




人工智能：模型与算法

深度学习

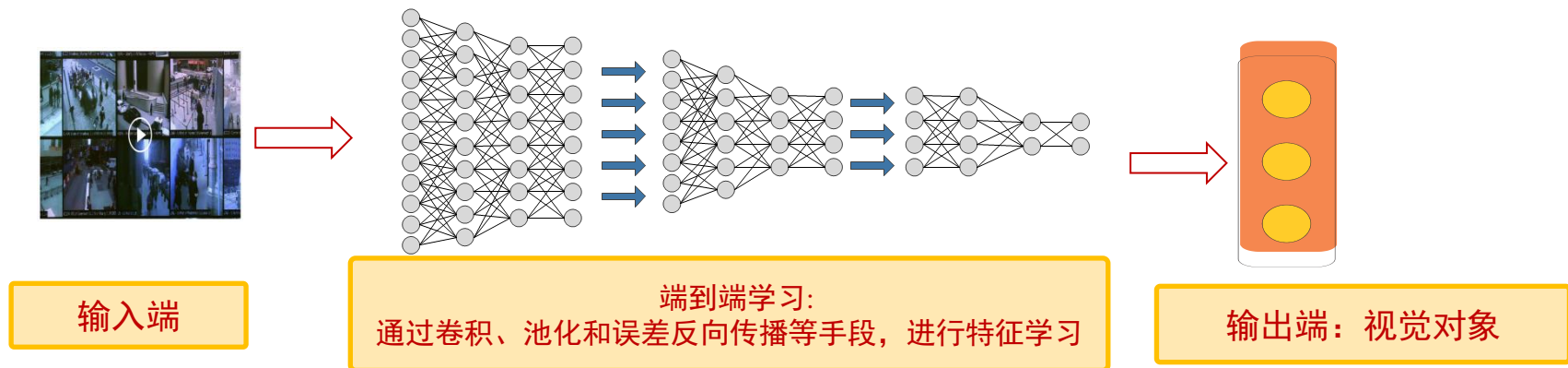
吴飞

浙江大学计算机学院

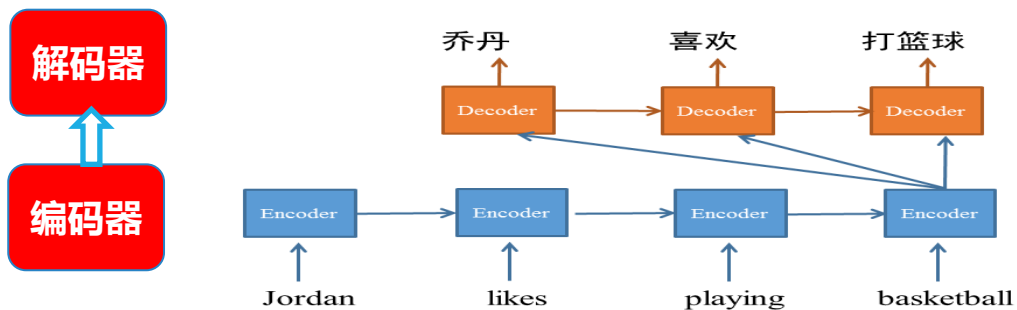
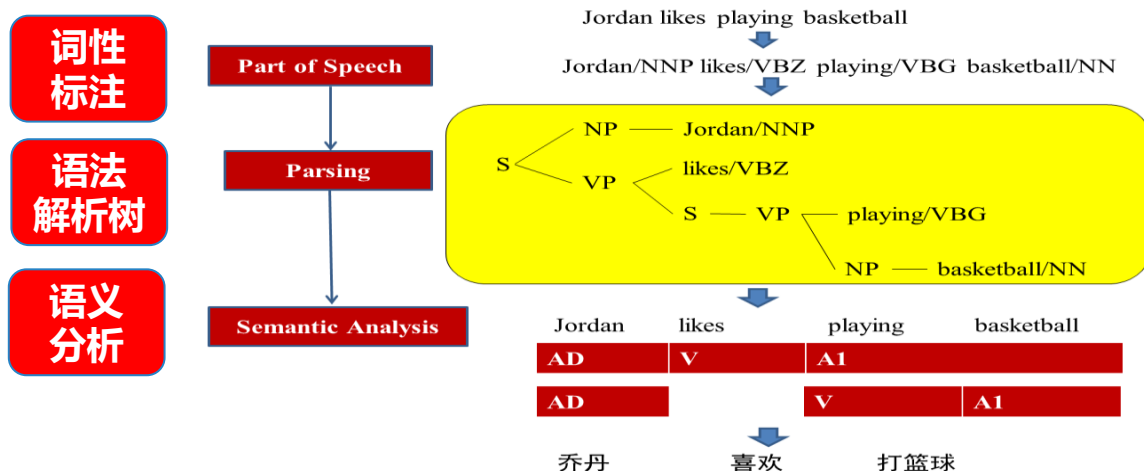
提纲

- 1、前馈神经网络
 - 2、卷积神经网络
 - 3、自然语言理解与视觉分析
- 

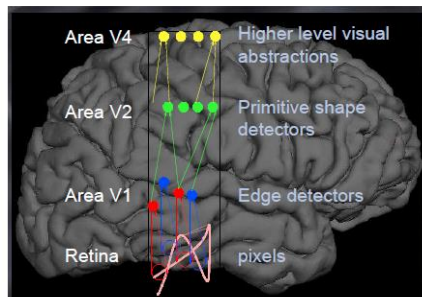
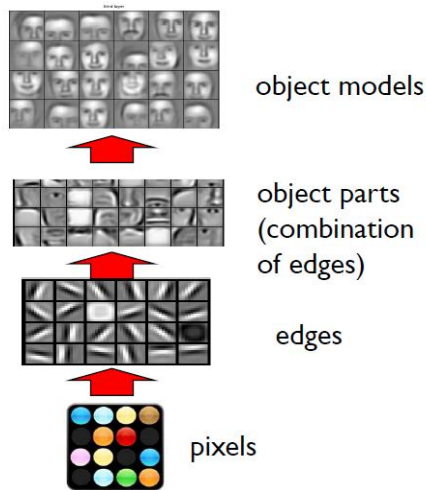
浅层学习 Versus 深度学习：从分段学习到端到端学习



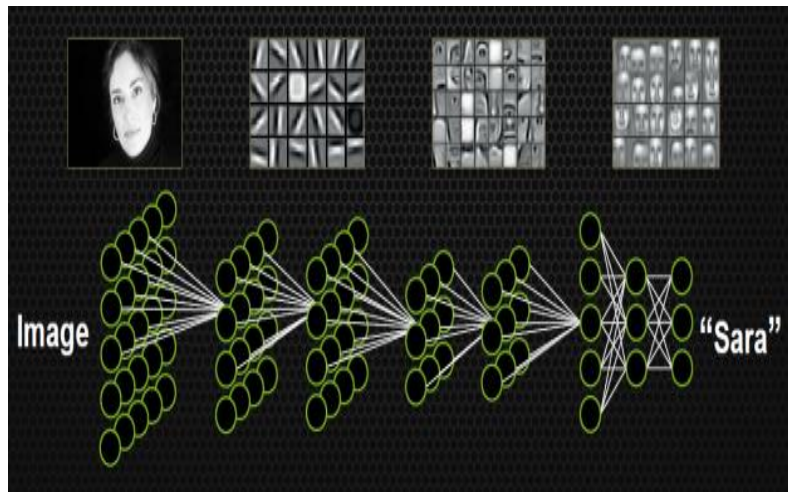
浅层学习 Versus 深度学习：从分段学习到端到端学习



深度学习：以端到端的方式逐层抽象、逐层学习



Slide credit: Andrew Ng



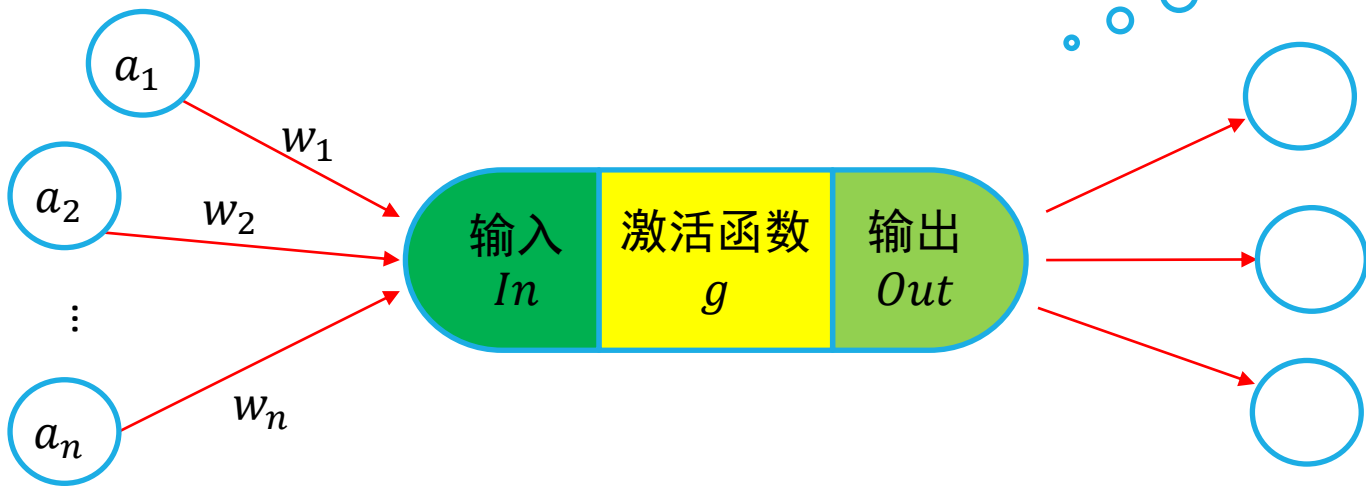
- 深度学习所得模型可视为一个复杂函数
- 非线性变换与映射的过程：像素点→语义

刻画神经元功能的数学模型

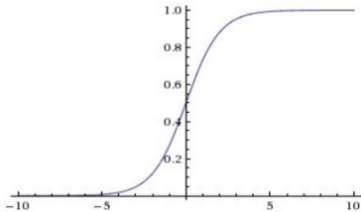
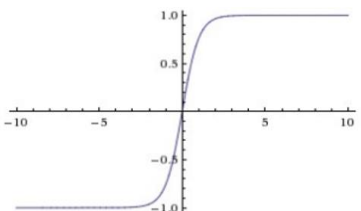
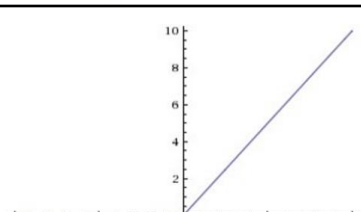
神经元是深度学习模型中基本单位，可以如下刻画神经元功能：

1. 对相邻前向神经元输入信息进行加权累加： $In = \sum_{i=1}^n w_i * a_i$
2. 对累加结果进行非线性变换（通过激活函数）： $g(x)$
3. 神经元的输出： $Out = g(In)$

神经元越多、非线性映射越复杂

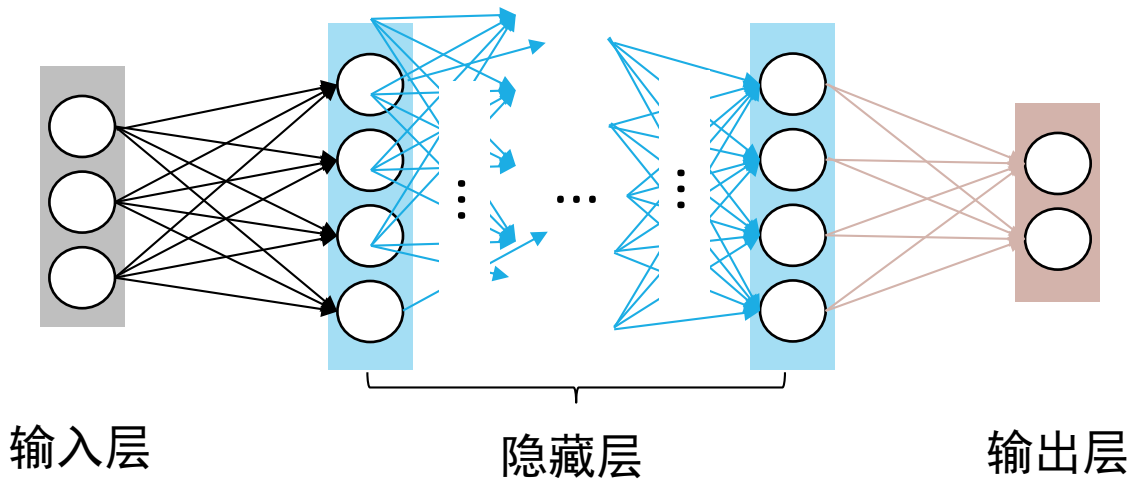


常用的激活函数：对输入信息进行非线性变换

激活函数名称	函数功能	函数图像	函数求导
Sigmoid	$f(x) = \frac{1}{1 + e^{-x}}$		$f'(x) = f(x)(1 - f(x))$
Tanh	$f(x) = \frac{2}{1 + e^{-2x}} - 1$		$f'(x) = 1 - f(x)^2$
Relu (Rectified Linear Unit)	$f(x) = \begin{cases} 0, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$		$f'(x) = \begin{cases} 0, & \text{for } x < 0 \\ 1, & \text{for } x \geq 0 \end{cases}$

前馈神经网络 (feedforward neural network)

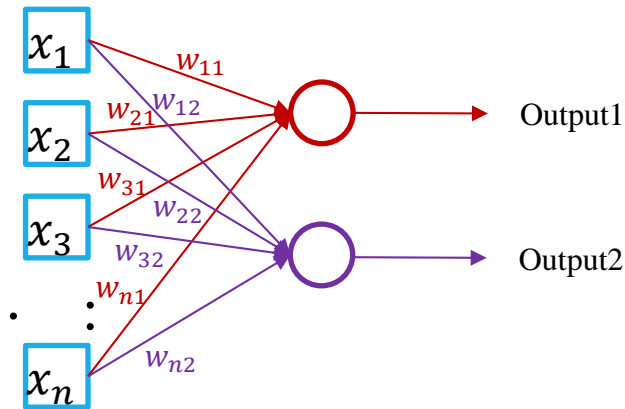
- 各个神经元接受前一级的输入，并输出到下一级，模型中没有反馈
- 层与层之间通过“全连接”进行链接，即两个相邻层之间的神经元完全成对连接，但层内的神经元不相互连接。



前馈神经网络 (feedforward neural network)

感知机网络 (Perceptron Networks) 是一种特殊的前馈神经网络：

- 无隐藏层，只有输入层/输出层
- 无法拟合复杂的数据



$w_{ij} (1 \leq i \leq n, 1 \leq j \leq 2)$
构成了感知机模型参数

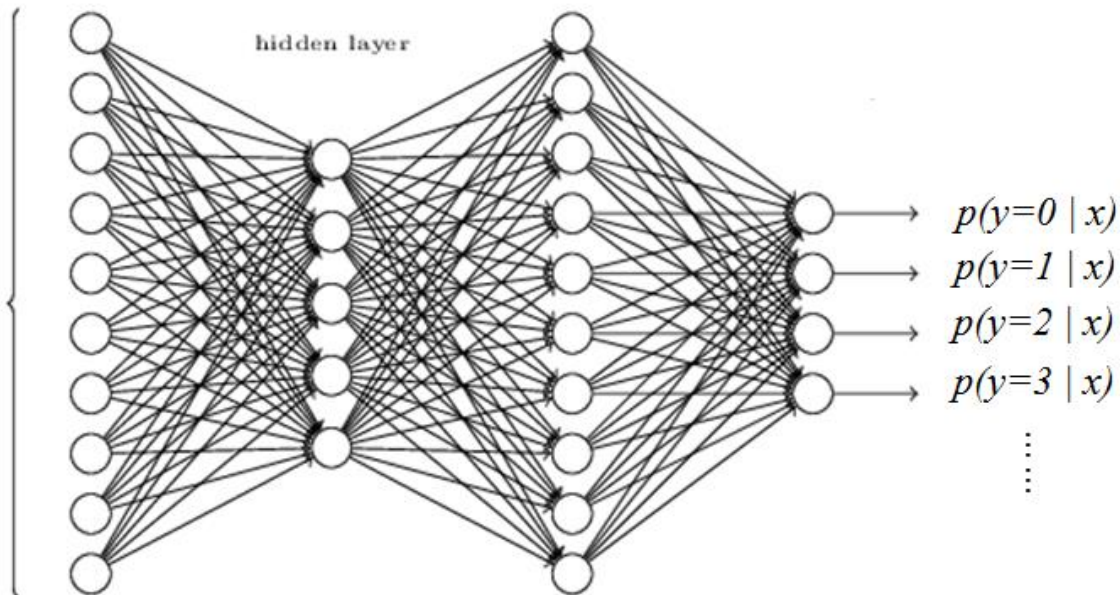
问题：如何优化网络参数？

$$w_{ij} (1 \leq i \leq n, 1 \leq j \leq m)$$

n 为神经网络层数、 m 为每层中神经元个数



input layer
(784 neurons)



问题：如何优化网络参数？

从标注数据出发，优化模型参数

- 标注数据： $(x_i, y_i) (1 \leq i \leq N)$
- 评分函数(scoring function)将输入数据映射为类别置信度大小： $s = f(x) = W\varphi(x)$
- 损失函数来估量模型预测值与真实值之间的差距。损失函数给出的差距越小，则模型鲁棒性就越好。常用的损失函数有softmax或者SVM。

softmax
$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

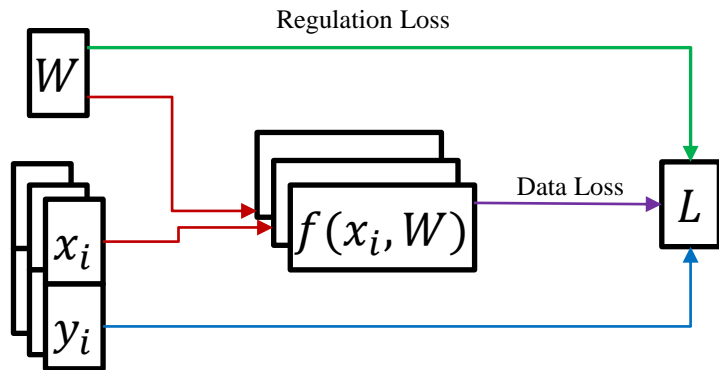
分类正确样本的Softmax值所占比重越大，其Loss越小

SVM
$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

分类正确样本置信度比分类错误样本置信度至少大1

Full loss
$$L = \frac{1}{N} \sum_{i=1}^N L_i + R(W)$$

优化目标：损失值小、参数不复杂



参数优化：梯度下降 (Gradient Descent)

梯度下降算法是一种使得损失函数最小化的方法。一元变量所构成函数 f 在 x 处梯度为：

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

- 在多元函数中，梯度是对每一变量所求导数组成的向量
- 梯度的反方向是函数值下降最快的方向



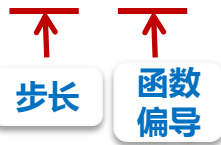
梯度下降 (Gradient Descent)

- 假设损失函数 $f(x)$ 是连续可微的多元变量函数，其泰勒展开如下(Δx 是微小的增量):

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)(\Delta x)^2 + \dots + \frac{1}{n!}f^{(n)}(x)(\Delta x)^n$$
$$f(x + \Delta x) - f(x) \approx (\nabla f(x))^T \Delta x$$

- 因为我们的目的是最小化损失函数 $f(x)$ ，则 $f(x + \Delta x) < f(x)$ ，于是 $(\nabla f(x))^T \Delta x < 0$
- 因为 $(\nabla f(x))^T \Delta x = \|\nabla f(x)\| \|\Delta x\| \cos \theta$ ，为了得到最小化损失函数，可以沿着损失函数梯度的反方向 (i.e., $\theta = \pi$)来寻找使得损失函数最小的参数取值。

$$f(x + \Delta x) - f(x) = \|\nabla f(x)\| \|\Delta x\| \cos \theta = -\alpha \|\nabla f(x)\|$$

