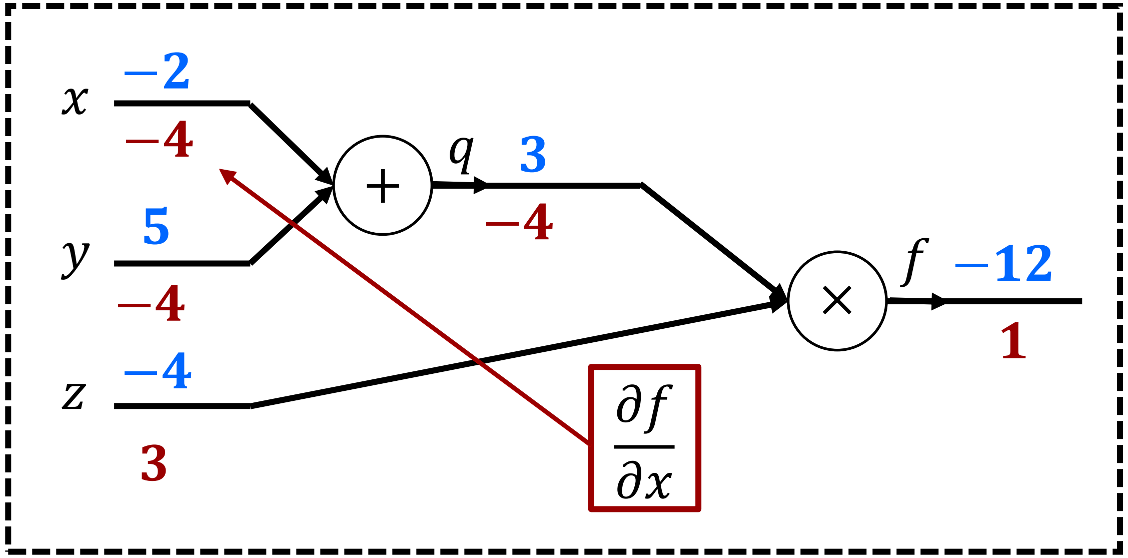
例1：

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

以 $\boldsymbol{x=-2}$，$\boldsymbol{y=5}$，$\boldsymbol{z=-4}$ 时为例，

$q = x + y$，$\dfrac{\partial q}{\partial x} = 1$，$\dfrac{\partial q}{\partial y} = 1$；$f = qz$，$\dfrac{\partial f}{\partial q} = z = -4$，$\dfrac{\partial f}{\partial z} = q = 3.$

求 $\dfrac{\partial f}{\partial x}$，$\dfrac{\partial f}{\partial y}$，$\dfrac{\partial f}{\partial z}$.



链式法则：

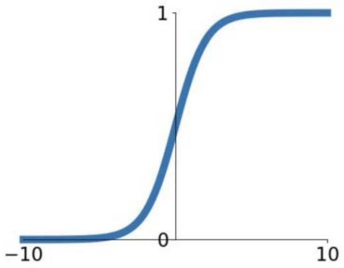
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

上游梯度　　本地梯度

激活函数选择

Sigmoid

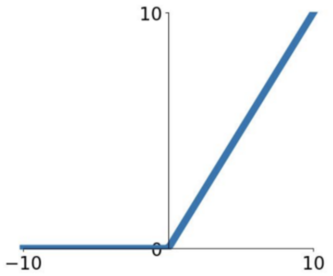
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



- 导数为 $(1-\sigma(x))\cdot\sigma(x)$，可能发生梯度消失现象
- 如：x的当前值在10,最优值在1，从10到1优化时梯度太小以至于"步子"特别小，很难达到优化目的地。特别是对于离输入层近的参数，更新效率极低

