

机器学习系统设计：深度学习

教学课程组

授课方式：直播+录播+实训平台

2020年

- 主要参考教材：吴飞，《人工智能导论：模型与算法》，高等教育出版社
- 在线实训平台：<https://mo.zju.edu.cn/classroom/class?id=5f51094fa88d77399a7957d8&type=practice>
- 在线课程(MOOC)：<https://www.icourse163.org/course/ZJU-1003377027>

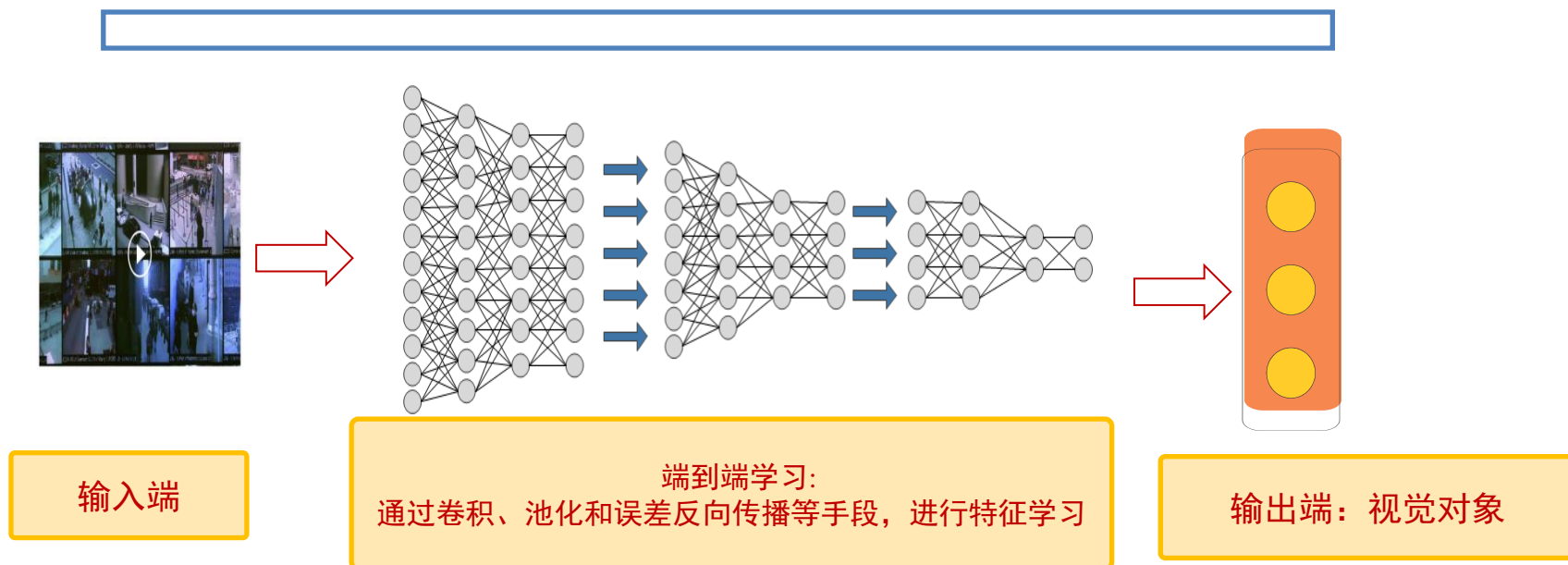
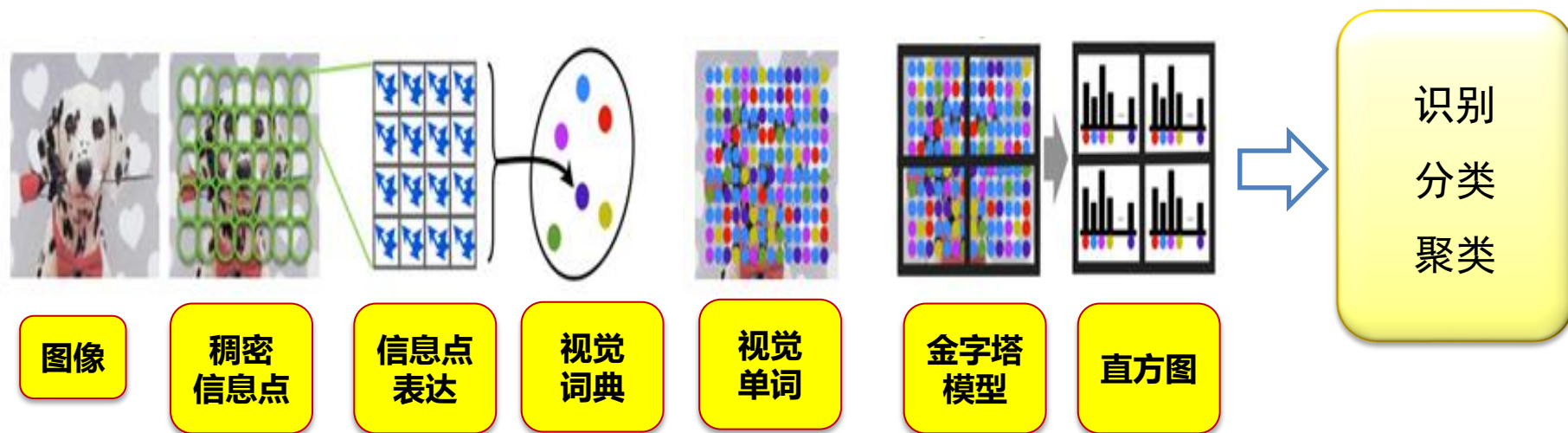
提纲