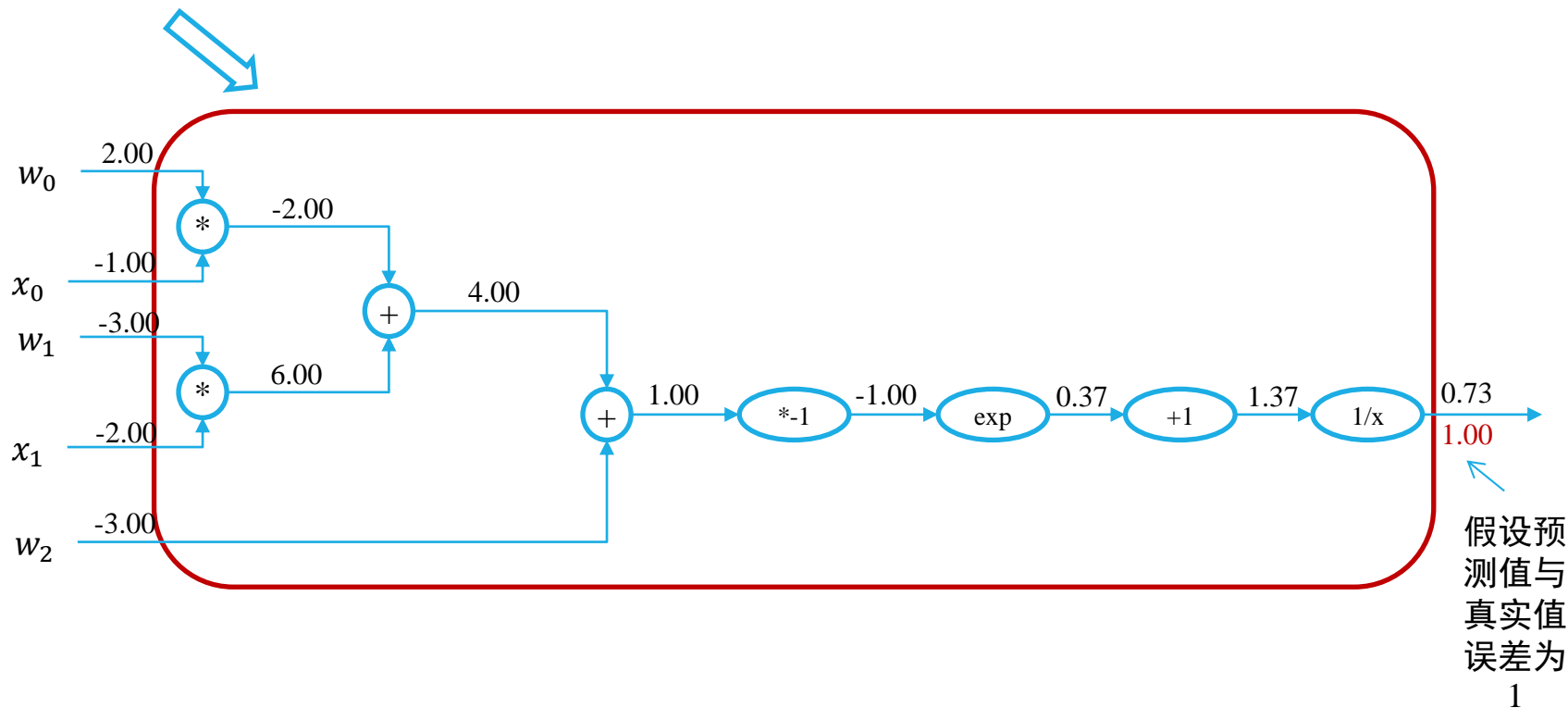


参数优化：误差反向传播 (error back propagation, BP)

- BP算法是一种将输出层误差反向传播给隐藏层进行参数更新的方法。
- 将误差从后向前传递，将误差分摊给各层所有单元，从而获得各层单元所产生的误差，进而依据这个误差来让各层单元负起各自责任、修正各单元参数。

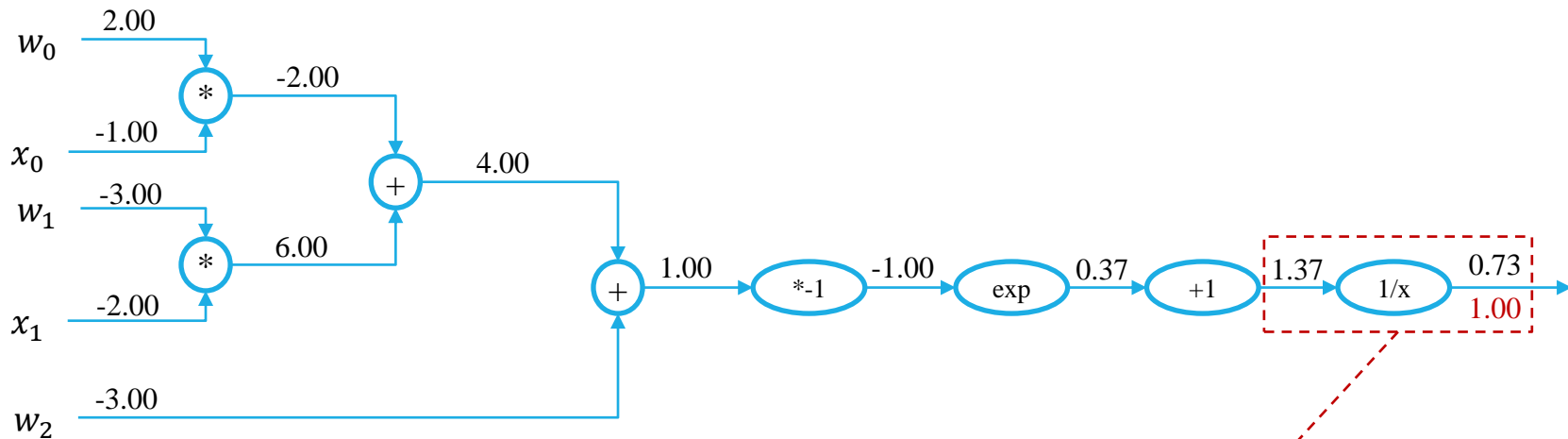
误差反向传播例子

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



误差反向传播例子

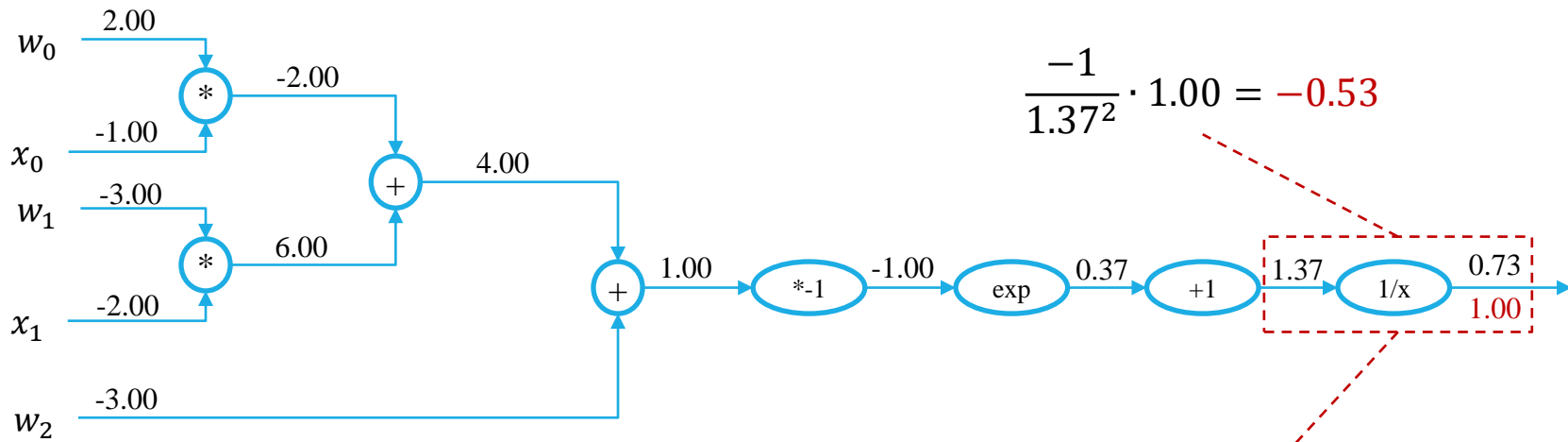
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



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

误差反向传播例子

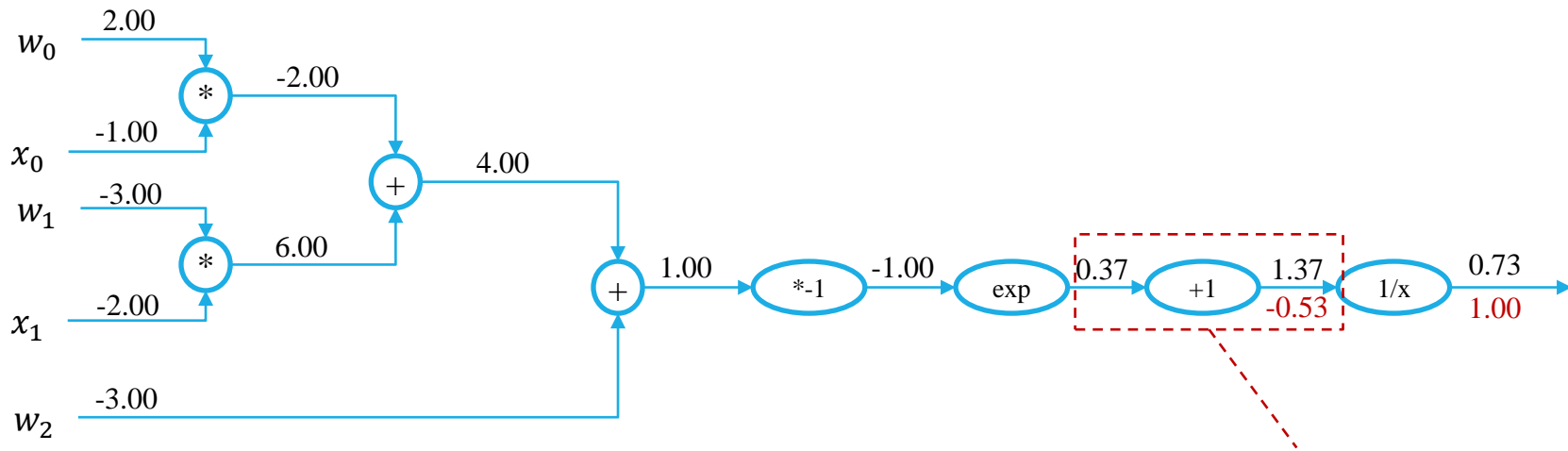
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



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

误差反向传播例子

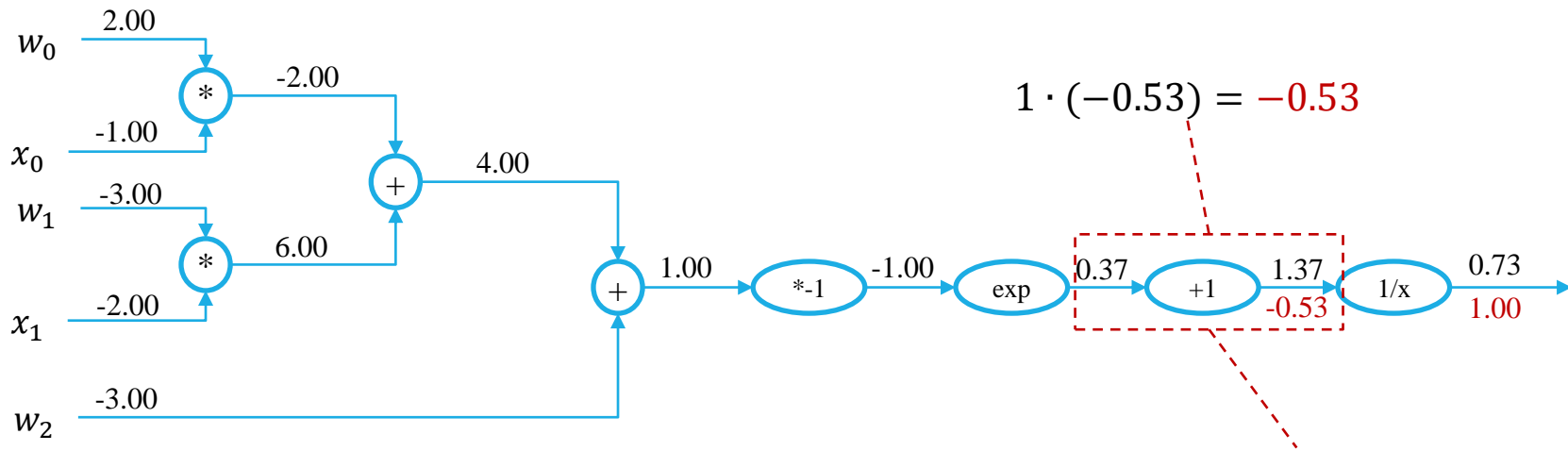
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



$$f(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

误差反向传播例子

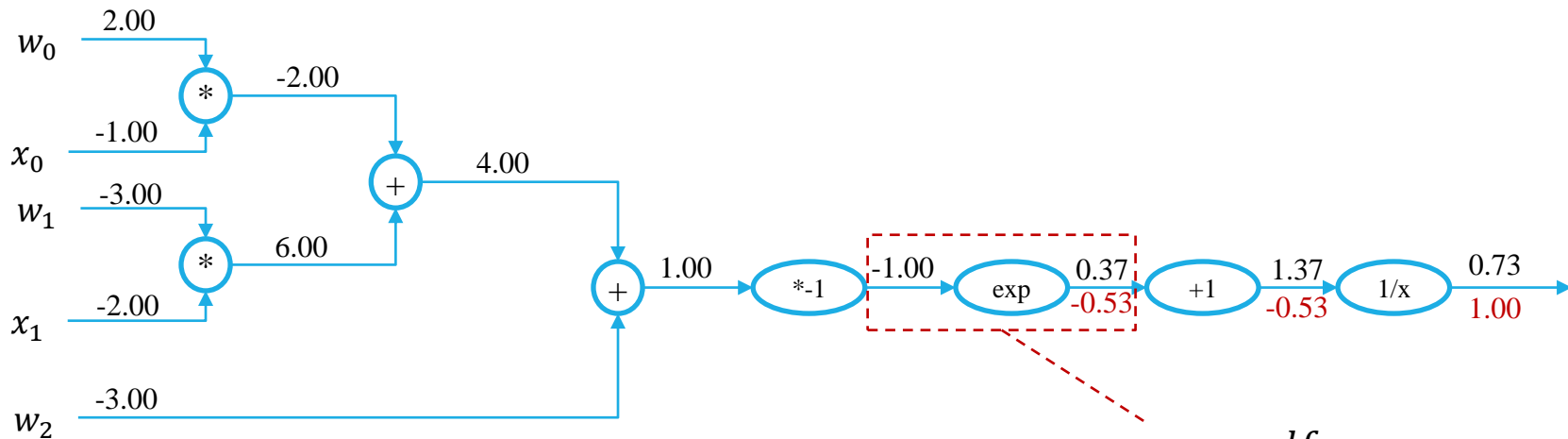
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



$$f(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

误差反向传播例子

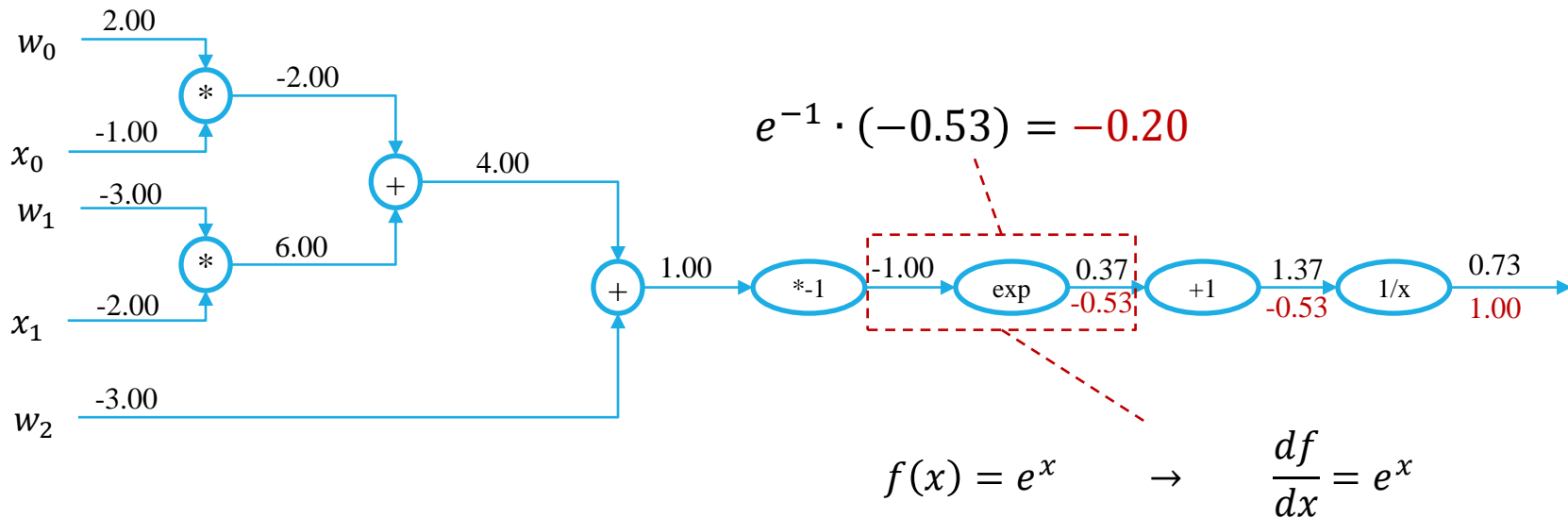
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