ReLU

$$\max(0, x)$$

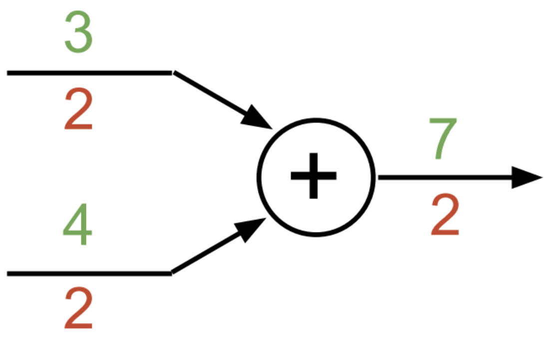


- 导数为 1 (x>0) 或 0 (x<=0)，不容易发生梯度消失
- 网络具有稀疏性：提高计算效率、在一定程度上避免过拟合
- 函数非常简单，计算速度更快

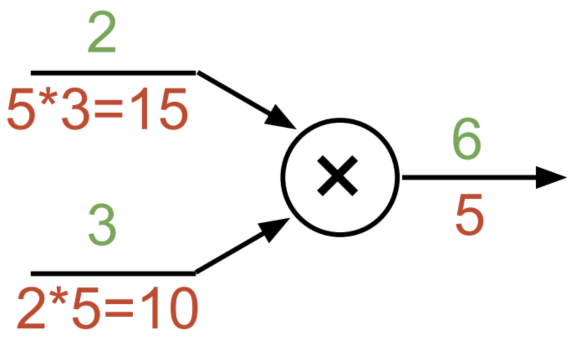
slide from Fei-Fei Li

常见操作的反向传播

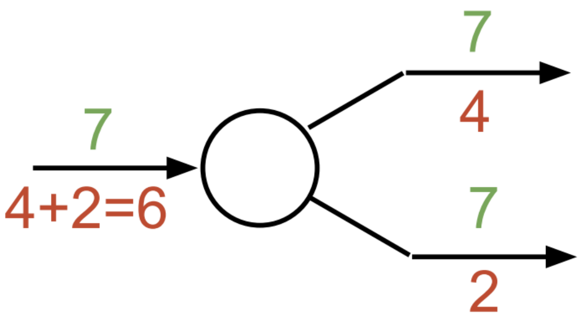
add gate: gradient distributor



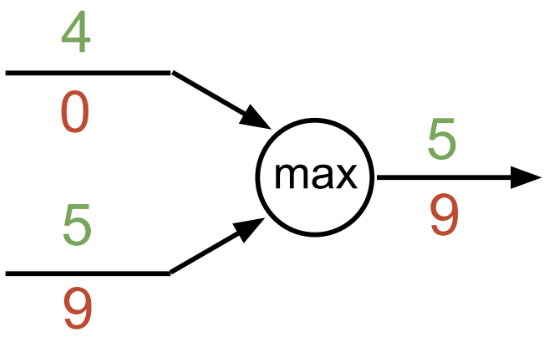
mul gate: “swap multiplier”