一、前馈神经网络

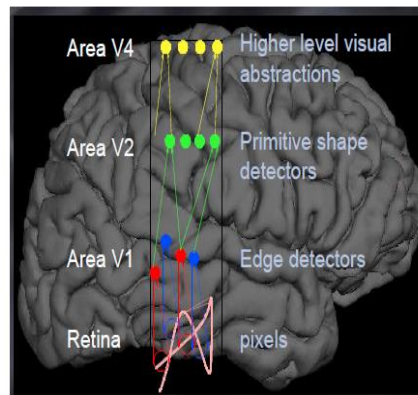
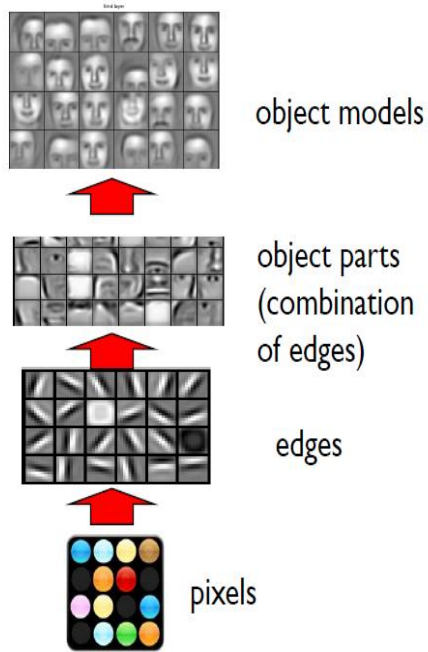
二、卷积神经网络