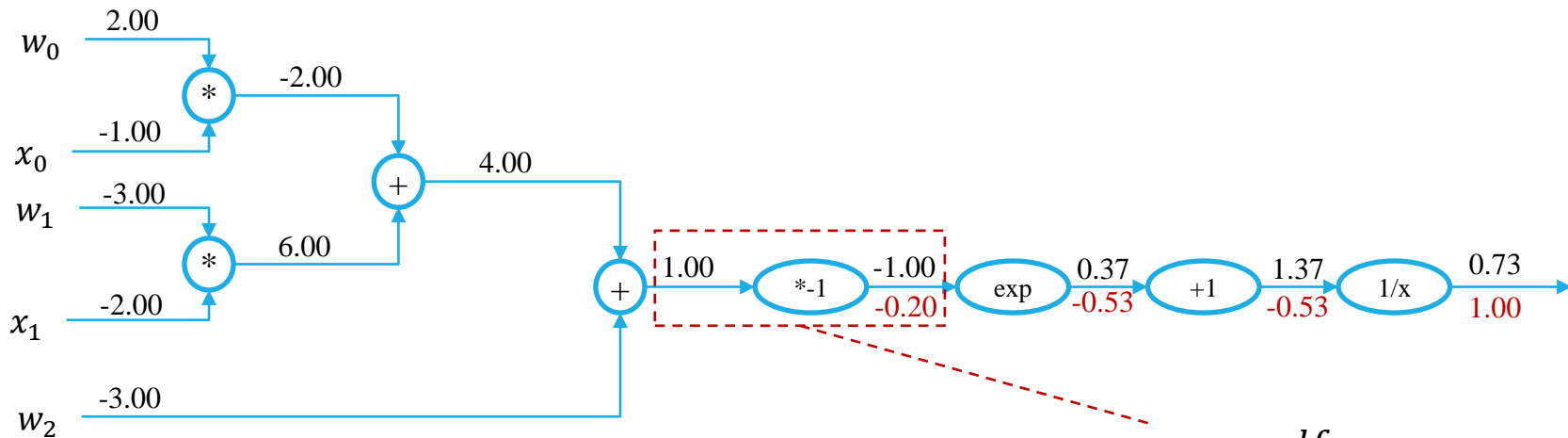
误差反向传播例子

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



误差反向传播例子

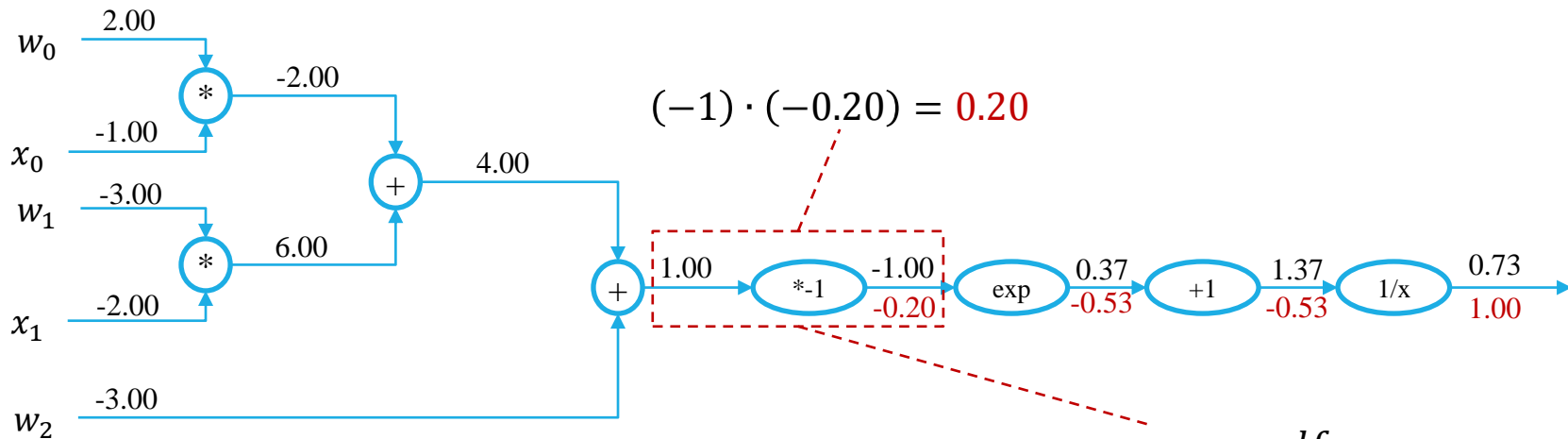
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



$$f(x) = ax \rightarrow \frac{df}{dx} = a$$

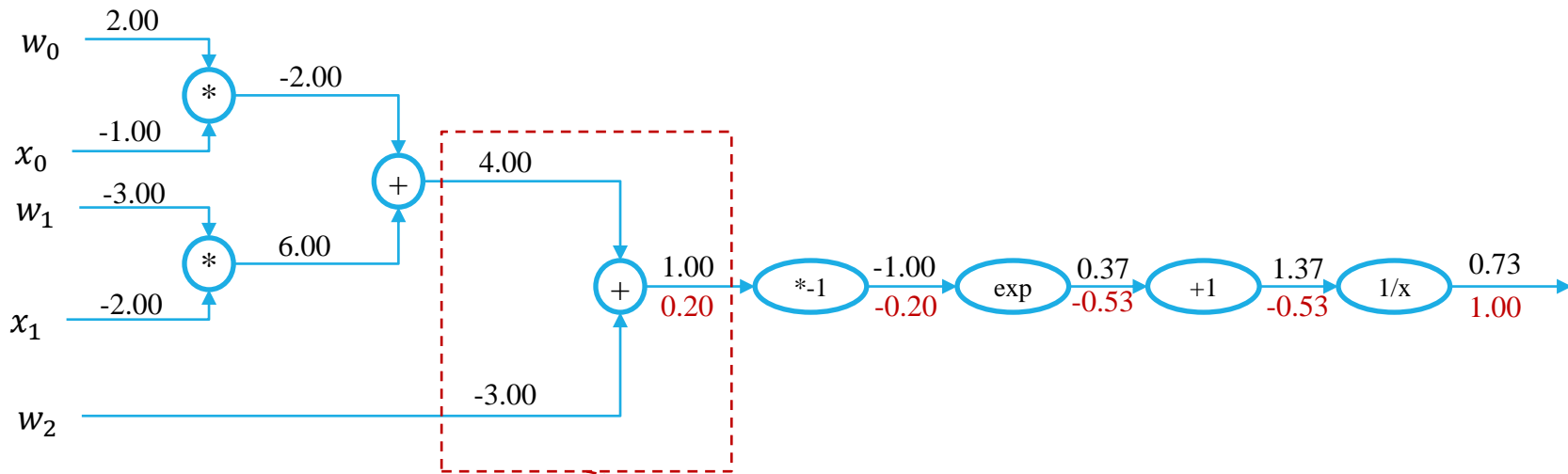
误差反向传播例子

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



误差反向传播例子

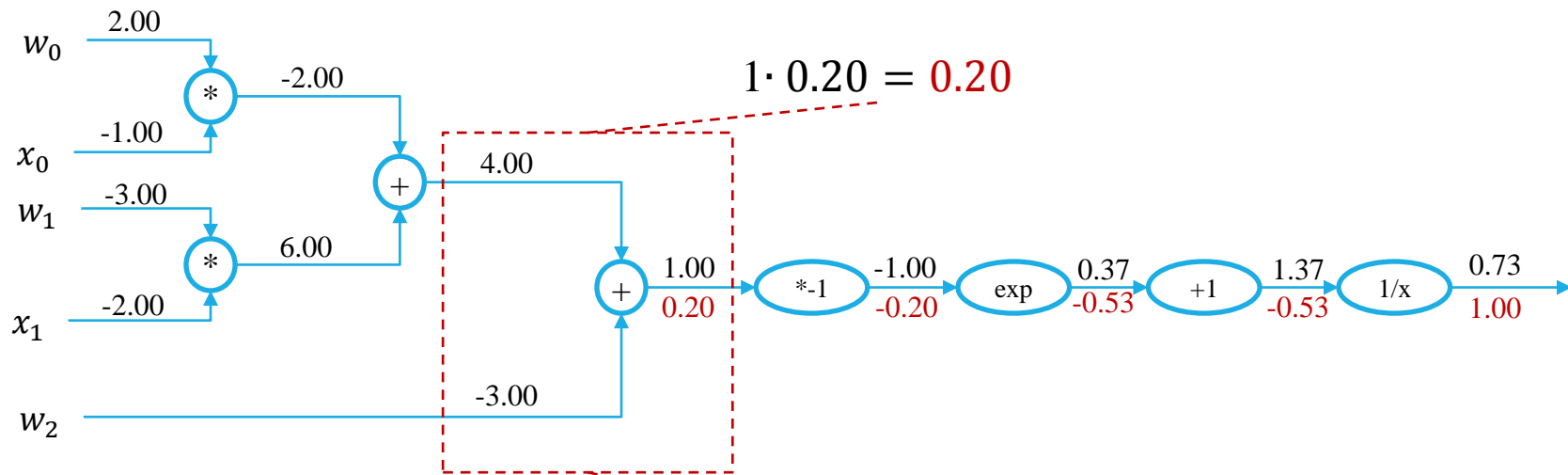
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



$$f(x) = x + x' \rightarrow \frac{df}{dx} = 1$$

误差反向传播例子

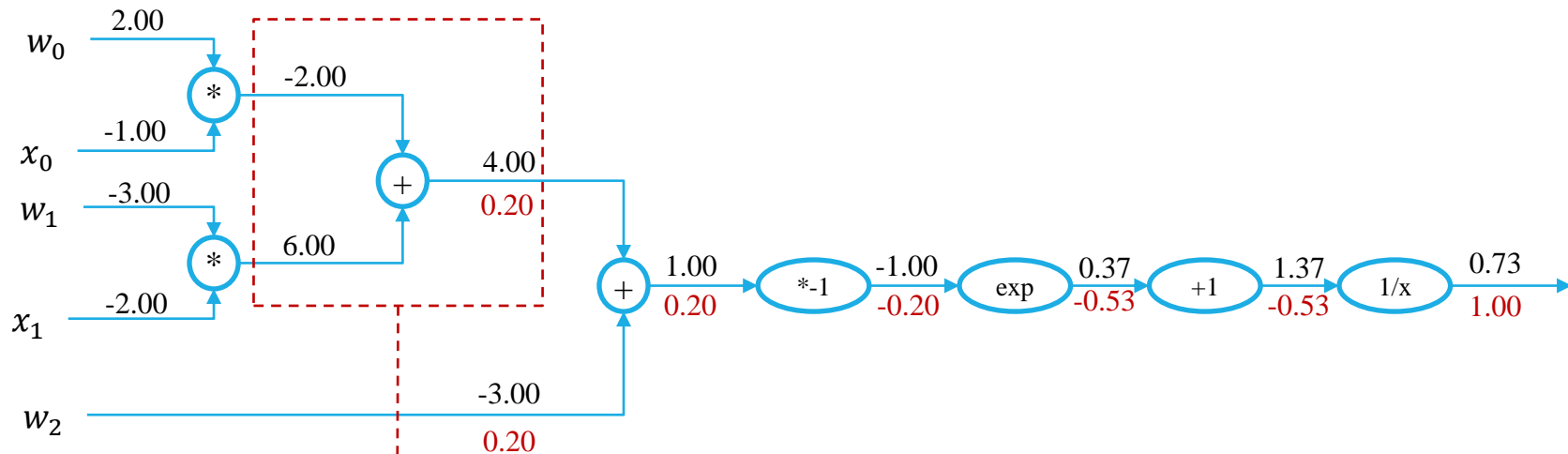
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



$$f(x) = x + x' \rightarrow \frac{df}{dx} = 1$$

误差反向传播例子

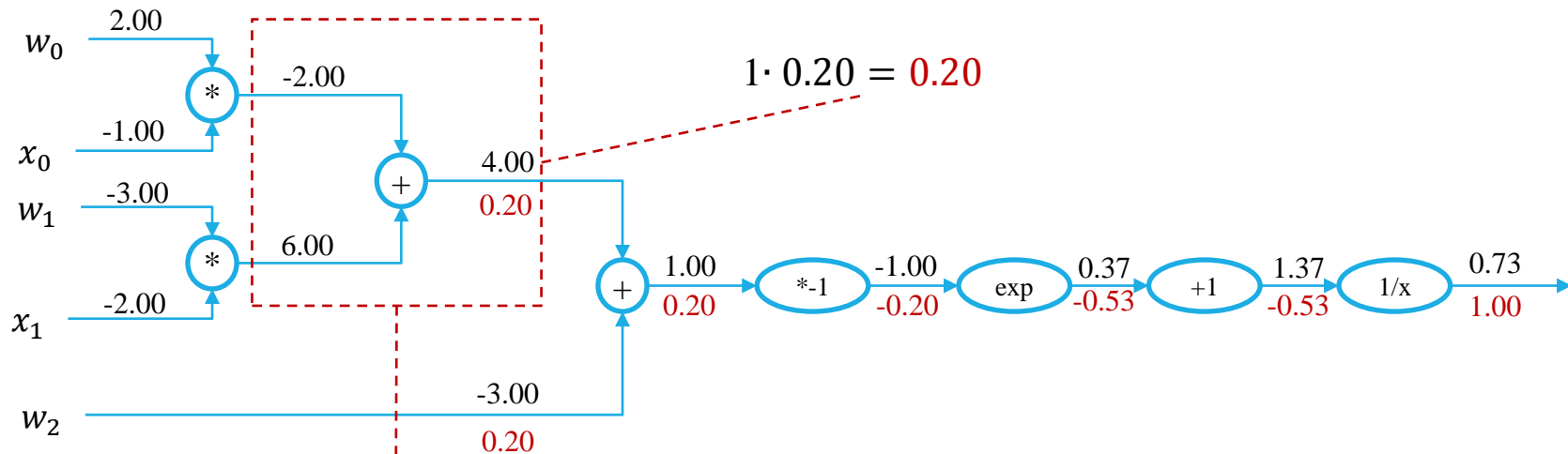
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



$$f(x) = x + x' \rightarrow \frac{df}{dx} = 1$$

误差反向传播例子

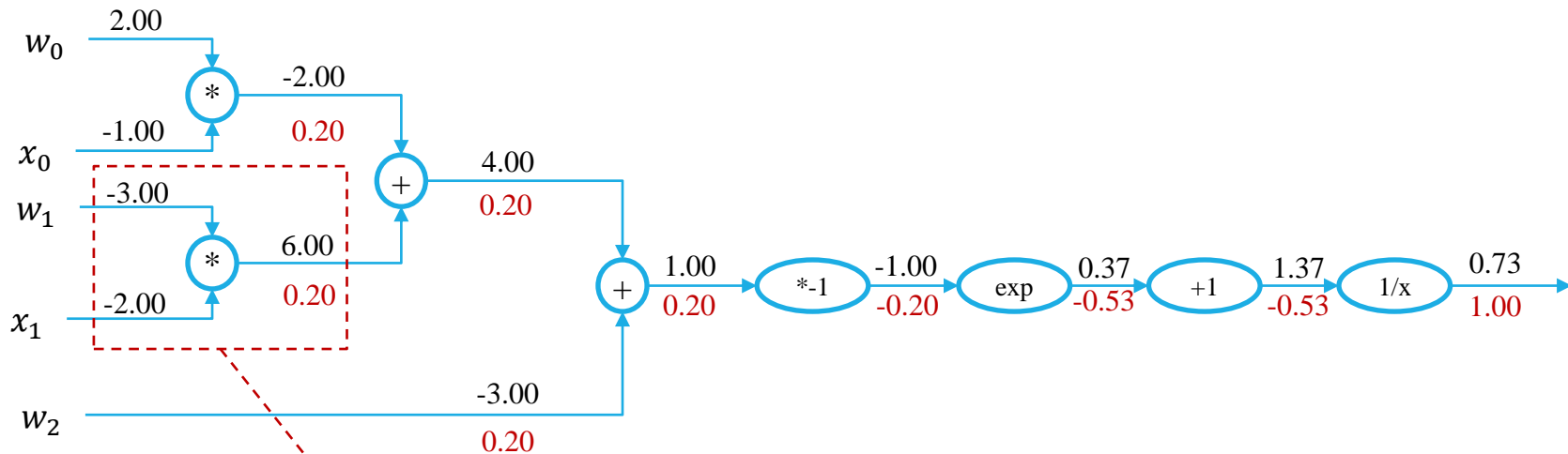
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



$$f(x) = x + x' \rightarrow \frac{df}{dx} = 1$$

误差反向传播例子

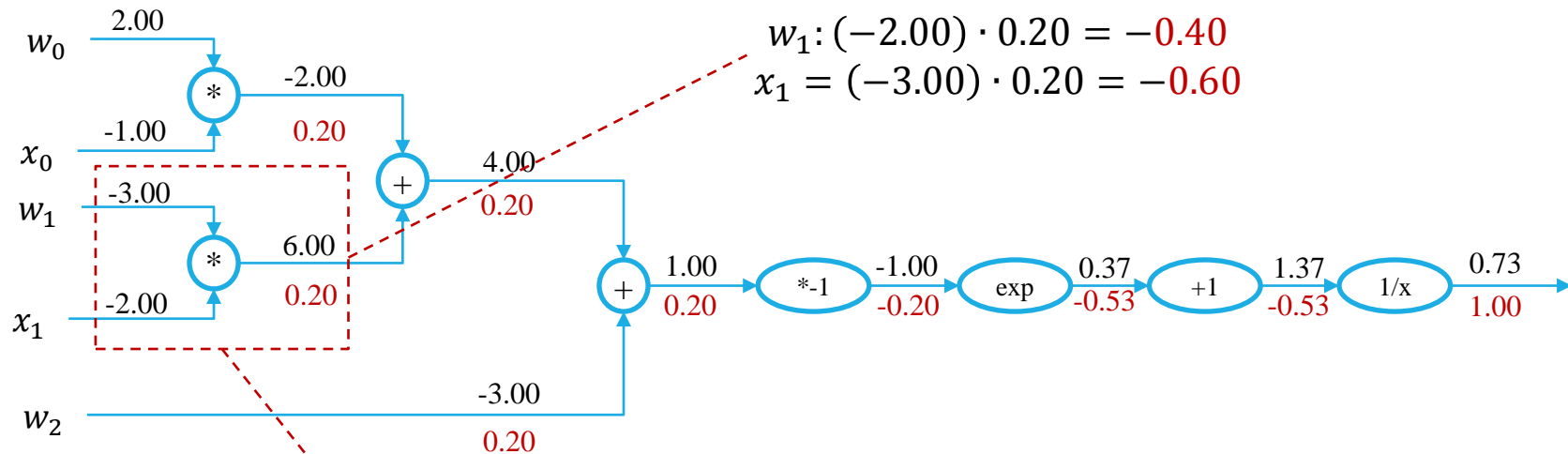
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



$$f(x) = x \cdot x' \rightarrow \frac{df}{dx} = x'$$

误差反向传播例子

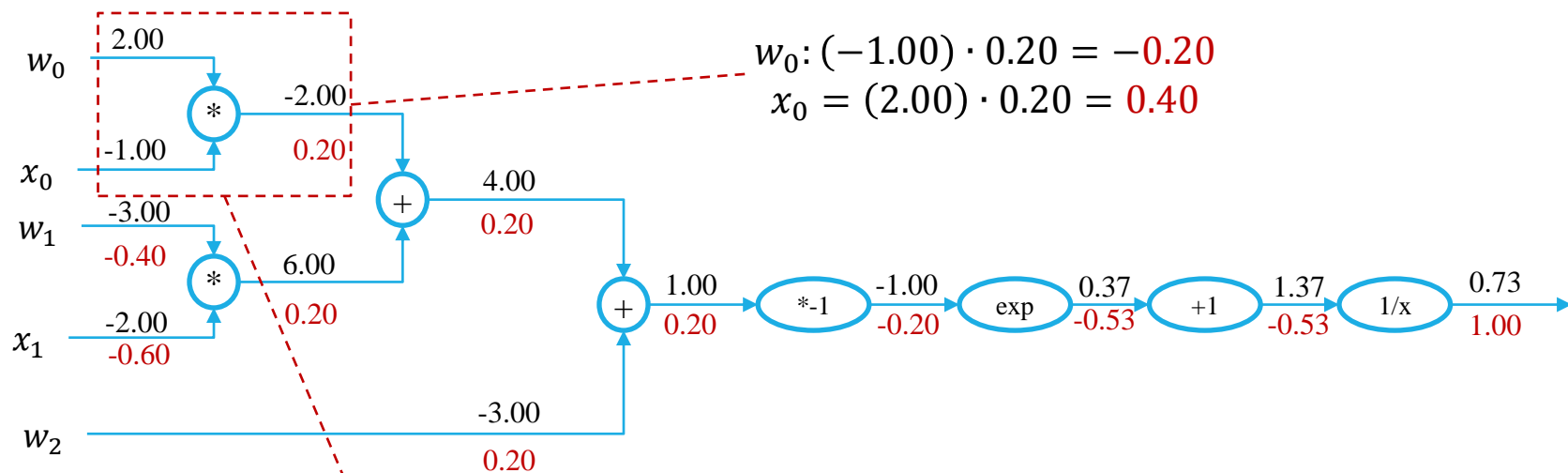
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