copy gate: gradient adder

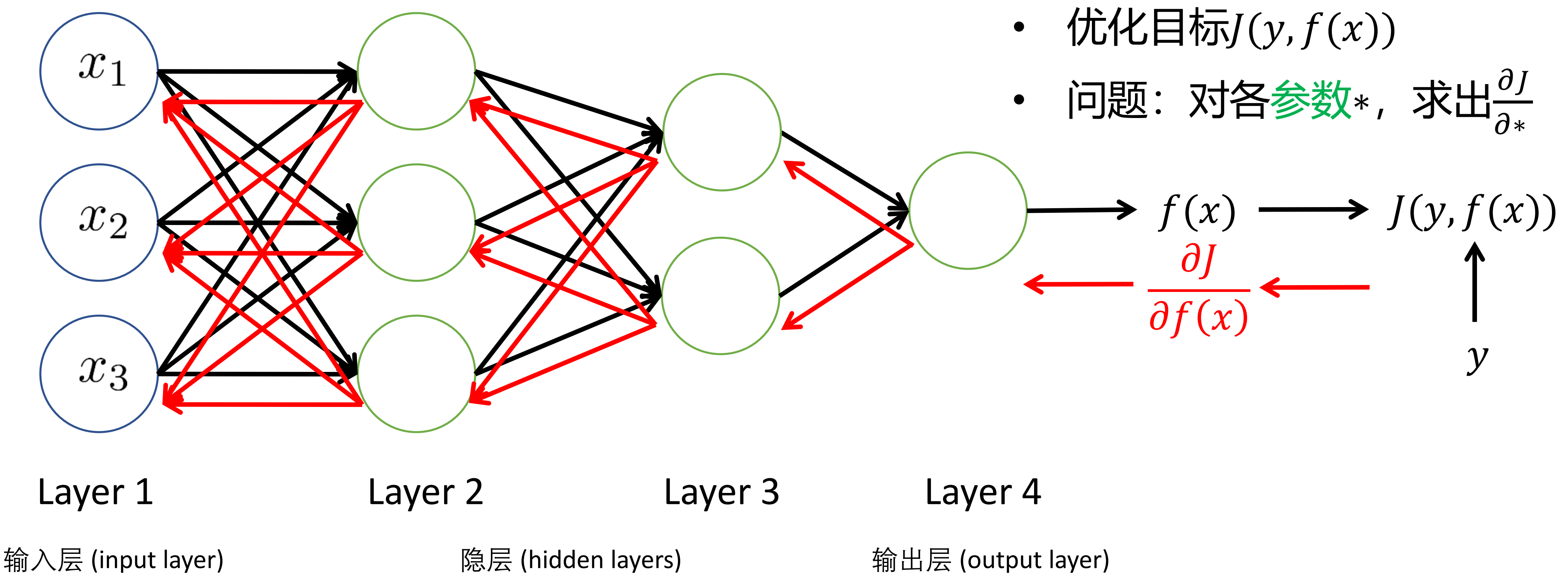


max gate: gradient router



神经网络的反向传播

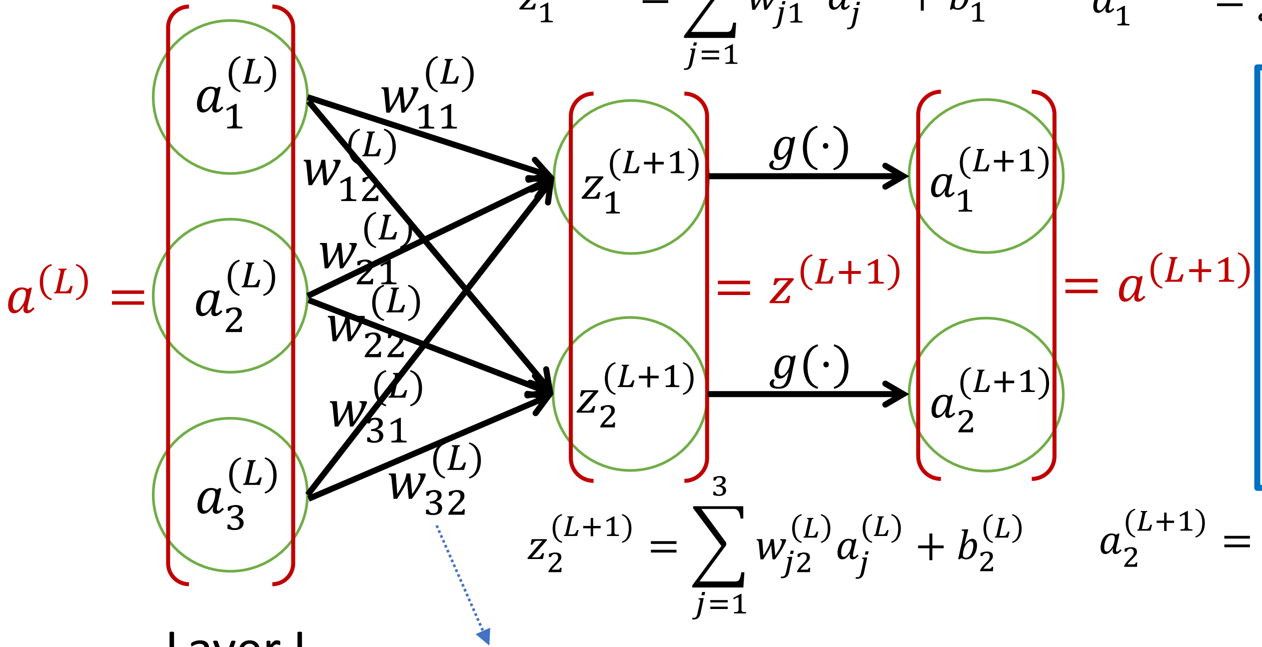
- 前向传播（Forward Propogation）从低层向高层计算隐藏层输出，直至输出层产生预测
- 反向传播（Backpropagation）在最高层计算损失函数，从高层向低层计算参数梯度