三、若干应用

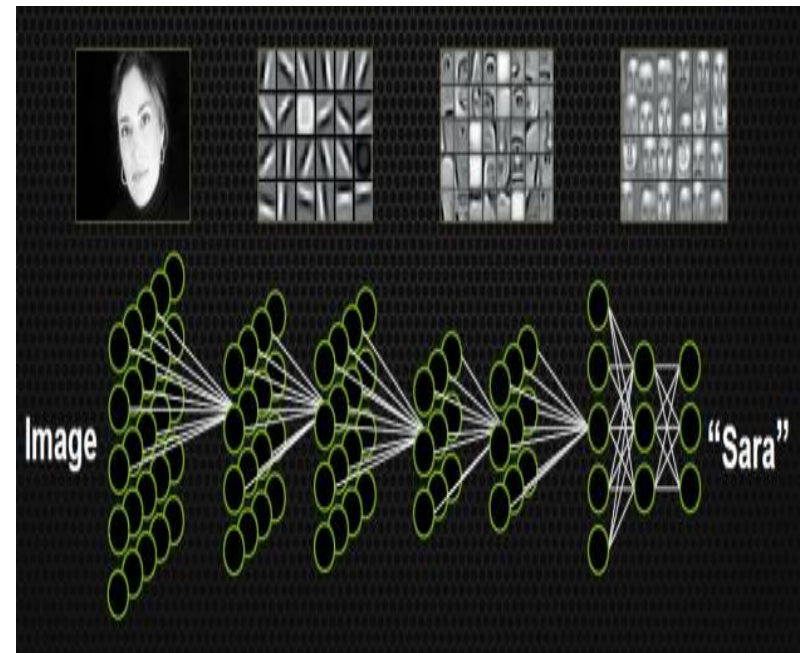
浅层学习 Versus 深度学习：从分段学习到端到端学习



深度学习：以端到端的方式逐层抽象、逐层学习



Slide credit: Andrew Ng

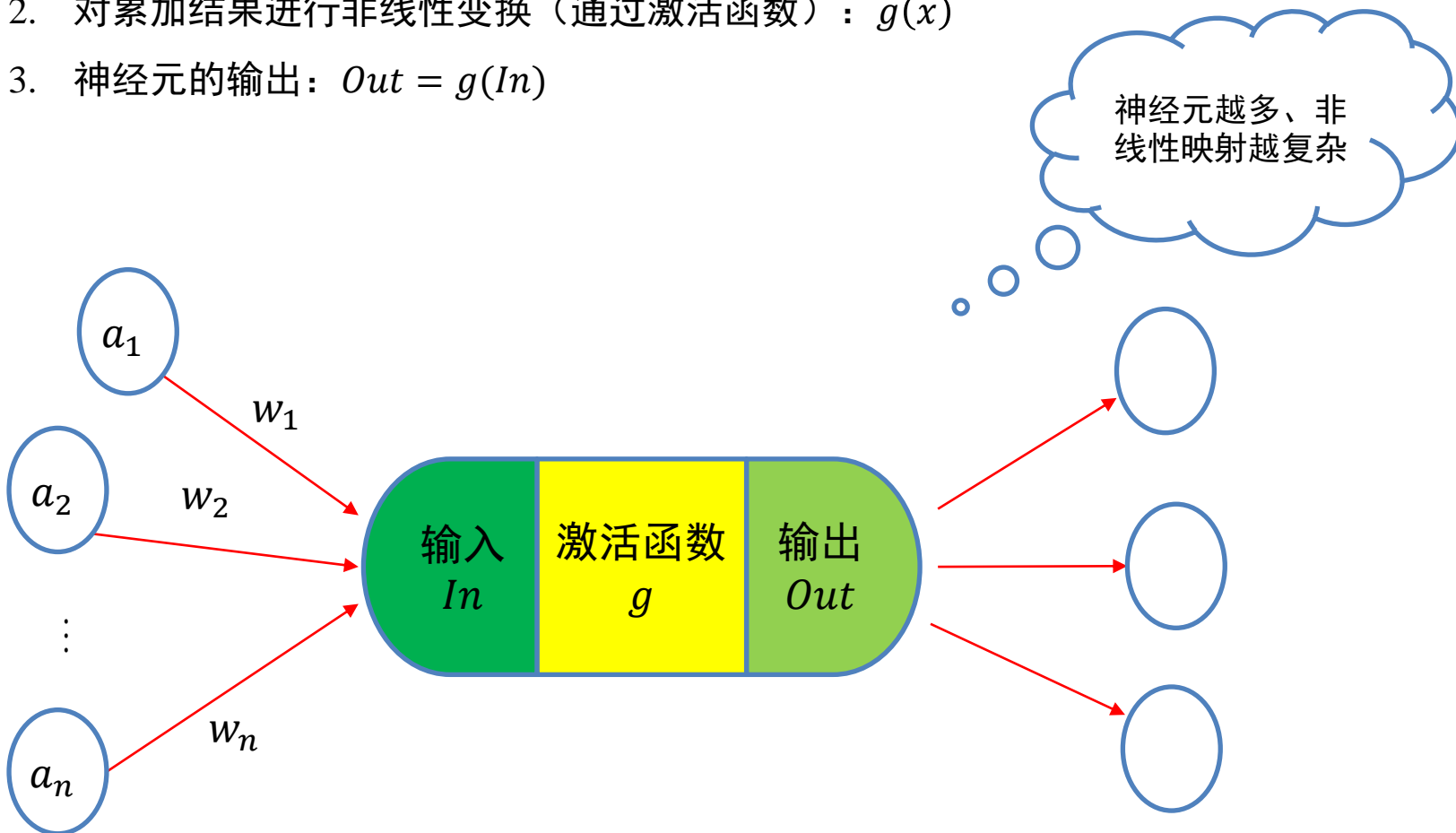


- 深度学习所得模型可视为一个复杂函数
- 非线性变换与映射的过程：像素点→语义

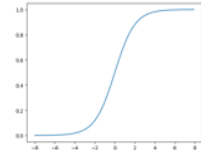
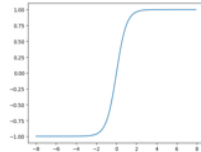
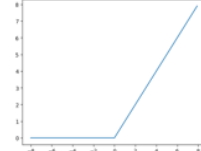
神经元

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： $Out = g(In)$



常用的激活函数：对输入信息进行非线性变换

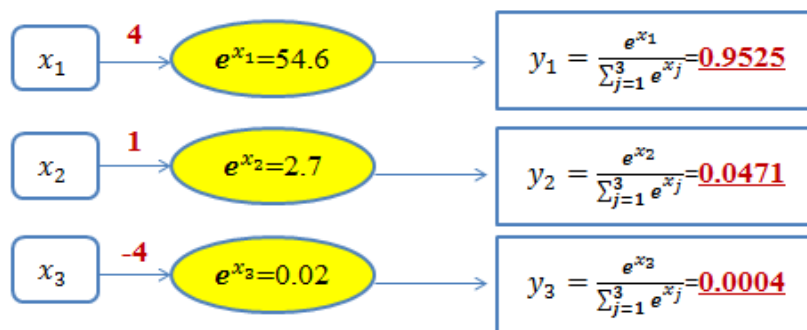
| 名称 | 函数 | 图像 | 导数 | 值域 |
|---------|--|--|---|----------------|
| Sigmoid | $f(x) = \frac{1}{1 + e^{-x}}$ |  | $f'(x) = f(x) * (1 - f(x))$ | $(0, 1)$ |
| Tanh | $f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$ |  | $f'(x) = 1 - f(x)^2$ | $(-1, 1)$ |
| ReLU | $f(x) = \begin{cases} 0, & \text{for } x \leq 0 \\ x, & \text{for } x > 0 \end{cases}$ |  | $f'(x) = \begin{cases} 0, & \text{for } x \leq 0 \\ 1, & \text{for } x > 0 \end{cases}$ | $[0, +\infty)$ |