$$f(x) = x \cdot x' \rightarrow \frac{df}{dx} = x'$$

误差反向传播例子

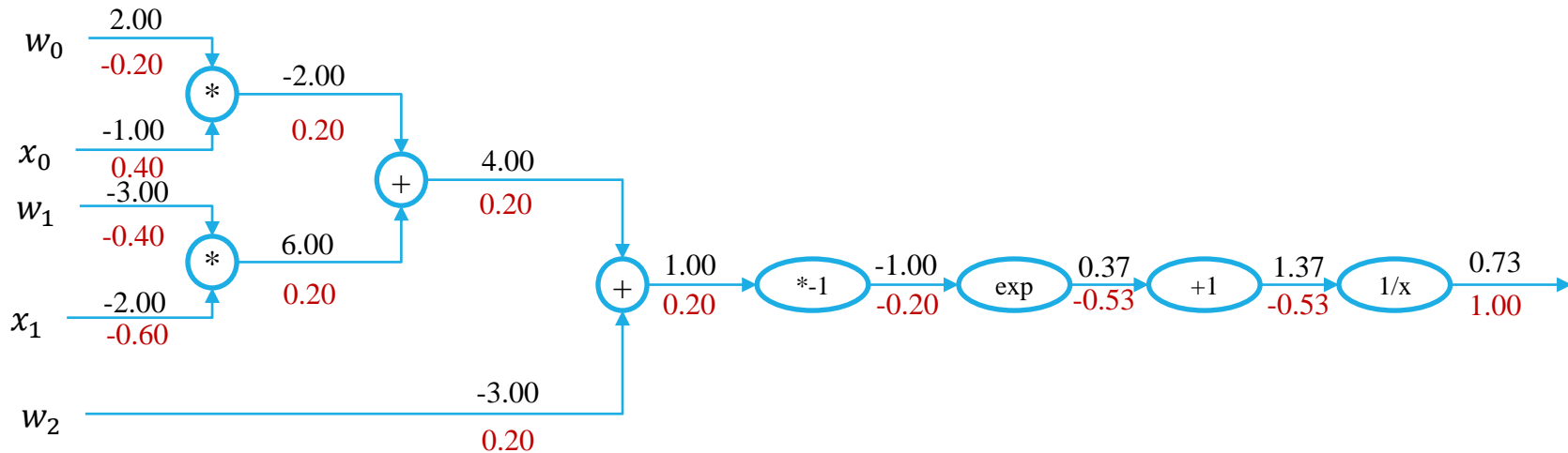
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



$$f(x) = x \cdot x' \rightarrow \frac{df}{dx} = x'$$

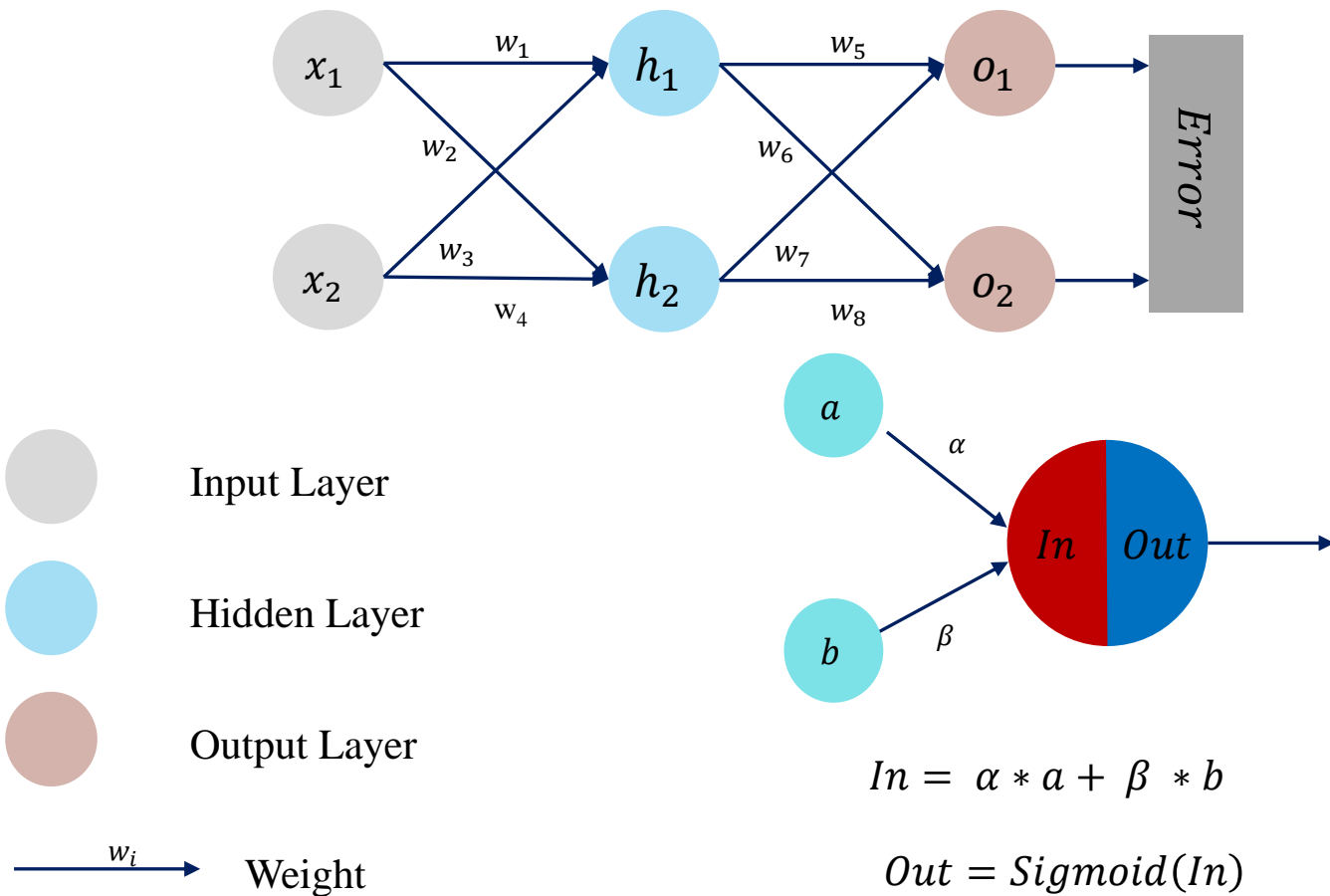
误差反向传播例子

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



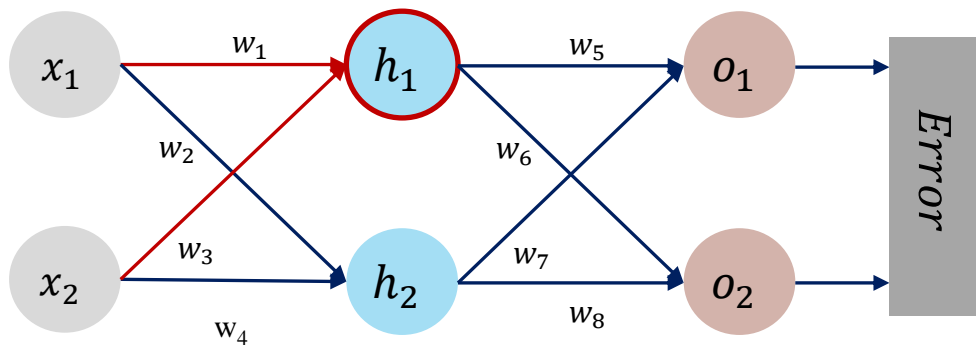
误差反向传播

误差反向传播：前馈神经网络



误差反向传播：前馈神经网络

前向传播

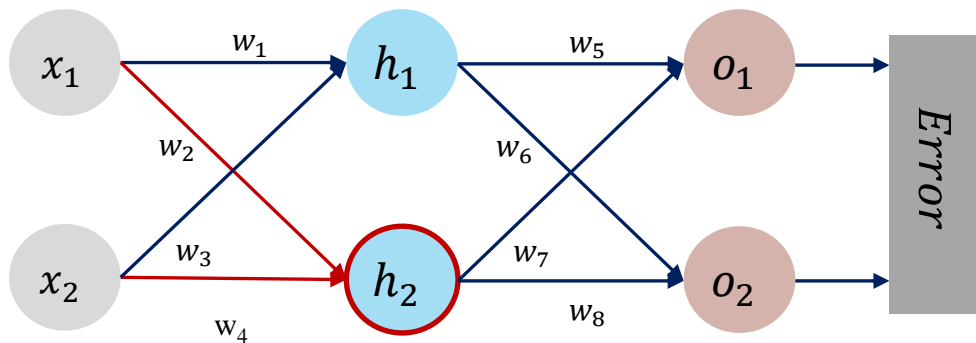


$$In_{h_1} = w_1 * x_1 + w_3 * x_2$$

$$h_1 = Out_{h_1} = Sigmoid(In_{h_1})$$

误差反向传播：前馈神经网络

前向传播

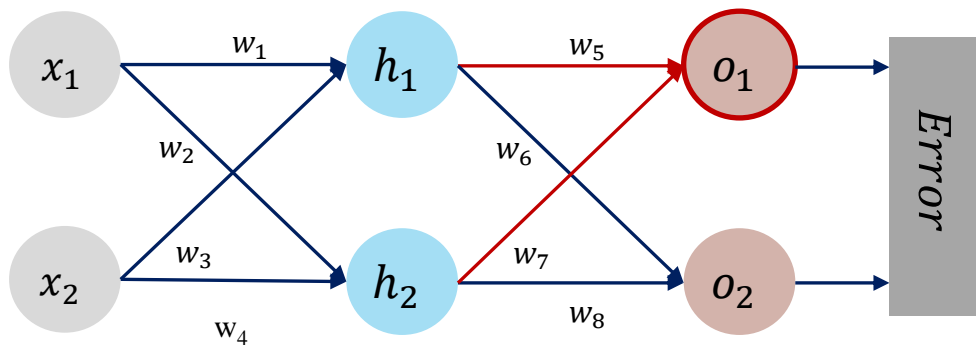


$$In_{h_2} = w_2 * x_1 + w_4 * x_2$$

$$h_2 = Out_{h_2} = Sigmoid(In_{h_2})$$

误差反向传播：前馈神经网络

前向传播

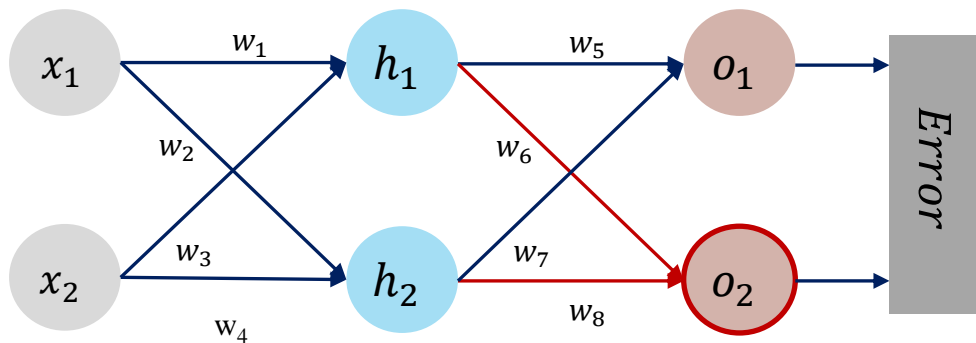


$$In_{o_1} = w_5 * h_1 + w_7 * h_2$$

$$o_1 = Out_{o_1} = Sigmoid(In_{o_1})$$

误差反向传播：前馈神经网络

前向传播

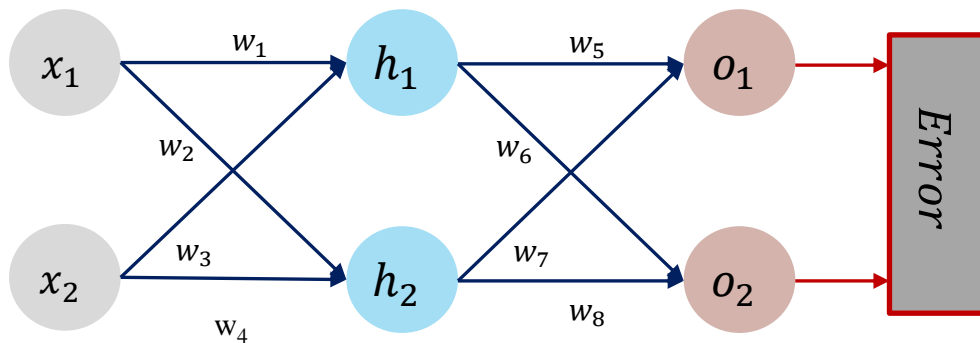


$$In_{o_2} = w_6 * h_1 + w_8 * h_2$$

$$o_2 = Out_{o_2} = Sigmoid(In_{o_2})$$

误差反向传播：前馈神经网络

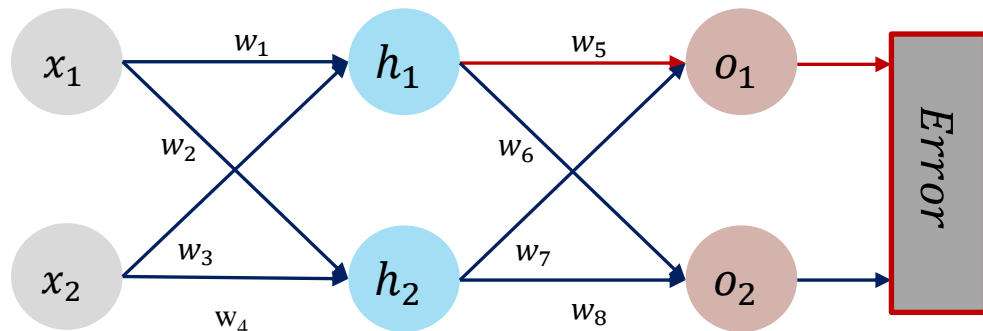
前向传播



$$Error = \frac{1}{2} \sum_{i=1}^2 (o_i - y_i)^2$$

误差反向传播：前馈神经网络

反向传播：梯度计算



1. 计算 $w_5 \sim w_8$ 的梯度(以 w_5 为例)

$$\delta_5 = \frac{\partial \text{Error}}{\partial w_5} = \frac{\partial \text{Error}}{\partial o_1} * \frac{\partial o_1}{\partial \text{In}_{o_1}} * \frac{\partial \text{In}_{o_1}}{\partial w_5}$$

where,

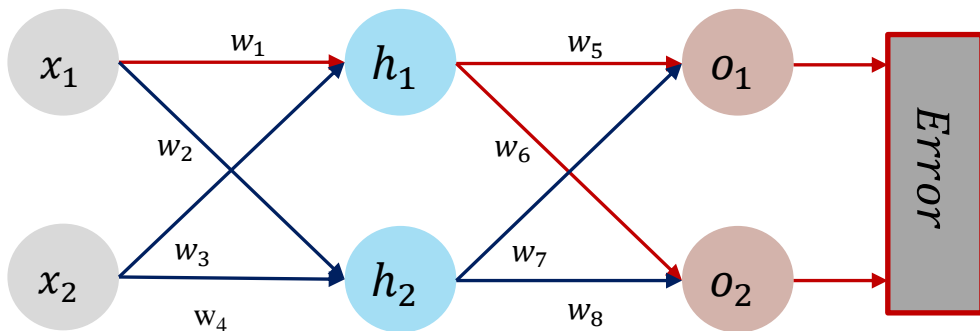
$$\frac{\partial \text{Error}}{\partial o_1} = o_1 - y_1 \quad \leftarrow \quad \text{Error} = \frac{1}{2} \sum_{i=1}^2 (o_i - y_i)^2$$

$$\frac{\partial o_1}{\partial \text{In}_{o_1}} = o_1 * (1 - o_1) \quad \leftarrow \quad o_1 = \text{Out}_{o_1} = \text{Sigmoid}(\text{In}_{o_1})$$

$$\frac{\partial \text{In}_{o_1}}{\partial w_5} = h_1 \quad \leftarrow \quad \text{In}_{o_1} = w_5 * h_1 + w_7 * h_2$$

误差反向传播：前馈神经网络

反向传播：梯度计算

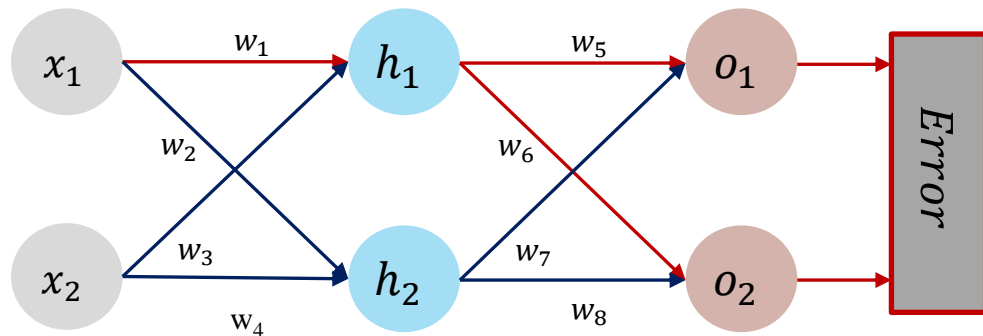


1. 计算 $w_1 \sim w_4$ 的梯度(以 w_1 为例)

$$\begin{aligned}\delta_1 &= \frac{\partial Error}{\partial w_1} = \frac{\partial Error}{\partial o_1} * \frac{\partial o_1}{\partial w_1} + \frac{\partial Error}{\partial o_2} * \frac{\partial o_2}{\partial w_1} \\ &= \frac{\partial Error}{\partial o_1} * \frac{\partial o_1}{\partial \ln_{o_1}} * \frac{\partial \ln_{o_1}}{\partial h_1} * \frac{\partial h_1}{\partial \ln_{h_1}} * \frac{\partial \ln_{h_1}}{\partial w_1} + \frac{\partial Error}{\partial o_2} * \frac{\partial o_2}{\partial \ln_{o_2}} * \frac{\partial \ln_{o_2}}{\partial h_1} * \frac{\partial h_1}{\partial \ln_{h_1}} * \frac{\partial \ln_{h_1}}{\partial w_1} \\ &= \left(\frac{\partial Error}{\partial o_1} * \frac{\partial o_1}{\partial \ln_{o_1}} * \frac{\partial \ln_{o_1}}{\partial h_1} + \frac{\partial Error}{\partial o_2} * \frac{\partial o_2}{\partial \ln_{o_2}} * \frac{\partial \ln_{o_2}}{\partial h_1} \right) * \frac{\partial h_1}{\partial \ln_{h_1}} * \frac{\partial \ln_{h_1}}{\partial w_1} \\ &= (\delta_5 + \delta_6) * \frac{\partial h_1}{\partial \ln_{h_1}} * \frac{\partial \ln_{h_1}}{\partial w_1}\end{aligned}$$

误差反向传播：前馈神经网络

反向传播：参数更新



2. 更新参数，其中 η 被称为学习率

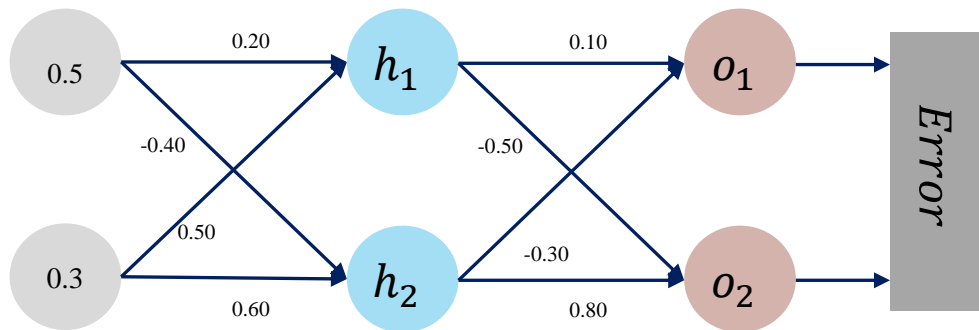
$$w'_i = w_i - \eta * \delta_i$$

↑ ↑ ↑

原有参数 步长 传递误差

误差反向传播：前馈神经网络

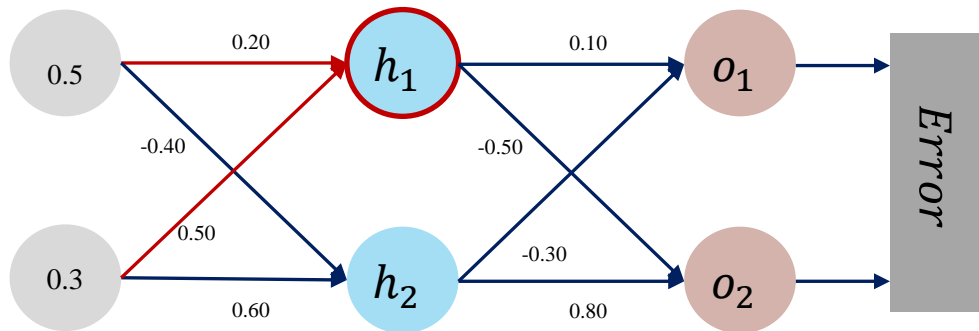
参数初始化



其中, $Error = \frac{1}{2}(o_1 - 0.23)^2 + \frac{1}{2}(o_2 - (-0.07))^2$, 这里0.23和-0.07是对输入样本数据(0.5, 0.3)的标注信息