矩阵形式

• 前向传播

$$z_1^{(L+1)} = \sum_{j=1}^{3} w_{j1}^{(L)} a_j^{(L)} + b_1^{(L)} \qquad a_1^{(L+1)} = g\left(z_1^{(L+1)}\right)$$



$$z_2^{(L+1)} = \sum_{j=1}^{3} w_{j2}^{(L)} a_j^{(L)} + b_2^{(L)} \qquad a_2^{(L+1)} = g\left(z_2^{(L+1)}\right)$$

Layer L

$$W^{(L)} = \begin{bmatrix} w_{11}^{(L)} & w_{21}^{(L)} & w_{31}^{(L)} \\ w_{12}^{(L)} & w_{22}^{(L)} & w_{32}^{(L)} \end{bmatrix}$$

通用标量形式（第L层输入维度为d_L）：

$$z_i^{(L+1)} = \sum_{j=1}^{d_L} w_{ji}^{(L)} a_j^{(L)} + b_i^{(L)}, \qquad a_i^{(L+1)} = g\left(z_i^{(L+1)}\right)$$

通用矩阵形式:

$$z^{(L+1)} = W^{(L)} a^{(L)} + b^{(L)}, \qquad a^{(L+1)} = g\left(z^{(L+1)}\right)$$

- $a^{(L)} \in \mathbb{R}^{d_L}$, $z^{(L+1)} \in \mathbb{R}^{d_{L+1}}$, $a^{(L+1)} \in \mathbb{R}^{d_{L+1}}$
- $W^{(L)} \in \mathbb{R}^{d_{L+1} \times d_L}$, $b^{(L)} \in \mathbb{R}^{d_{L+1}}$
- $W_{ij}^{(L)} = w_{ji}^{(L)}$
- $g(\cdot)$：逐元素(element-wise)激活函数

向量对标量求导

定义：设向量 $\boldsymbol{y}=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$，其对标量 x 的导数为：

$$\frac{\partial \boldsymbol{y}}{\partial x}=\begin{bmatrix} \frac{\partial y_1}{\partial x} \\ \frac{\partial y_2}{\partial x} \\ \vdots \\ \frac{\partial y_m}{\partial x} \end{bmatrix}.$$

标量对向量求导

定义：设$y=f(\boldsymbol{x})=f(x_1,x_2,\cdots,x_n)$，$\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$，则

$$\frac{dy}{d\boldsymbol{x}}=\frac{df(\boldsymbol{x})}{d\boldsymbol{x}}=\left[\frac{\partial f}{\partial x_1},\frac{\partial f}{\partial x_2},\cdots,\frac{\partial f}{\partial x_n}\right]^{\mathrm{T}}.$$

$\frac{df(\boldsymbol{x})}{d\boldsymbol{x}}$ 也被称为函数 $y=f(\boldsymbol{x})$ 在点 $\boldsymbol{x}$ 处的梯度，记为