神经网络使用非线性函数作为激活函数（activation function），通过对多个非线性函数进行组合，来实现对输入信息的非线性变换

常用的激活函数：softmax函数

Softmax函数一般用于多分类问题中，其将输入数据 x_i 映射到第 i 个类别的概率 y_i 如下计算：

$$y_i = \text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^k e^{x_j}}$$

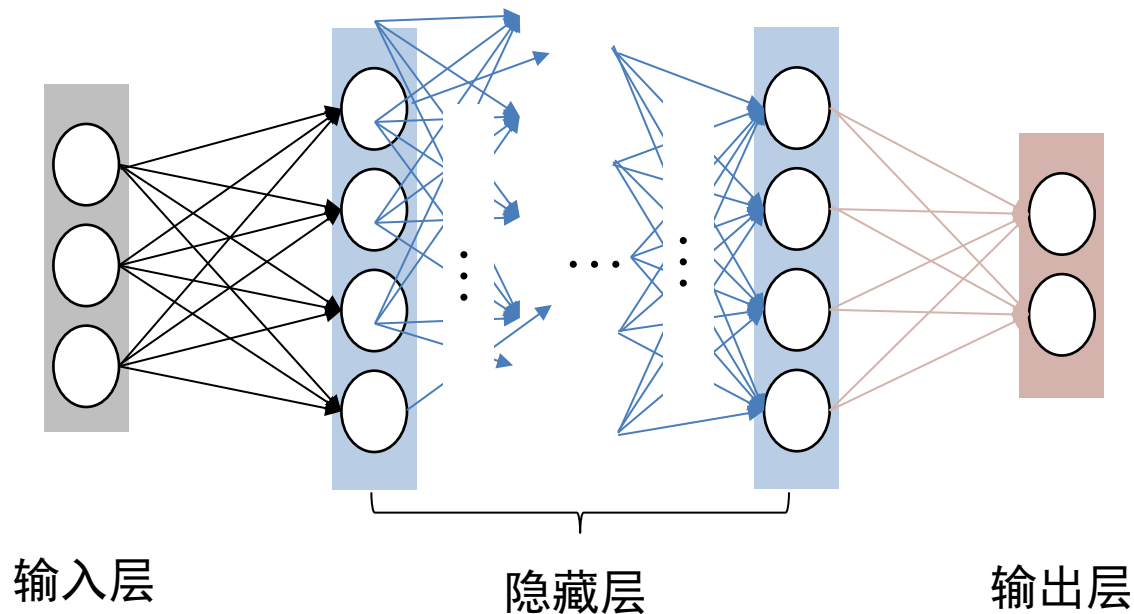


$$0 < y_i < 1, \sum_i y_i = 1$$

- 对于取值为4、1和-4的 x_1 、 x_2 和 x_3 ，通过softmax变换后，将其映射到(0,1)之间的概率值。
- 由于Softmax输出结果的值累加起来为1，因此可将输出概率最大的作为分类目标。

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。



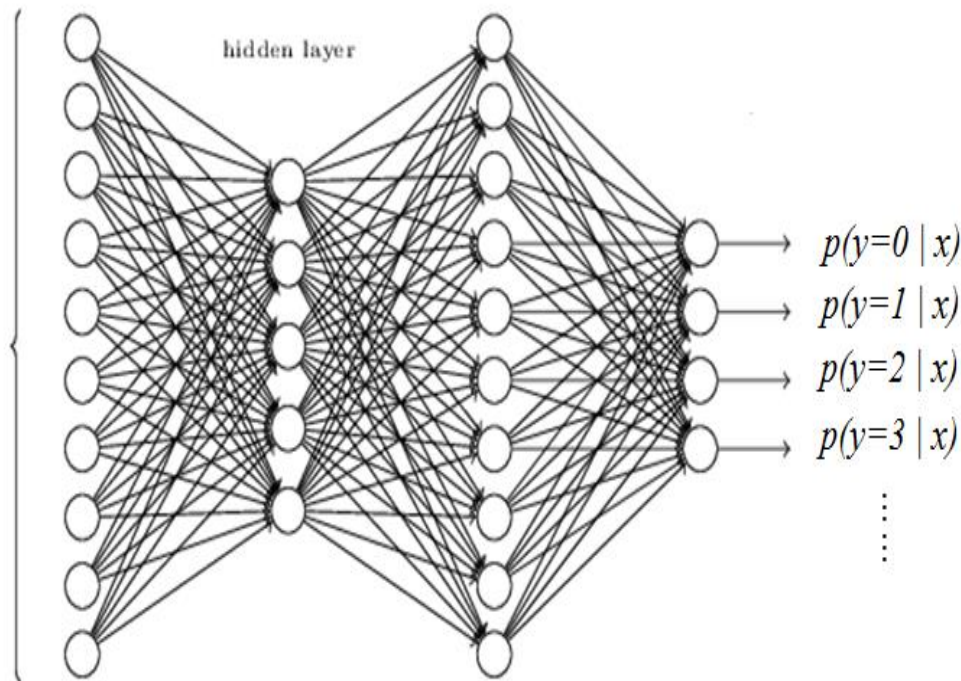
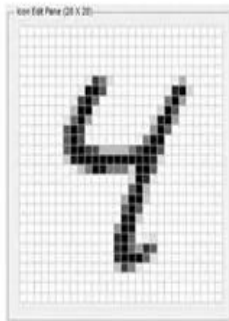
问题：如何优化网络参数？