误差反向传播：前馈神经网络

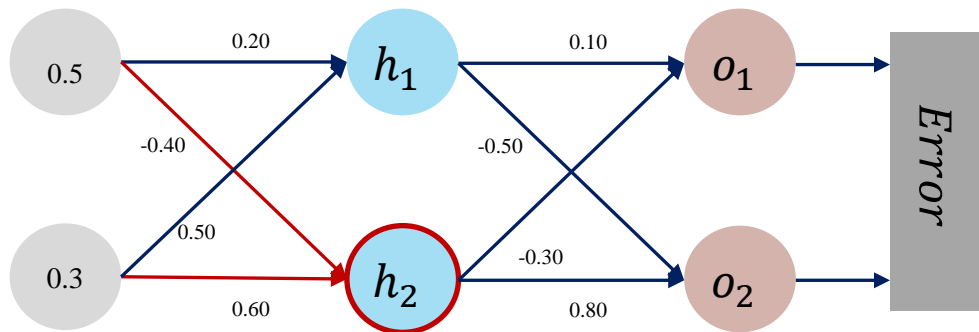
前向传播



$$h_1 = \text{sigmoid}(0.20 * 0.50 + 0.50 * 0.30) = 0.56$$

误差反向传播：前馈神经网络

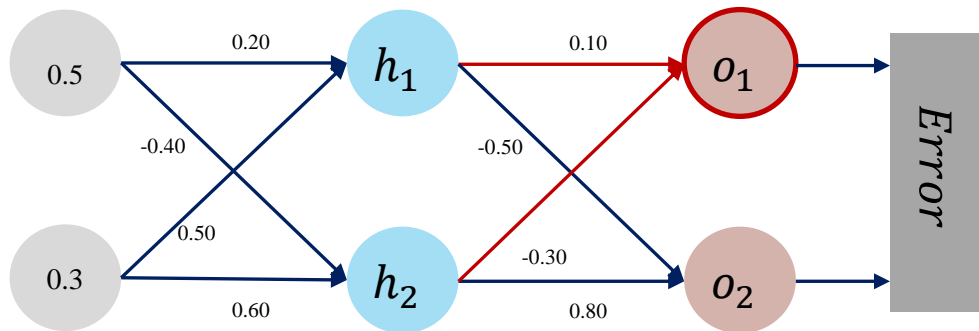
前向传播



$$h_2 = \text{sigmoid}(-0.40 * 0.50 + 0.60 * 0.30) = 0.50$$

误差反向传播：前馈神经网络

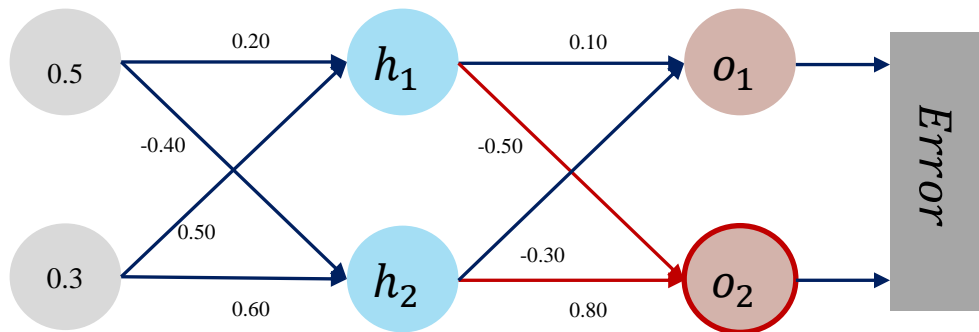
前向传播



$$o_1 = \text{sigmoid}(0.10 * 0.56 + (-0.30 * 0.50)) = 0.48$$

误差反向传播：前馈神经网络

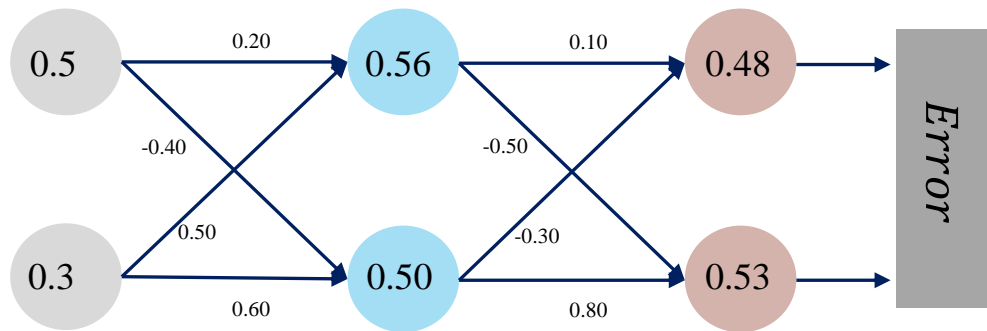
前向传播



$$o_2 = \text{sigmoid}(-0.50 * 0.56 + 0.80 * 0.50) = 0.53$$

误差反向传播：前馈神经网络

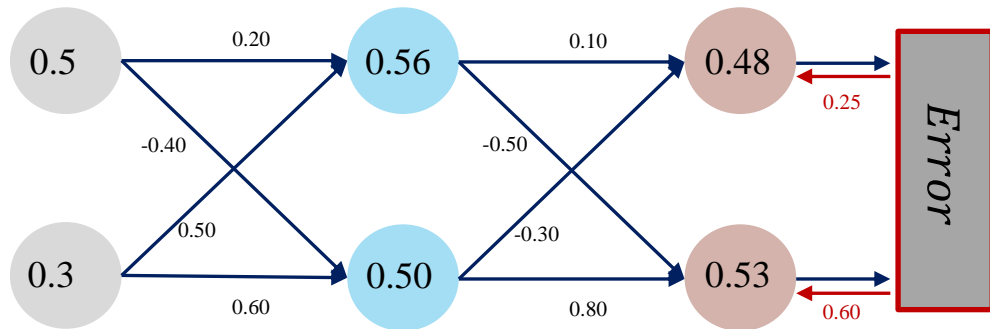
反向传播（假设学习速率 Learning Rate $\eta = 1$ ）



$$Error = \frac{1}{2} (0.48 - 0.23)^2 + \frac{1}{2} (0.53 - (-0.07))^2 = 0.21$$

误差反向传播：前馈神经网络

梯度计算

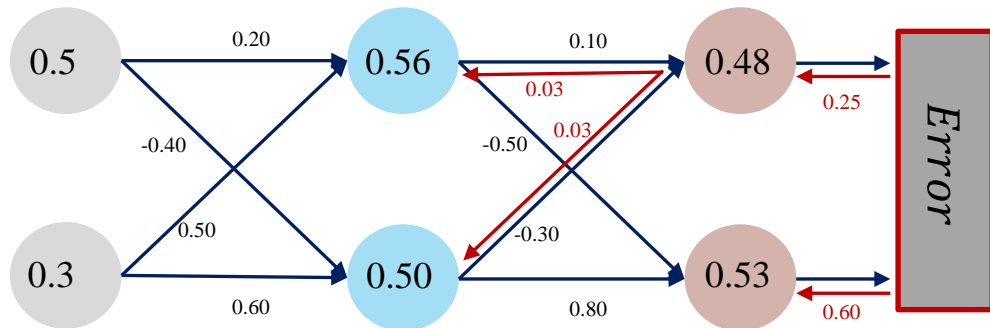


$$\delta_{o^1} = \frac{\partial Error}{\partial o_1} = o_1 - 0.23 = 0.25$$

$$\delta_{o^2} = \frac{\partial Error}{\partial o_2} = o_2 - (-0.07) = 0.60$$

误差反向传播：前馈神经网络

梯度计算

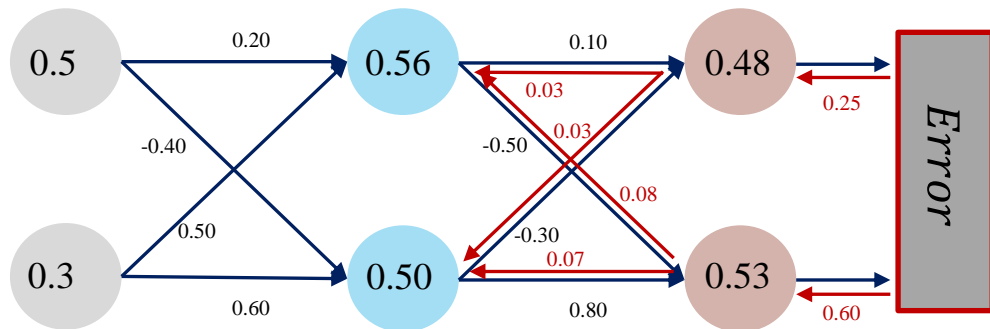


$$\delta_5 = \frac{\partial Error}{\partial w_5} = \frac{\partial Error}{\partial o_1} * \frac{\partial o_1}{\partial In_{o_1}} * \frac{\partial In_{o_1}}{\partial w_5} = 0.25 * 0.48 * (1 - 0.48) * 0.56 = 0.03$$

$$\delta_7 = \frac{\partial Error}{\partial w_7} = \frac{\partial Error}{\partial o_1} * \frac{\partial o_1}{\partial In_{o_1}} * \frac{\partial In_{o_1}}{\partial w_7} = 0.25 * 0.48 * (1 - 0.48) * 0.50 = 0.03$$

误差反向传播：前馈神经网络

梯度计算

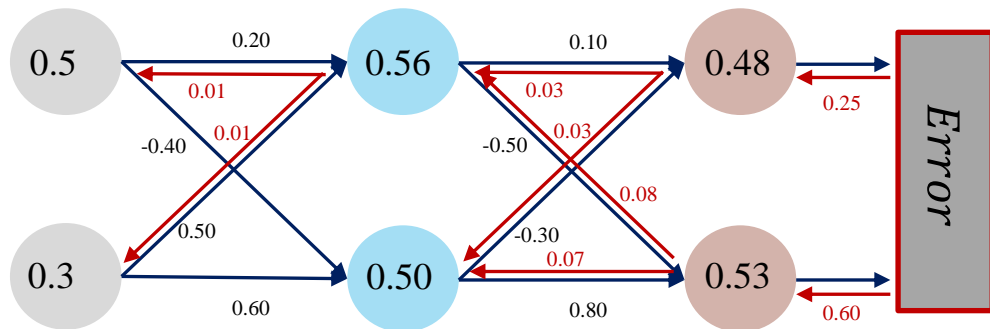


$$\delta_6 = \frac{\partial Error}{\partial w_6} = \frac{\partial Error}{\partial o_2} * \frac{\partial o_2}{\partial In_{o_2}} * \frac{\partial In_{o_2}}{\partial w_6} = 0.60 * 0.53 * (1 - 0.53) * 0.56 = 0.08$$

$$\delta_8 = \frac{\partial Error}{\partial w_8} = \frac{\partial Error}{\partial o_2} * \frac{\partial o_2}{\partial In_{o_2}} * \frac{\partial In_{o_2}}{\partial w_8} = 0.60 * 0.53 * (1 - 0.53) * 0.50 = 0.07$$

误差反向传播：前馈神经网络

梯度计算

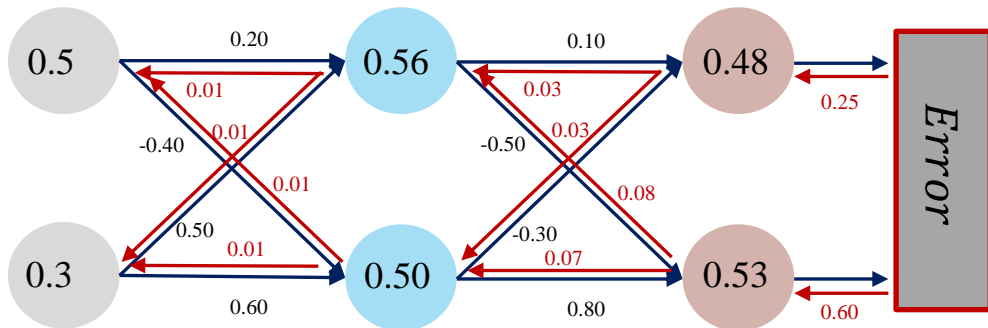


$$\delta_1 = \frac{\partial \text{Error}}{\partial w_1} = (\delta_5 + \delta_6) * \frac{\partial h_1}{\partial \ln_{h_1}} * \frac{\partial \ln_{h_1}}{\partial w_1} = (0.03 + 0.08) * 0.56 * (1 - 0.56) * 0.50 = 0.01$$

$$\delta_3 = \frac{\partial \text{Error}}{\partial w_3} = (\delta_5 + \delta_6) * \frac{\partial h_1}{\partial \ln_{h_1}} * \frac{\partial \ln_{h_1}}{\partial w_3} = (0.03 + 0.08) * 0.56 * (1 - 0.56) * 0.30 = 0.01$$

误差反向传播：前馈神经网络

梯度计算



$$\delta_2 = \frac{\partial Error}{\partial w_2} = (\delta_7 + \delta_8) * \frac{\partial h_2}{\partial \ln_{h_2}} * \frac{\partial \ln_{h_2}}{\partial w_2} = (0.03 + 0.07) * 0.50 * (1 - 0.50) * 0.5 = 0.01$$

$$\delta_4 = \frac{\partial Error}{\partial w_4} = (\delta_7 + \delta_8) * \frac{\partial h_2}{\partial \ln_{h_2}} * \frac{\partial \ln_{h_2}}{\partial w_4} = (0.03 + 0.07) * 0.50 * (1 - 0.50) * 0.3 = 0.01$$

误差反向传播：前馈神经网络

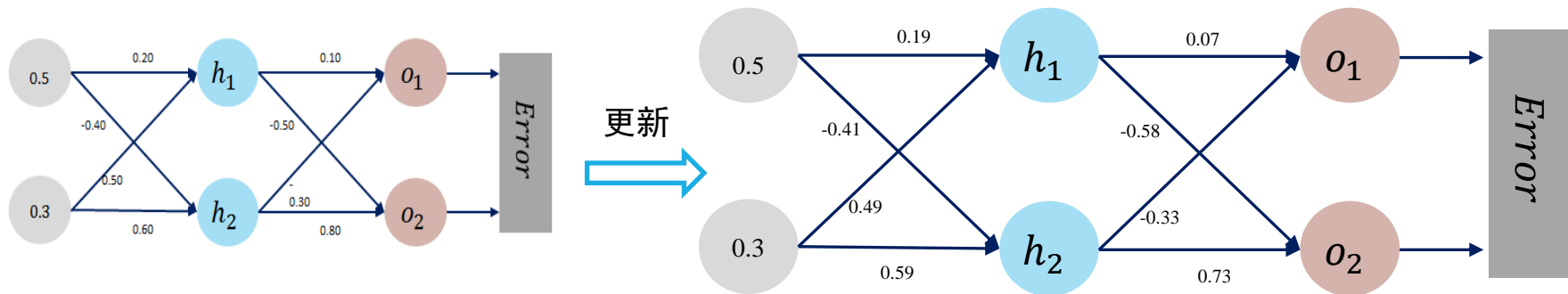
参数更新

$$w'_i = w_i - \eta * \delta_i$$


原有
参数

步长

传递
误差

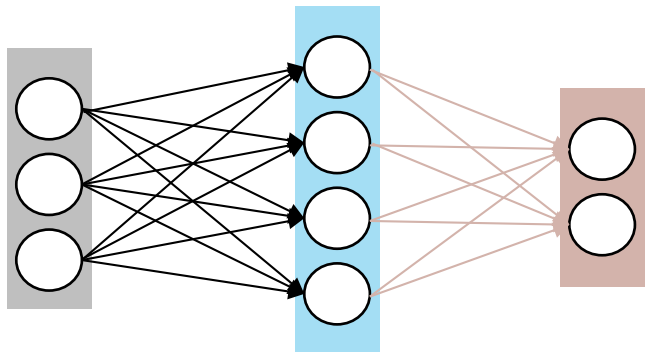


提纲

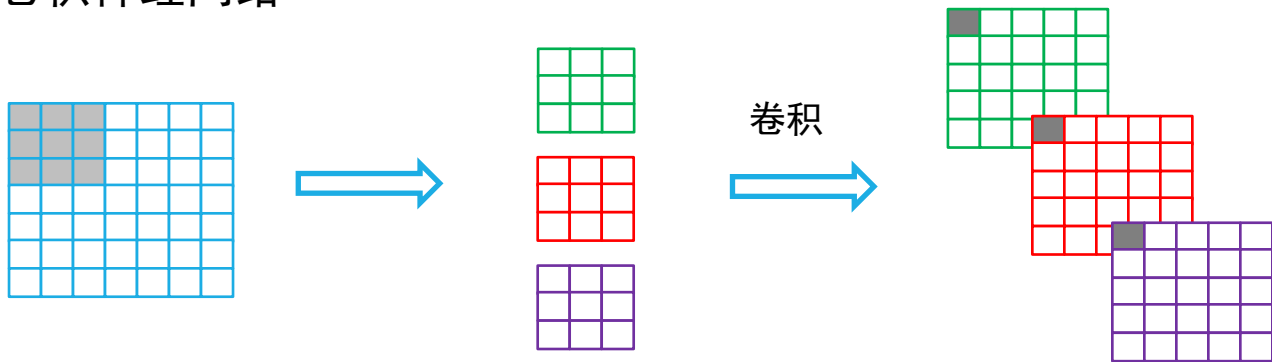
- 1、前向神经网络
 - 2、卷积神经网络
 - 3、自然语言理解与视觉分析
- 

从前馈神经网络到卷积神经网络(convolution neural network, CNN)

三层前馈神经网络



卷积神经网络

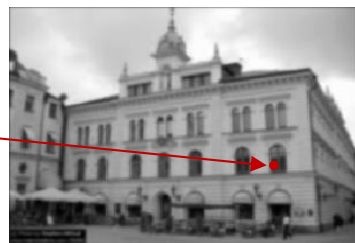


卷积神经网络：卷积操作

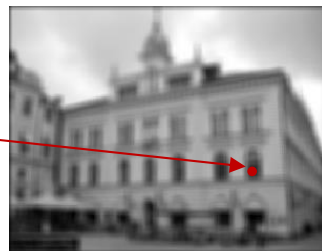
- 图像经过特定卷积矩阵滤波后，所得到的卷积结果可认为是保留了像素点所构成的特定空间分布模式
- 如下给出了两个7*7高斯卷积核的卷积结果