$$grad[f(\boldsymbol{x})] \text{ 或 } \nabla f(\boldsymbol{x}).$$

标量对矩阵求导

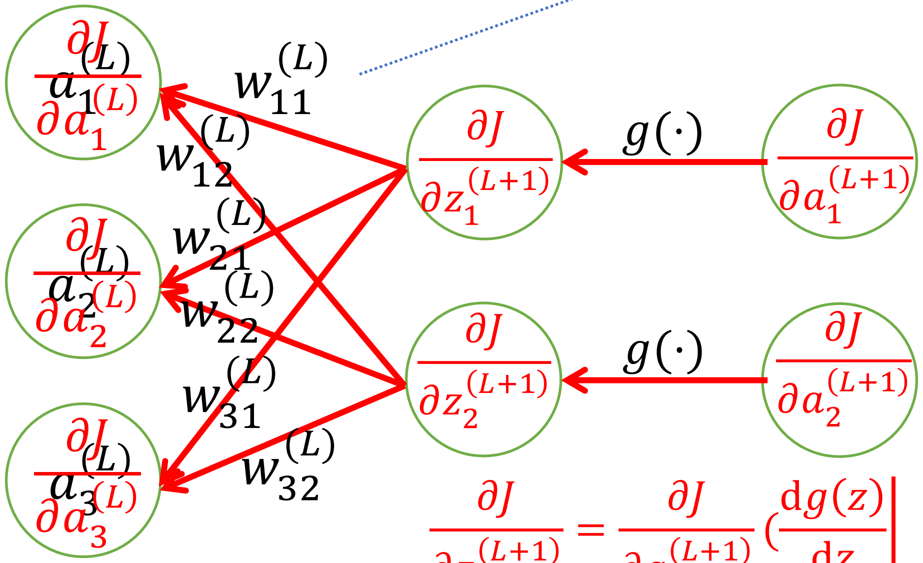
定义：设 $n \times m$ 维矩阵 $\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}$，则标量 y 对矩阵 $\boldsymbol{X}$ 的导数为：

$$\frac{\partial y}{\partial \boldsymbol{X}} = \begin{bmatrix} \frac{\partial y}{\partial x_{11}} & \frac{\partial y}{\partial x_{12}} & \cdots & \frac{\partial y}{\partial_{1m}} \\ \frac{\partial y}{\partial x_{21}} & \frac{\partial y}{\partial x_{22}} & \cdots & \frac{\partial y}{\partial x_{2m}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial x_{n1}} & \frac{\partial y}{\partial x_{n2}} & \cdots & \frac{\partial y}{\partial x_{nm}} \end{bmatrix}.$$

矩阵形式

- 反向传播

$$\frac{\partial J}{\partial a_j^{(L)}} = \sum_{i=1}^{2} w_{ji}^{(L)} \frac{\partial J}{\partial z_i^{(L+1)}} \qquad \frac{\partial J}{\partial w_{ij}^{(L)}} = \frac{\partial J}{\partial z_j^{(L+1)}} a_i^{(L)} \qquad \frac{\partial J}{\partial b_i^{(L)}} = \frac{\partial J}{\partial z_i^{(L+1)}}$$



$$\frac{\partial J}{\partial z_i^{(L+1)}} = \frac{\partial J}{\partial a_i^{(L+1)}} \left(\frac{\mathrm{d}g(z)}{\mathrm{d}z} \Big|_{z=z_i^{(L+1)}} \right)$$