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

n 为神经网络层数、 m 为每层中神经元个数

| | | | | |
|---|---|---|---|---|
| 2 | 9 | 6 | 1 | 3 |
| 3 | 9 | 4 | 0 | 3 |
| 6 | 9 | 4 | 1 | 9 |
| 9 | 5 | 0 | 8 | 5 |
| 8 | 8 | 3 | 5 | 0 |

input layer
(784 neurons)



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):

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)(\Delta x)^2 + \cdots + \frac{1}{n!}f^{(n)}(x)(\Delta x)^n$$

$$f(x + \Delta x) - f(x) \approx (\nabla f(x))^T \Delta x$$

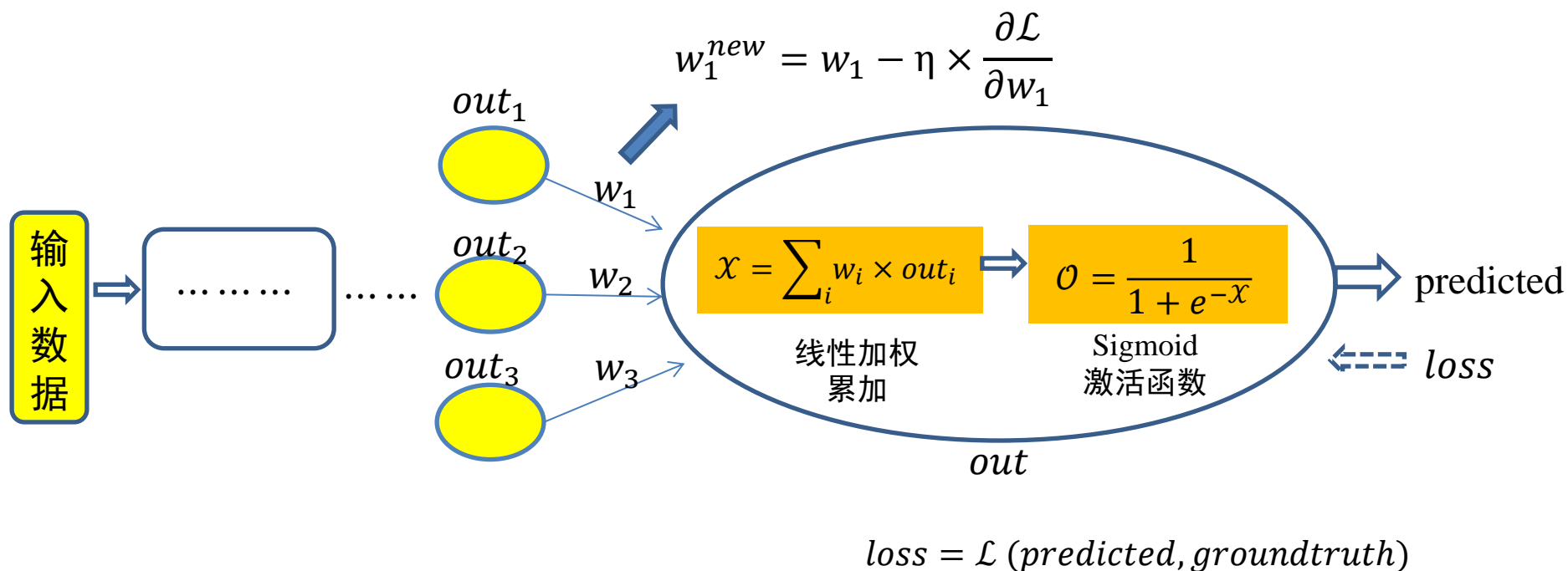
- 因为我们的目的是最小化损失函数 $f(x)$ ，则 $f(x + \Delta x) < f(x)$ ，于是 $(\nabla f(x))^T \Delta x < 0$
- 在 $(\nabla f(x))^T \Delta x = \|\nabla f(x)\| \|\Delta x\| \cos \theta$ 中， $\|\nabla f(x)\|$ 和 $\|\Delta x\|$ 分别为损失函数梯度的模和下一轮迭代中 x 取值增量的模，两者均为正数。为了保证损失误差减少，只要保证 $\cos \theta < 0$ 。当 $\theta = 180^\circ$ 时， $\cos \theta = -1$ ，这时损失函数减少的幅度值 $(\nabla f(x))^T \Delta x$ 取到最小。