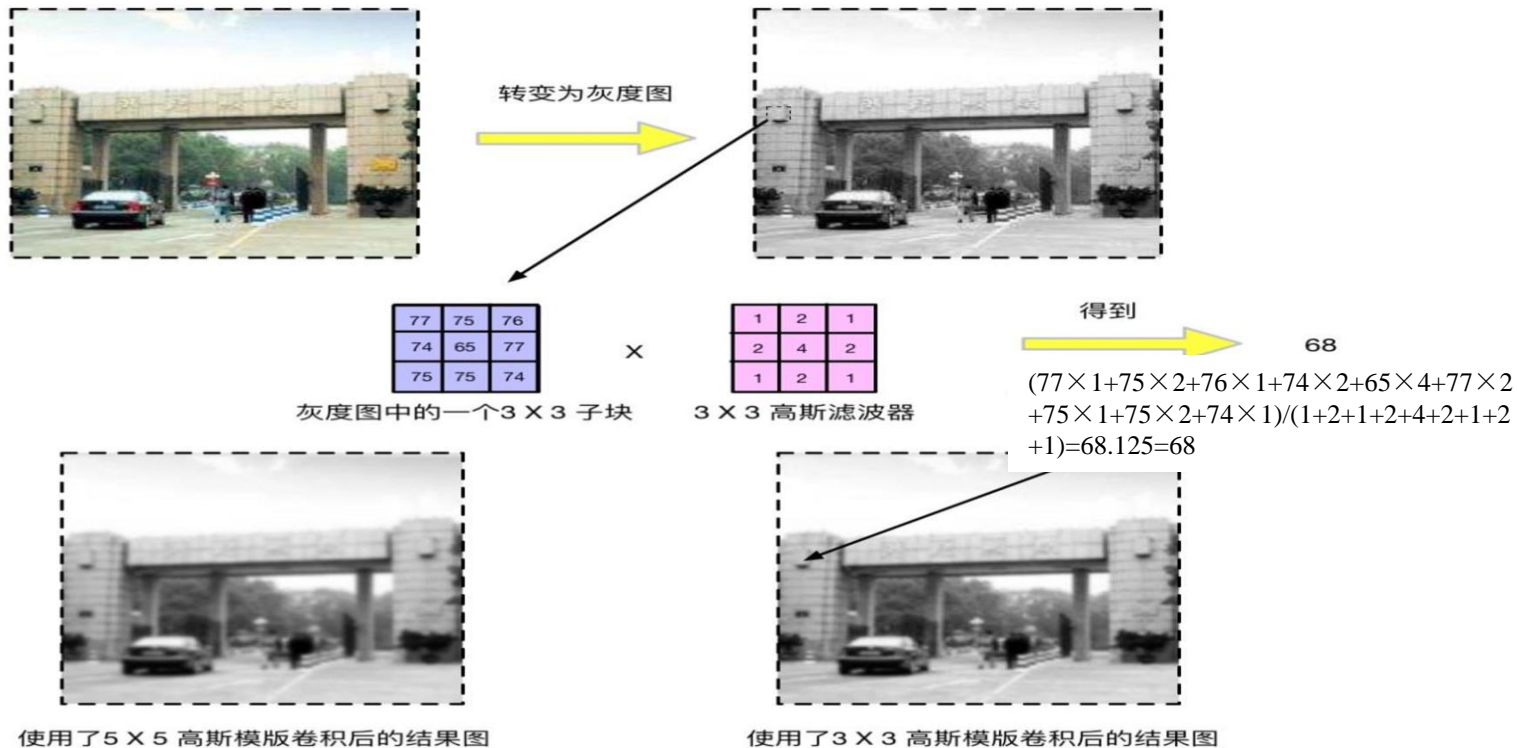
0.0000	0.0002	0.0011	0.0018	0.0011	0.0002	0.0000
0.0002	0.0029	0.0131	0.0216	0.0131	0.0029	0.0002
0.0011	0.0131	0.0586	0.0966	0.0586	0.0131	0.0011
0.0018	0.0216	0.0966	0.1592	0.0966	0.0216	0.0018
0.0011	0.0131	0.0586	0.0966	0.0586	0.0131	0.0011
0.0002	0.0029	0.0131	0.0216	0.0131	0.0029	0.0002
0.0000	0.0002	0.0011	0.0018	0.0011	0.0002	0.0000



0.0194	0.0199	0.0202	0.0203	0.0202	0.0199	0.0194
0.0199	0.0204	0.0207	0.0208	0.0207	0.0204	0.0199
0.0202	0.0207	0.0210	0.0211	0.0210	0.0207	0.0202
0.0203	0.0208	0.0211	0.0212	0.0211	0.0208	0.0203
0.0202	0.0207	0.0210	0.0211	0.0210	0.0207	0.0202
0.0199	0.0204	0.0207	0.0208	0.0207	0.0204	0.0199
0.0194	0.0199	0.0202	0.0203	0.0202	0.0199	0.0194

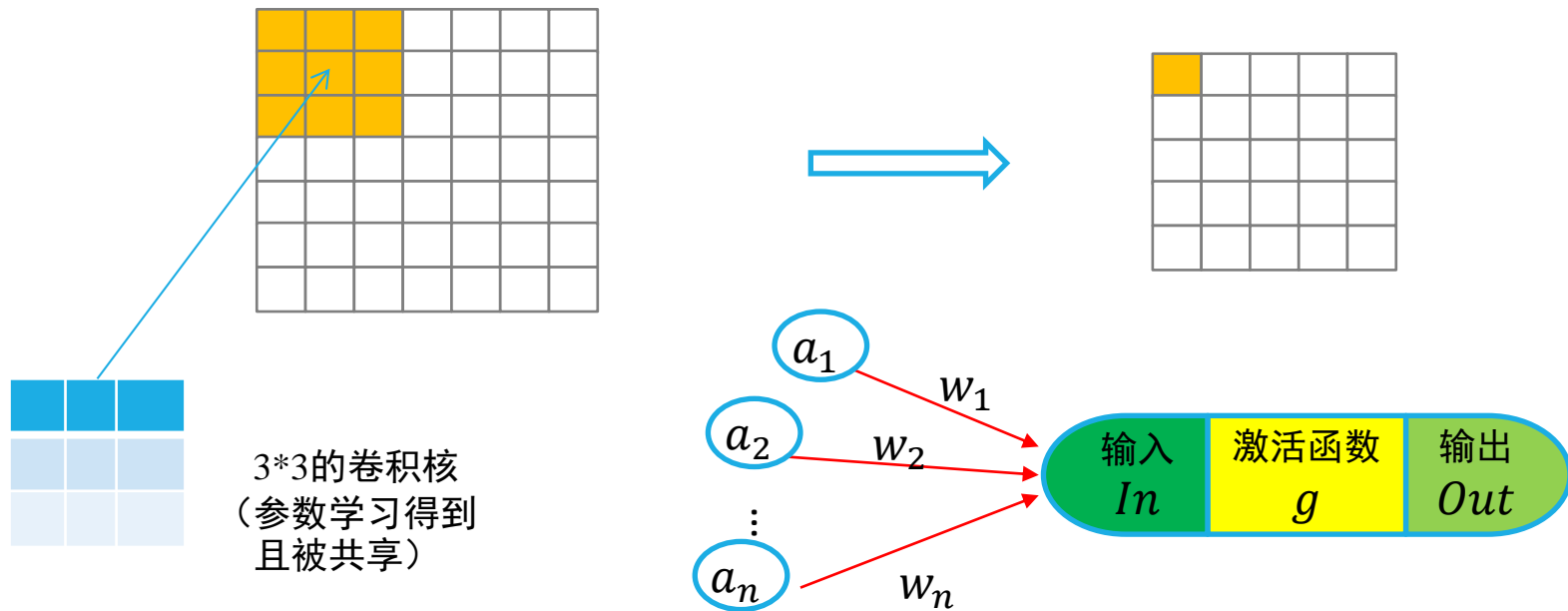


卷积神经网络：卷积操作示意



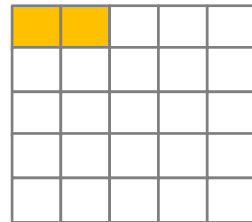
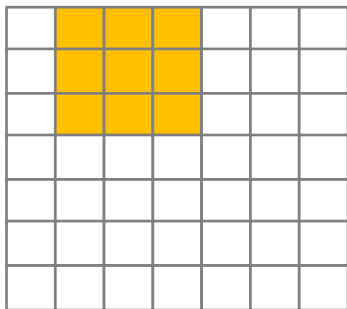
卷积神经网络：卷积操作

7*7大小的图像，通过3*3大小卷积矩阵以1的步长进行卷积操作，可得到5*5大小的卷积结果



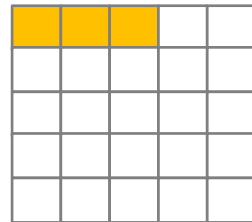
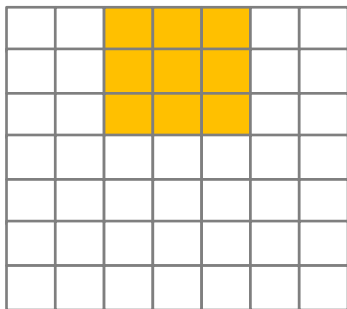
卷积神经网络：卷积操作

7*7大小的图像，通过3*3大小卷积矩阵以1的步长进行卷积操作，可得到5*5大小的卷积结果



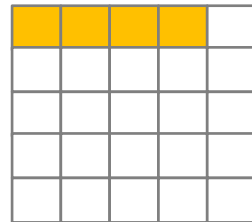
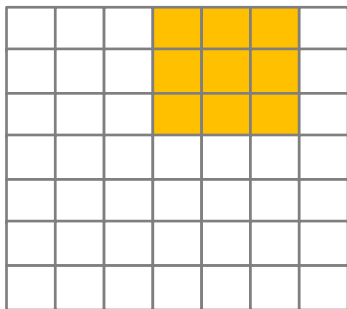
卷积神经网络：卷积操作

7*7大小的图像，通过3*3大小卷积矩阵以1的步长进行卷积操作，可得到5*5大小的卷积结果



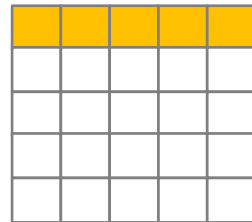
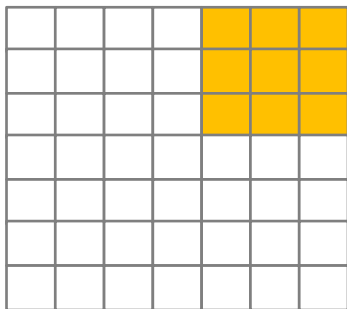
卷积神经网络：卷积操作

7*7大小的图像，通过3*3大小卷积矩阵以1的步长进行卷积操作，可得到5*5大小的卷积结果



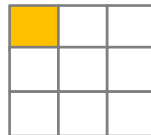
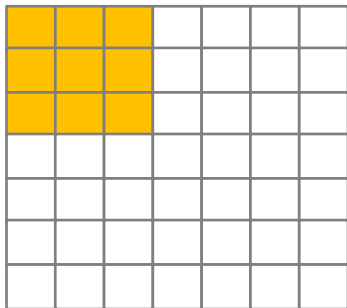
卷积神经网络：卷积操作

7*7大小的图像，通过3*3大小卷积矩阵以1的步长进行卷积操作，可得到5*5大小的卷积结果



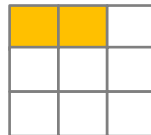
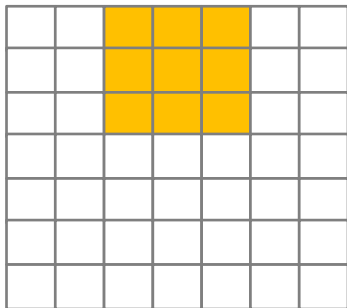
卷积神经网络：卷积操作

如果步长改为2



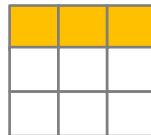
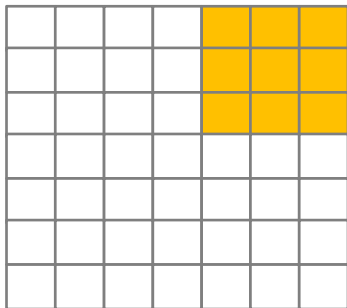
卷积神经网络：卷积操作

如果步长改为2



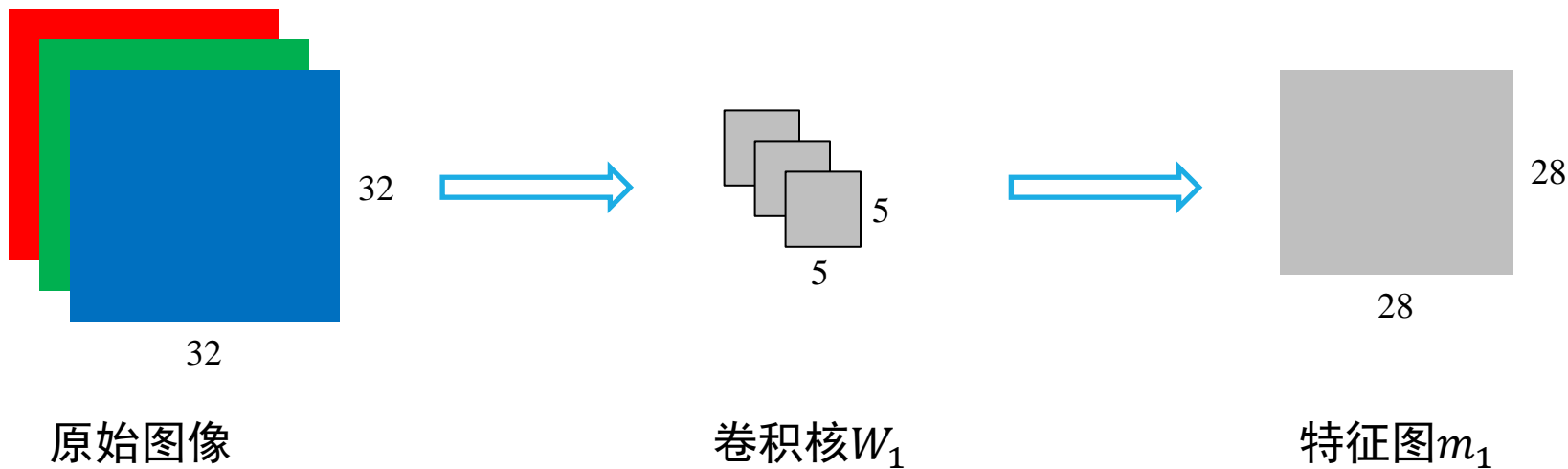
卷积神经网络：卷积操作

如果步长改为2



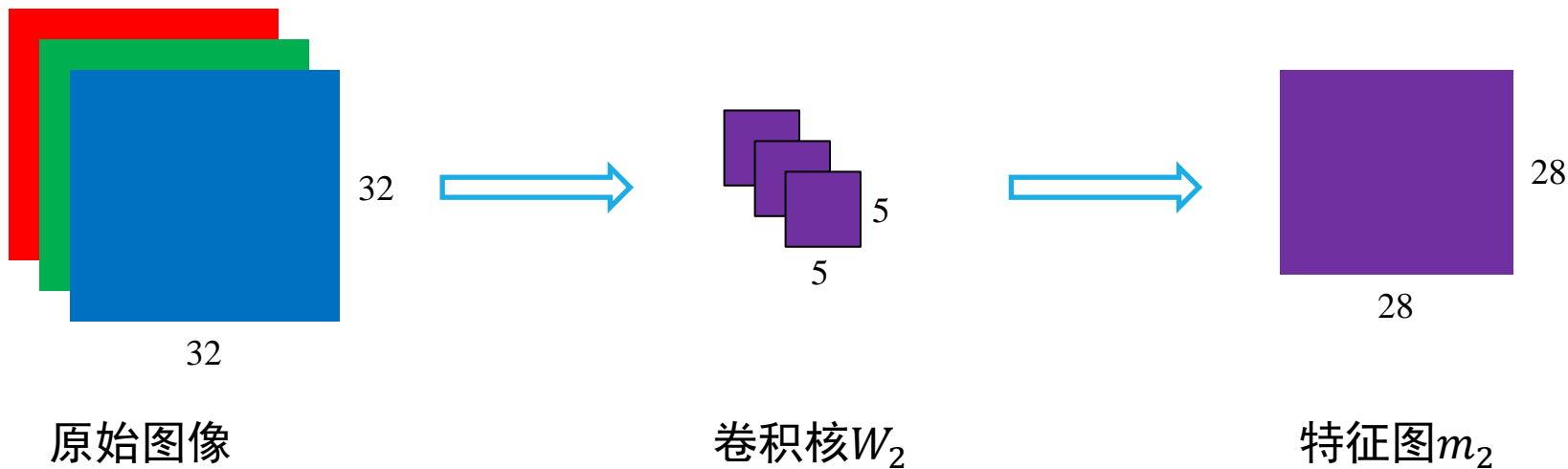
卷积神经网络：卷积操作

有一张 $32*32*3$ (RGB)的图像，使用 $5*5*3$ 的卷积核 W_1 ，步长为1对其进行卷积操作。卷积核 W_1 在原始图像上从左到右、从上到下进行计算，改变 $5*5$ 子块区域中的中心像素点值，得到 $28*28$ 的特征图 m_1