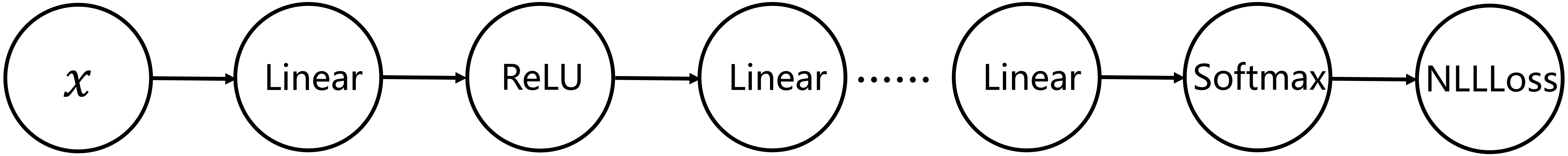
Layer L

通用矩阵形式：

- $\frac{\partial J}{\partial z^{(L+1)}} = \frac{\partial J}{\partial a^{(L+1)}} \odot \left(\frac{dg(z)}{dz} \Big|_{z=z^{(L+1)}} \right) \in \mathbb{R}^{d_{L+1}}$
- $\frac{\partial J}{\partial b^{(L)}} = \frac{\partial J}{\partial z^{(L+1)}} \in \mathbb{R}^{d_{L+1}}$
- $\frac{\partial J}{\partial W^{(L)}} = \frac{\partial J}{\partial z^{(L+1)}} {a^{(L)}}^T \in \mathbb{R}^{d_{L+1} \times d_L}$
- $\frac{\partial J}{\partial a^{(L)}} = {W^{(L)}}^T \frac{\partial J}{\partial z^{(L+1)}} \in \mathbb{R}^{d_L}$
- $a^{(L)} \in \mathbb{R}^{d_L}$, $z^{(L+1)} \in \mathbb{R}^{d_{L+1}}$, $a^{(L+1)} \in \mathbb{R}^{d_{L+1}}$
- $W^{(L)} \in \mathbb{R}^{d_{L+1} \times d_L}$, $b^{(L)} \in \mathbb{R}^{d_{L+1}}$
- $\odot$ 为逐元素相乘

计算图

- 矩阵形式允许我们将神经网络简化为层与层连接的计算图形式



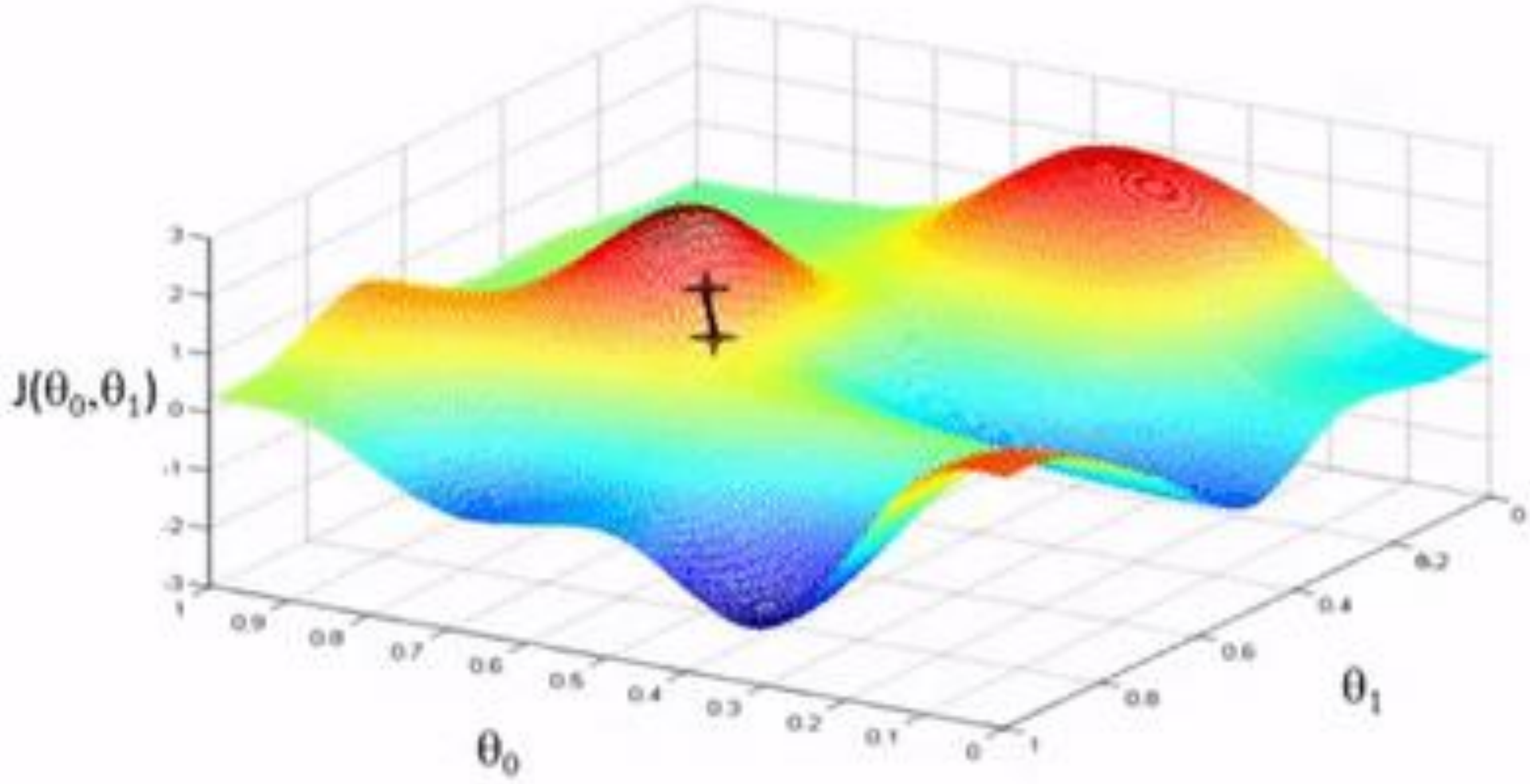
- 定义每种层的前向/反向传播
- 定义计算图通用的输入格式和计算顺序
- 用代码顺序定义网络结构
- 批量将训练数据输入计算图，执行前向+反向传播
- 求出所有参数的梯度后，执行一次梯度下降
- 用更新后的参数进行下次前向+反向传播
- 迭代梯度下降直到收敛

```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.computation_graph = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10),
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.computation_graph(x)
        return logits
```