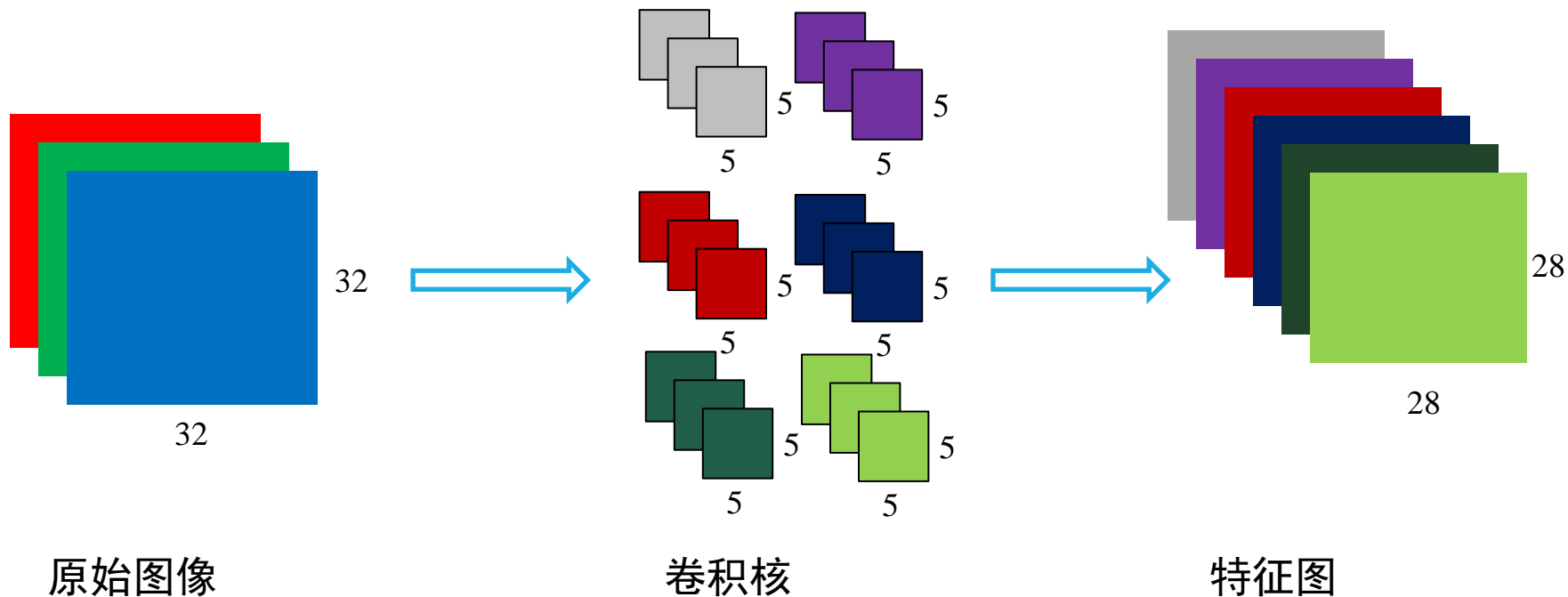
卷积神经网络：卷积操作

使用另一个 $5*5*3$ 的卷积核 W_2 与原始图像做卷积操作，得到特征图 m_2



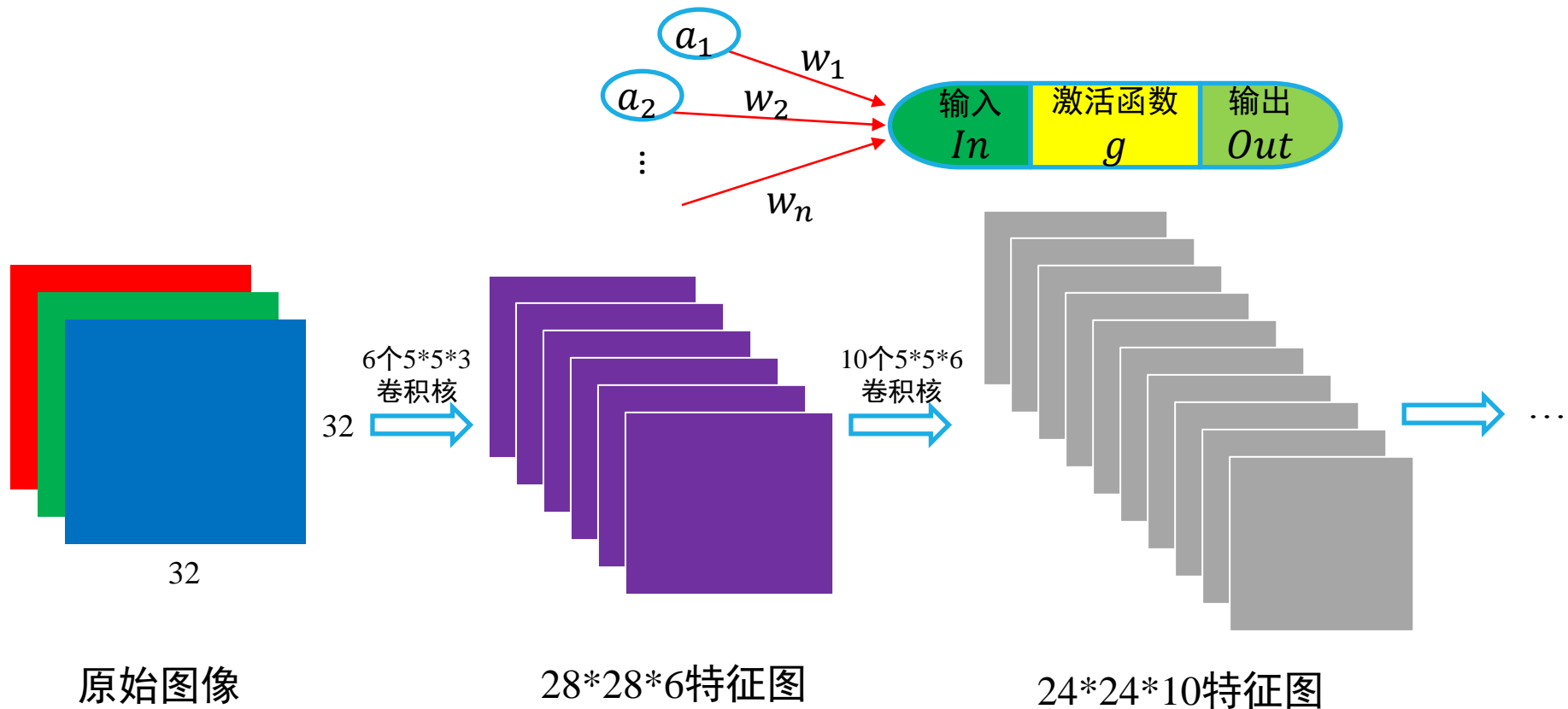
卷积神经网络：卷积操作

- 使用6个 $5 \times 5 \times 3$ 的卷积核与原始图像做卷积操作，则得到6个 28×28 的特征图
- 注意：6个 $5 \times 5 \times 3$ 的卷积核均是数据驱动学习得到，其刻画了不同的视觉模式



卷积神经网络：卷积+激活函数(非线性映射)

在对原始图像做卷积操作后，可使用ReLU激活函数对卷积后结果进行处理



卷积神经网络：池化(pooling)操作

- 对输入的特征图进行下采样，以获得最主要信息
- 常用的池化操作有：
 - 最大池化
 - 平均池化

卷积神经网络：最大池化操作

在输入特征图中每一个区域寻找最大值。

1	5	4	3
2	6	4	0
3	1	7	9
2	8	6	5

对2*2大小区域，按照步长为2进行最大化池化操作



6	4
8	9

卷积神经网络：平均池化操作

对输入特征图的每一个区域的值求平均值（取整）

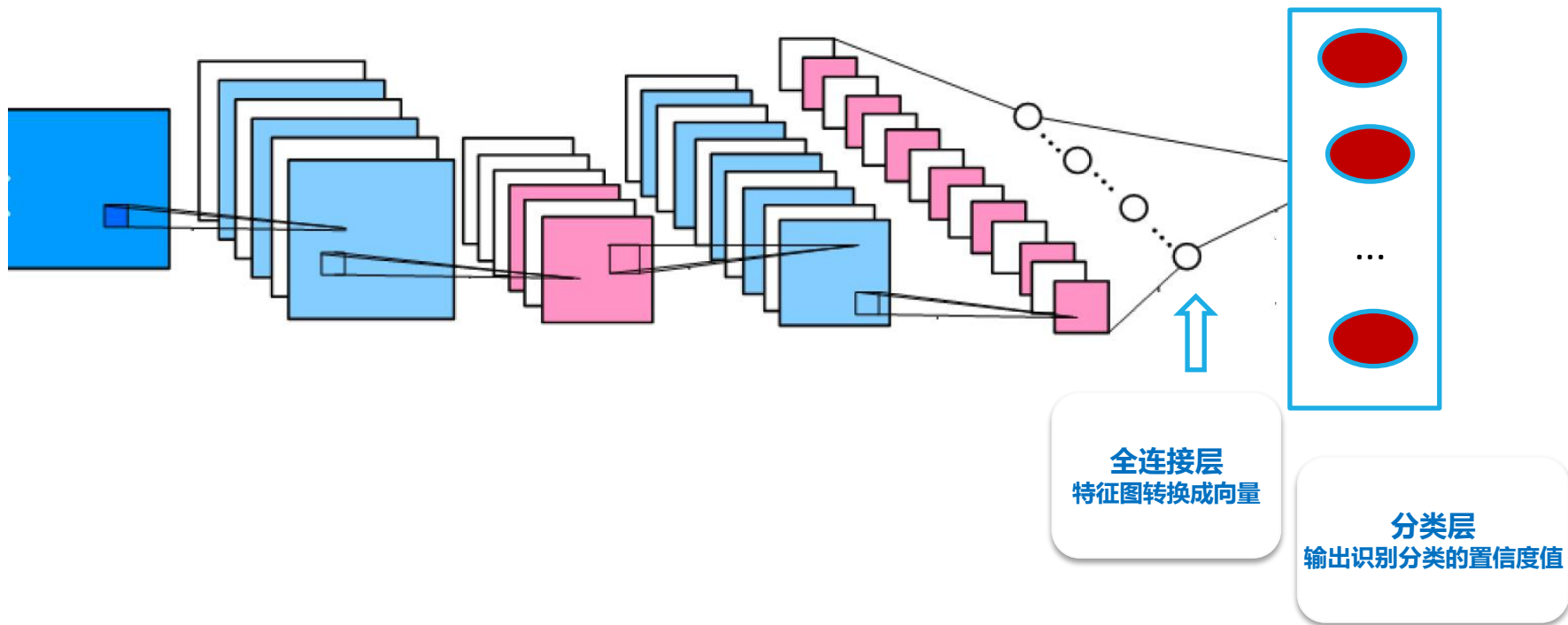
1	5	4	3
2	6	4	0
3	1	7	9
2	8	6	5

对2*2大小区域，按照步长为2进行平均池化操作



3	2
3	6

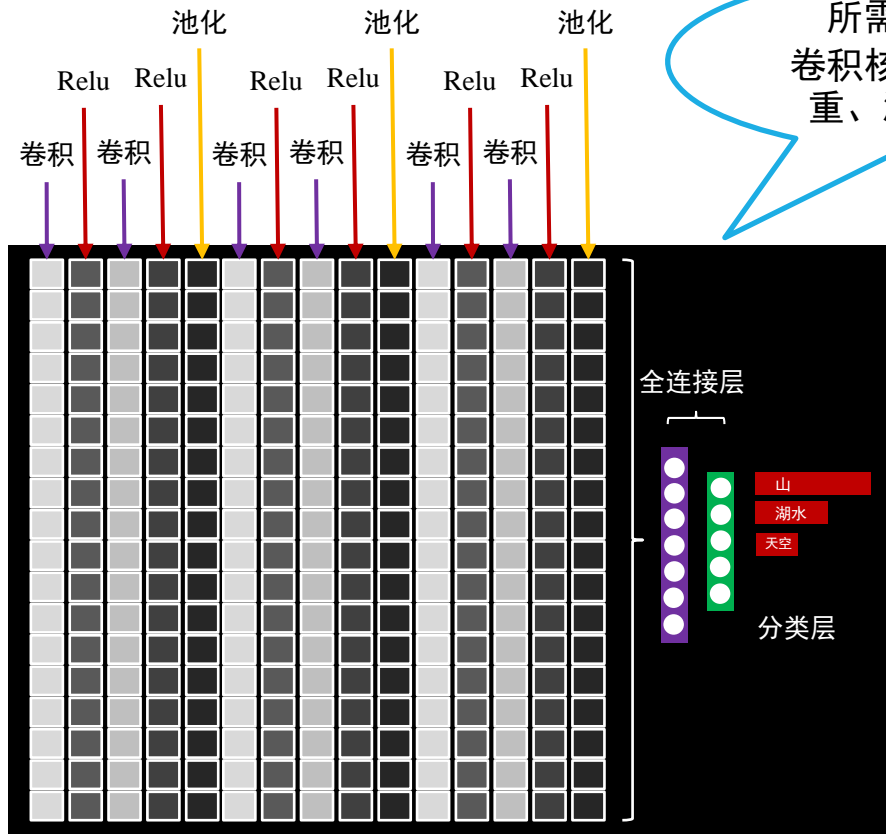
卷积神经网络：全连接层与分类层



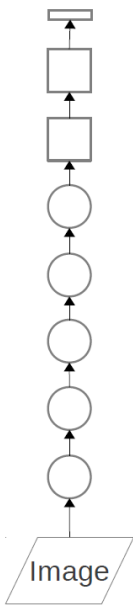
卷积神经网络基本架构



输入



卷积神经网络的参数



- Trained with stochastic gradient descent on two NVIDIA GPUs for about a week
- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- **Final feature layer:** 4096-dimensional

体现了数据、模型和算力的结合



Convolutional layer: convolves its input with a bank of 3D filters, then applies point-wise non-linearity



Fully-connected layer: applies linear filters to its input, then applies point-wise non-linearity

Alex Krizhevsky, et.al., ImageNet Classification with Deep Convolutional Neural Networks

提纲

- 1、前向神经网络
- 2、卷积神经网络
- 3、自然语言理解与视觉分析

深度学习的应用：学习单词的表达-----词向量(Word2Vec)

- 在基于规则和统计的自然语言传统方法中，将单词视为独立符号
- 在向量空间中，一个单词按照其在文档中出现的有无，被表示为如下向量（按照字典序）：

...
I own a new **ford Explorer**, I really love it!
and besides the **power** I ... spending the money for
it! The **Jeep** was great but I just love the **Explorer**! I have a
2WD and I got through the blizzard of 93 just fine! I **drove**
about 400 miles in the worst part of storm and it never faulted!
...

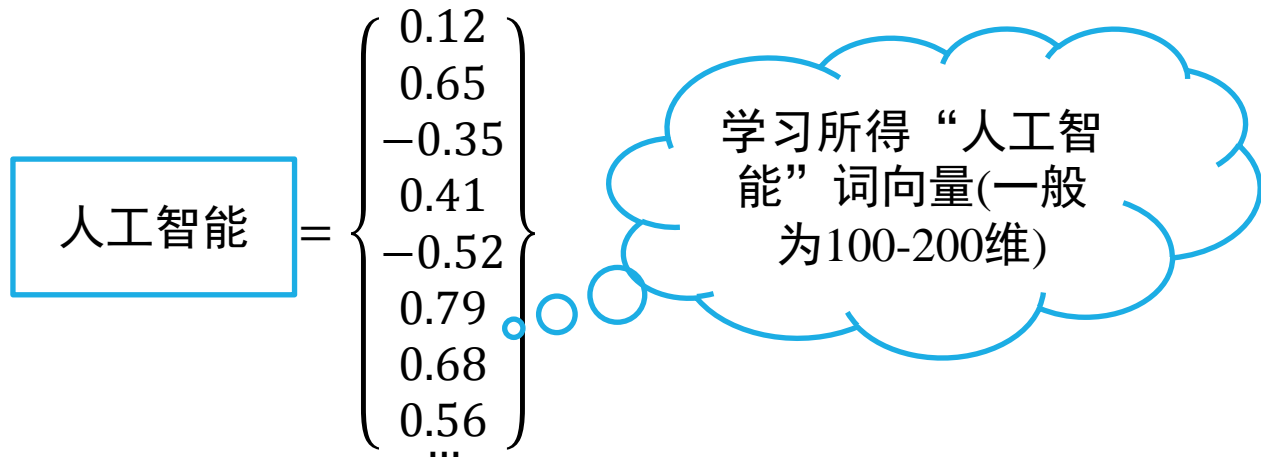
$[0, 0, 0, 1, 0, 0, \dots, 0, 0, 0, 0]$

上述表示方法称为**One-hot**向量。

- 缺点：
 - 维数灾难的困扰
 - 无法刻画词与词之间的相似性：任意两个词之间都是孤立的

深度学习的应用：学习单词的表达-----词向量

- One-hot 表达与单词的分布无关
- 通过深度学习方法，将单词表征为K维实数值向量(distribution representation)。这样，把对文本内容分析简化为 K 维向量空间中的向量运算，而向量空间上的相似度可以用来表示文本语义上的相似。用深度学习算法生成每个单词的向量表达所有单词的向量表达组成了一个“词向量空间”
- 单词表达为词向量后，很多 NLP 相关工作（如聚类、同义词计算、主题挖掘等)可以顺利开展。



深度学习的应用：学习单词的表达-----词向量

深度学习是机器学习中一种基于对数据进行表征学习的方法。观测值（例如一幅图像）可以使用多种方式来表示，如每个像素强度值的向量，或者更抽象地表示成一系列边、特定形状的区域等。而使用某些特定的表示方法更容易从实例中学习任务（例如，人脸识别或面部表情识别）。深度学习的好处是用非监督式或半监督式的特征学习和分层特征提取高效算法来替代手工获取特征。

深度学习 [0, 0, 0, 0 ... 0, 1, 0, ... 0, 0, 0]
机器学习 [0, 1, 0, 0 ... 0, 0, 0, ... 0, 0, 0]
 一种 [0, 0, 0, 0 ... 0, 0, 0, ... 1, 0, 0]
 基于 [0, 0, 1, 0 ... 0, 0, 0, ... 0, 0, 0]
 数据 [0, 0, 0, 0 ... 0, 0, 0, ... 0, 1, 0]
 表征学习 [1, 0, 0, 0 ... 0, 0, 0, ... 0, 0, 0]
 方法 [0, 0, 0, 0 ... 1, 0, 0, ... 0, 0, 0]
 ⋮
手工 [0, 0, 0, 1 ... 0, 0, 0, ... 0, 0, 0]
获取 [0, 0, 0, 0 ... 0, 0, 0, ... 0, 0, 1]
特征 [0, 0, 0, 0 ... 0, 0, 1, ... 0, 0, 0]

One-Hot
单词之间的关联丢失

深度学习是机器学习中一种基于对数据进行表征学习的方法。观测值（例如一幅图像）可以使用多种方式来表示，如每个像素强度值的向量，或者更抽象地表示成一系列边、特定形状的区域等。而使用某些特定的表示方法更容易从实例中学习任务（例如，人脸识别或面部表情识别）。深度学习的好处是用非监督式或半监督式的特征学习和分层特征提取高效算法来替代手工获取特征。

深度学习 [0.23, -0.18, 0.45... .. 0.68, -0.33]
机器学习 [0.11, -0.25, 0.78... .. 0.22, 0.61]
 一种 [0.08, 0.09, 0.52... .. -0.37, -0.42]
 基于 [-0.16, 0.29, 0.80... .. -0.21, 0.20]
 数据 [-0.67, -0.03, 0.29... .. 0.46, 0.87]
 表征学习 [0.26, 0.55, 0.70... .. 0.62, -0.40]
 方法 [0.59, 0.92, 0.33... .. 0.53, 0.90]
 ⋮
手工 [0.15, -0.66, -0.72... .. 0.48, 0.59]
获取 [0.19, 0.23, 0.81, 0.43, -0.81]
特征 [0.76, 0.49, -0.38... .. 0.65, 0.07]

词向量
可基于单词形成的向量进行后续操作

词向量模型的训练

通过某个单词 w_t 上下文单词的词向量来预测单词 w_t 的词向量：

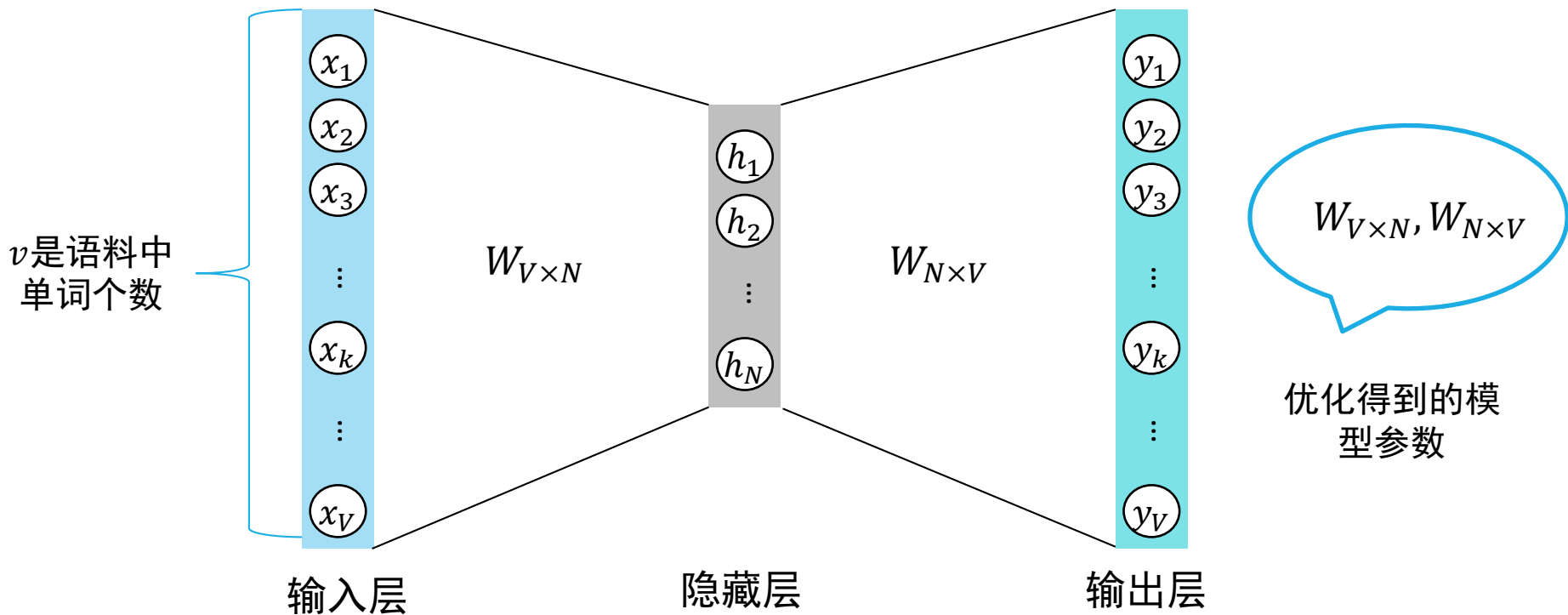
$$f(w_t, w_{t-1}, \dots, w_{t-n+2}, w_{t-n+1}) = p(w_t | context)$$

如下优化模型参数 θ ，以最大化训练数据的对数似然函数：

$$J = \max_{\theta} (\log f(w_t, w_{t-1}, \dots, w_{t-n+2}, w_{t-n+1}; \theta) + R(\theta))$$