复习

- 我们已经学习了梯度下降(GD)：每次沿梯度的相反方向走一个步长，直至目标函数收敛，或者达到预设的最大迭代次数。



全量梯度下降(Full GD)

- Full GD、 SGD、 mini-batch GD之间的核心区别在于，每一次迭代更新模型参数，使用的数据量(batch size)不同。
- 原始的梯度下降使用全量数据(Full GD)：

```
# Full Gradient Descent

while True:
    data_batch = data
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
    weights += - step_size * weights_grad # perform parameter update
```

- 全量梯度下降每一步都朝着使“在全部训练数据上损失”降低最快的方向更新
- 然而这种方法每一次迭代都需要全部数据参与，对于大数据集如拥有120万张图片的ImageNet，经常无法一次装下所有的训练数据

随机梯度下降(SGD)

北京大学 PEKING UNIVERSITY

- 随机梯度下降方法每次迭代时，从全量数据中随机抽取**一个**样本：

```
# Stochastic Gradient Descent

while True:
    data_batch = random_sample_training_data(data) # sample one example from dataset
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
    weights += - step_size * weights_grad
```

- 随机梯度下降大大减小了每次迭代的开销，使得超大训练集也可以训练。

- 然而，这种方法每次迭代均朝着使“某个随机样本的损失”降低最快的方向更新所有参数。这个梯度方向经常严重**偏离**真正的全量数据损失的梯度。
- 因此目标函数的波动可能很大，模型较难收敛。

小批量梯度下降(mini-batch GD)

- 小批量梯度下降是前两种方法的折衷：每次迭代从全量数据中抽取**一批**样本：

```
# Minibatch Gradient Descent

while True:
    data_batch = sample_training_data(data, 256) # sample 256 examples
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
    weights += - step_size * weights_grad
```

小批量梯度下降的计算效率相比全量梯度下降更高，可以根据硬件调整批大小（batch size）。

- 小批量梯度下降的优化方向是对全量梯度下降方向的较好近似。
- 由于具有更好的效率和效果，小批量梯度下降方法被广泛使用。
- 值得一提的是，**随机梯度下降(SGD)是batch size为1的小批量梯度下降**。因此在各种场景中，经常使用随机梯度下降的称呼来代替小批量梯度下降。
- 同理，全量梯度下降相当于batch size为n的小批量梯度下降。
- 实际中，batch size是一个重要的超参数，可以通过验证集调整。

谢谢

北京大學
PEKING UNIVERSITY