词向量模型的基本思想

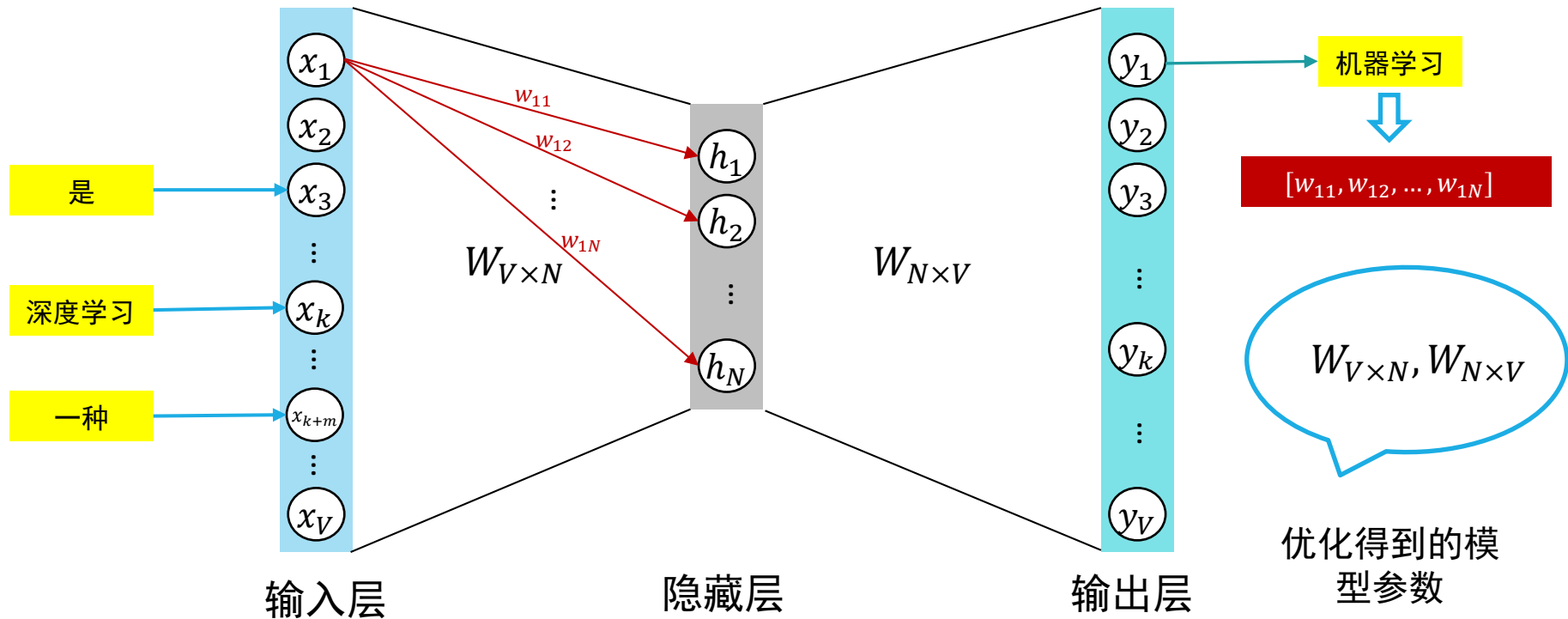
词向量模型由一层输入层，一层隐藏层，一层输出层构成：实现了每个单词 N 维向量的表达



词向量模型的基本思想

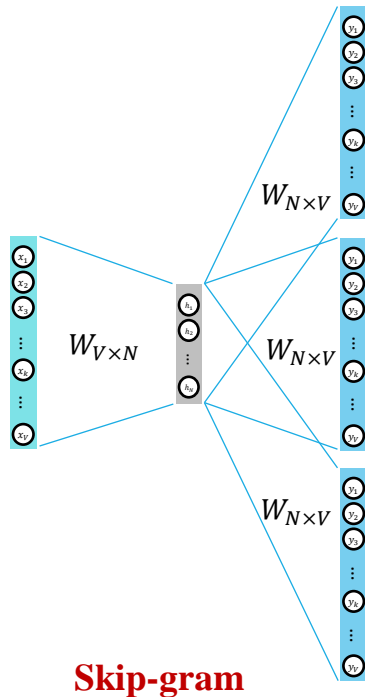
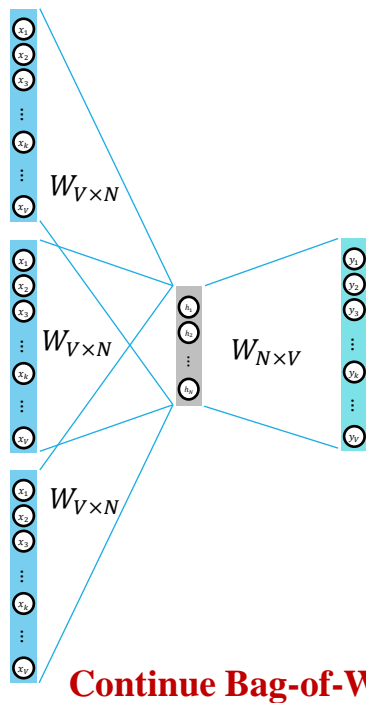
词向量模型由一层输入层，一层隐藏层，一层输出层构成：实现了每个单词 N 维向量的表达

以“深度学习 是一种 机器学习 的方法”为例



词向量模型：两种训练模式

- Continue Bag-of-Words (CBoW): 根据某个单词的上下文单词来预测该单词
- Skip-gram: 利用某个单词来分别预测该单词的上下文单词



$W_{V \times N}, W_{N \times V}$

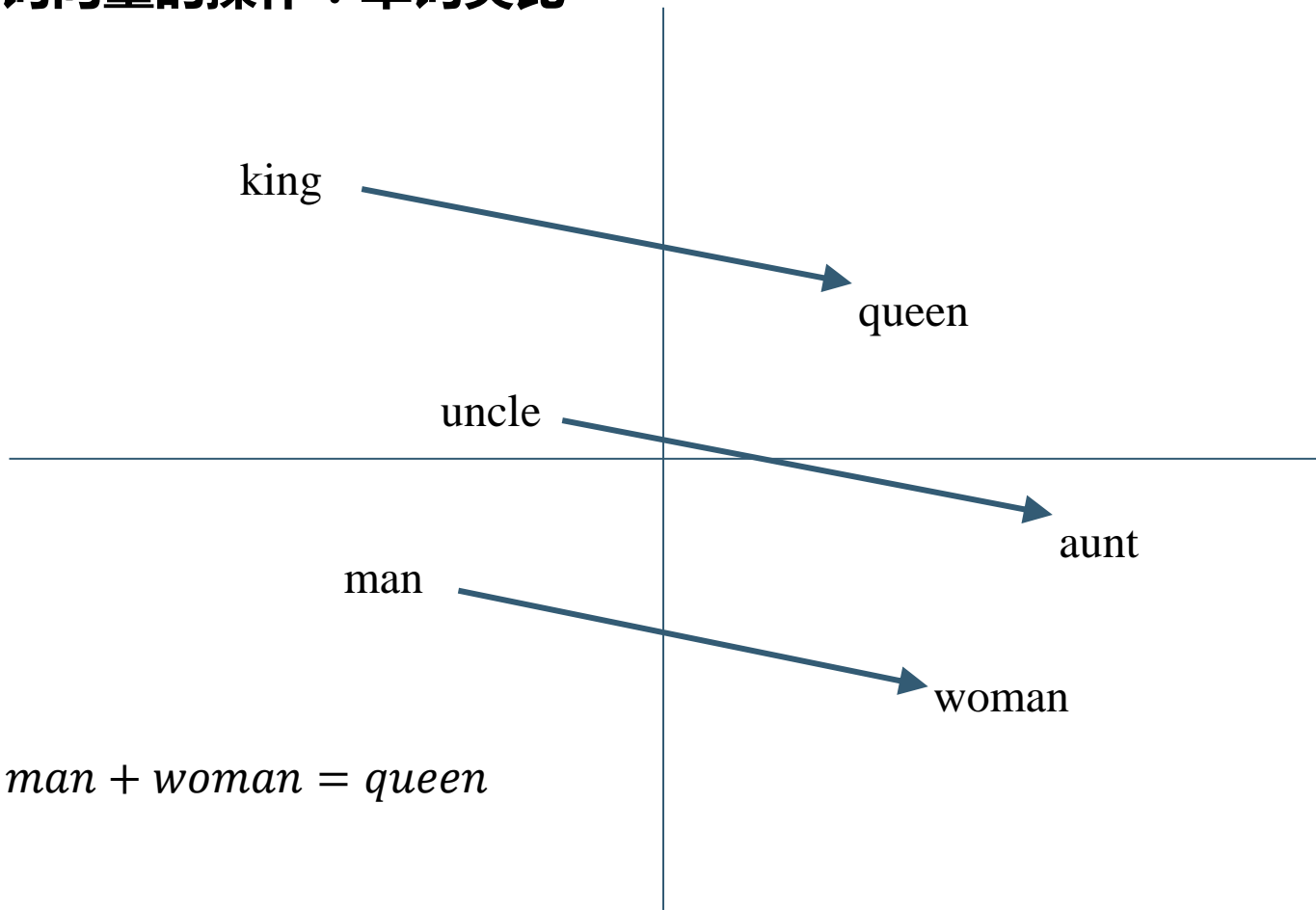
学习训练得到的
模型参数

Word2Vec的改进算法

- 对一个包含10000个单词的语料库，每个单词的词向量设为200维，则需要 $200 \times 10000 (2000000)$ 和 $10000 \times 200 (2000000)$ 异常庞大的权重矩阵
- 在如此大神经网络上进行梯度下降耗时
- 为了解决这个不足，后续出现了如下改进手段：
 - Hierarchical Softmax (引入霍夫曼树)
 - Negative Sampling

Distributed Representations of Words and Phrases and their Compositionality

基于词向量的操作：单词类比



$$\textit{king} - \textit{man} + \textit{woman} = \textit{queen}$$

卷积神经网络应用

Classification



Car

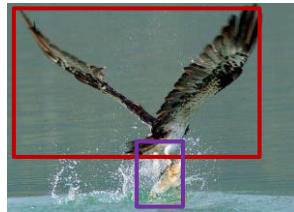
Localization



(x, y, w, h)

...

Object Detection



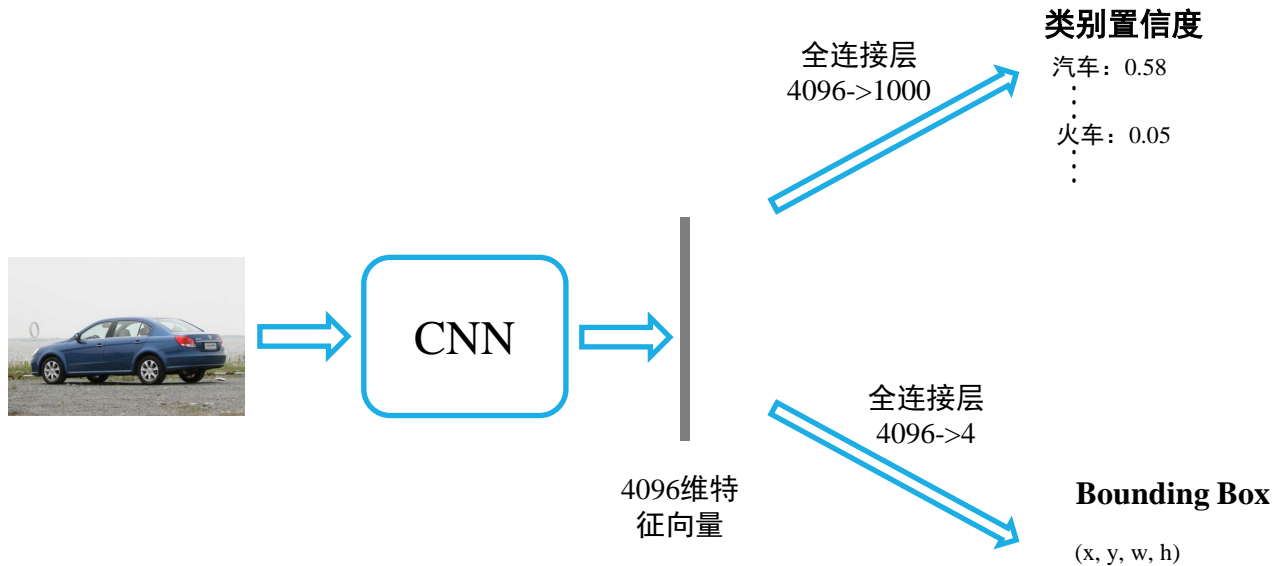
Eagle, Fish

...

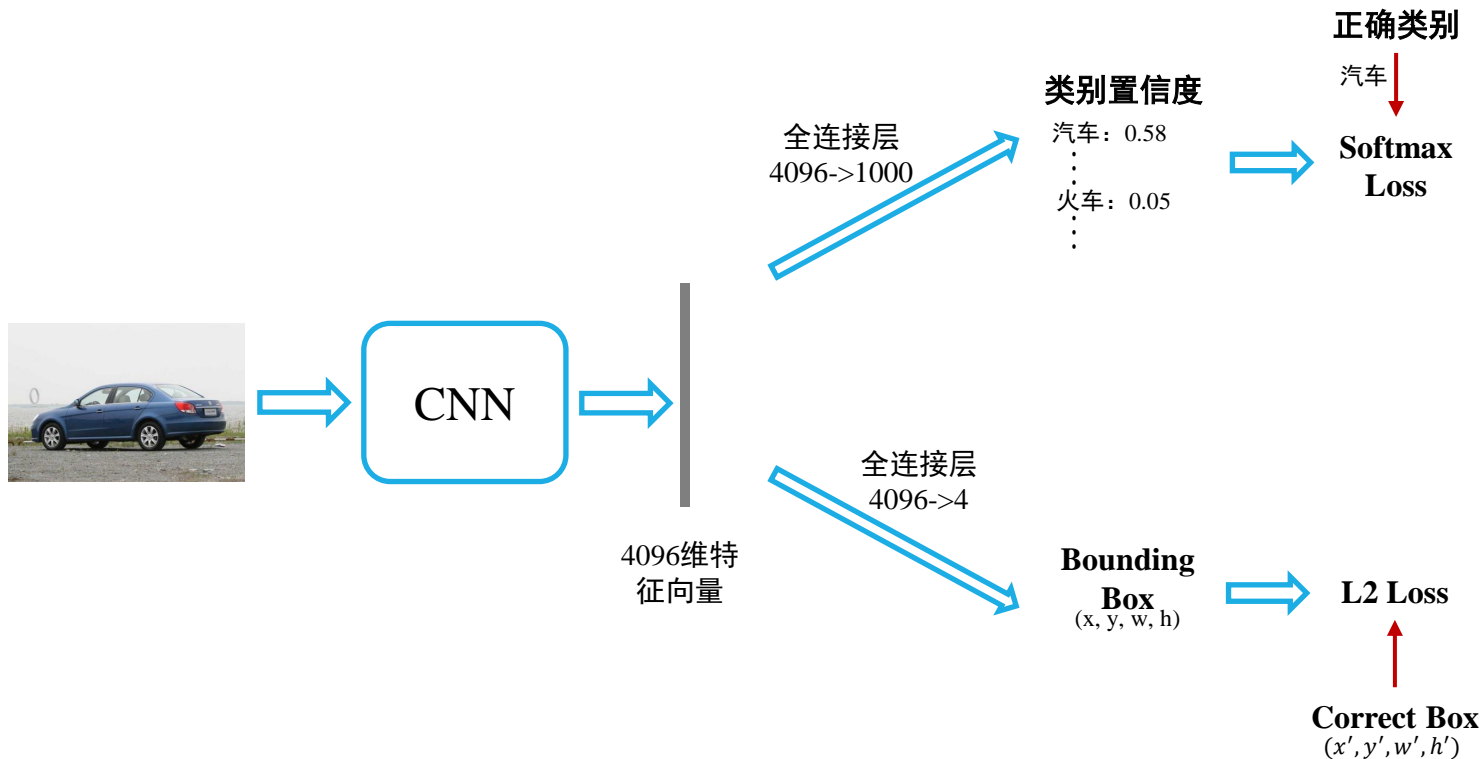
Single
Object

Multi
Objects

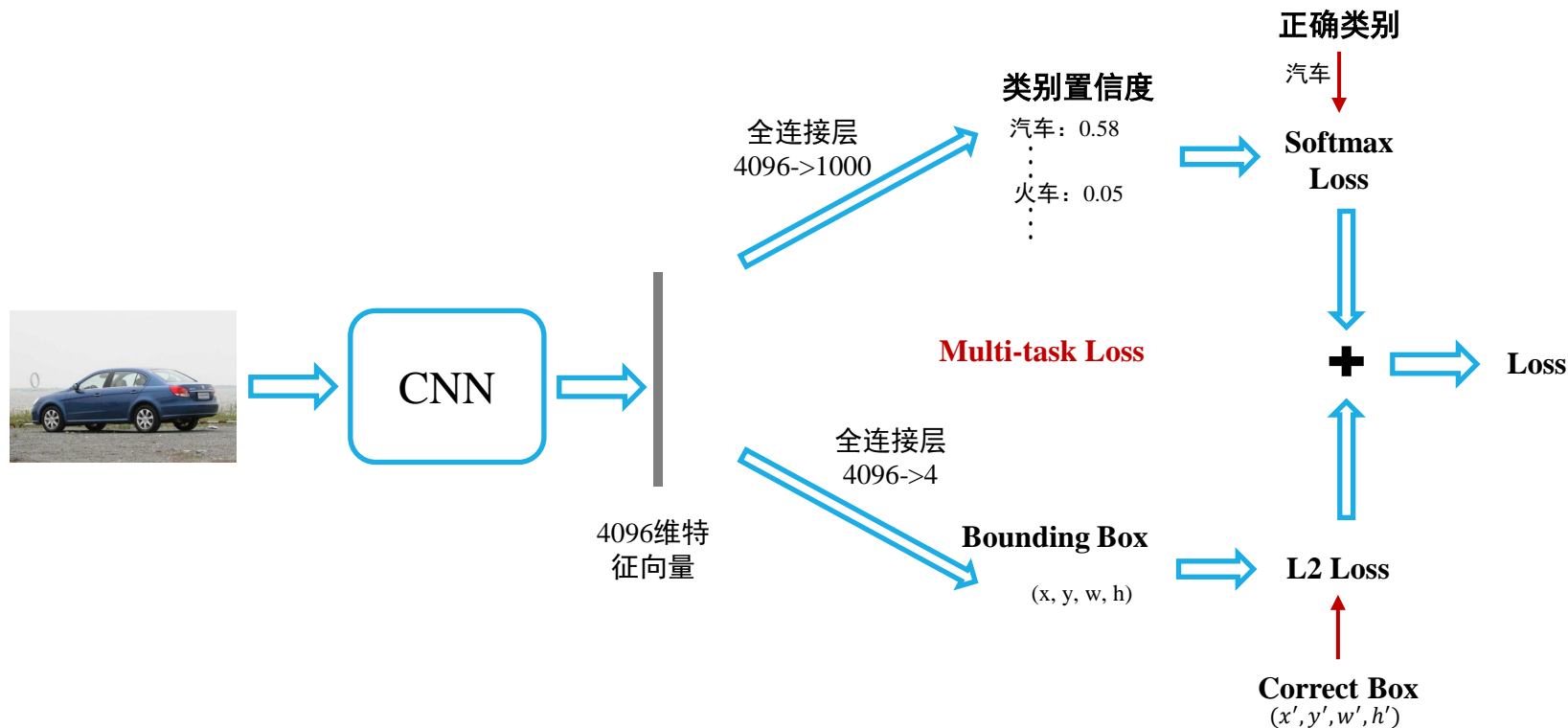
卷积神经网络应用：图像分类与定位



卷积神经网络应用：图像分类与定位



卷积神经网络应用：图像分类与定位



学习算法的改造：从浅层模型到深层模型

浅层模型	深层模型
Language model	Neural language model
Bayesian Learning	Bayesian deep learning
Turing Machine	Neural Turing Machine
Reinforcement Learning	Deep Reinforcement Learning
Generative Model	Deep Generative Model
X	Deep or Neural + X