梯度下降 (Gradient Descent)

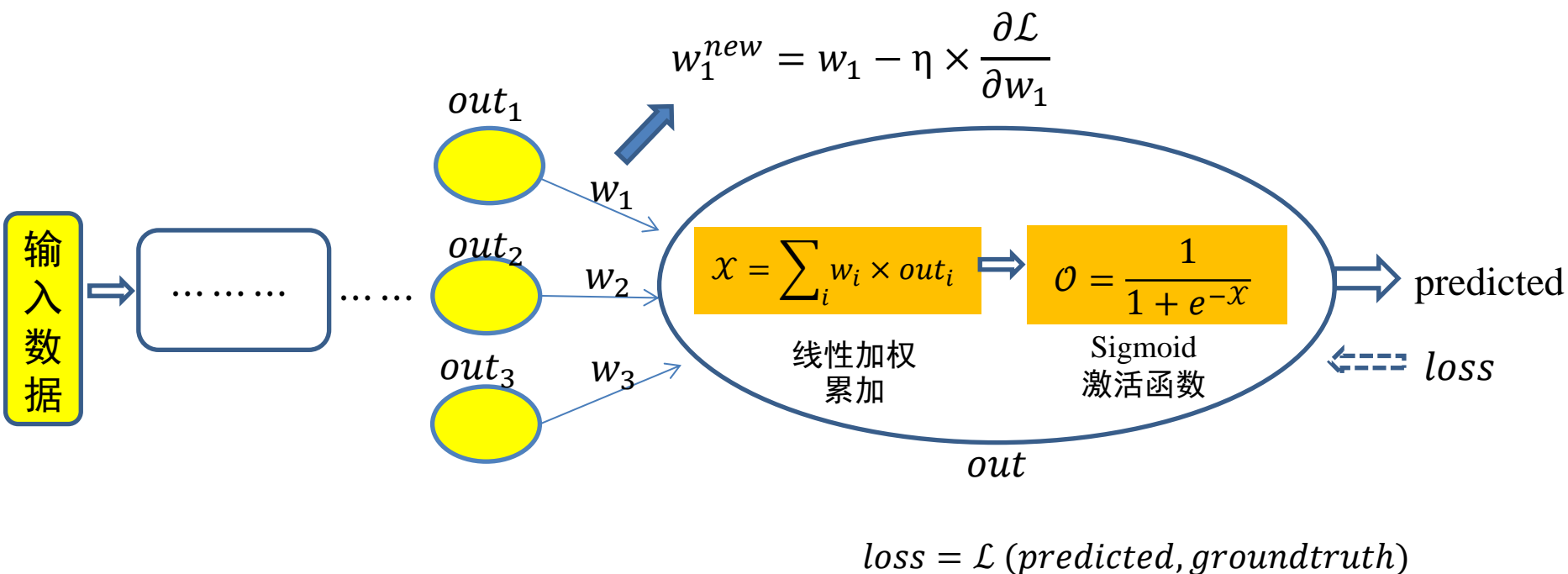
当 $\theta = 180^\circ$ 时, $f(x + \Delta x)$ 和 $f(x)$ 之间的差值为: $f(x + \Delta x) - f(x) = \|\nabla f(x)\| \|\Delta x\| \cos \theta = -\|\Delta x\| \|\nabla f(x)\|$ 。这说明只要沿着损失函数梯度的反方向选取 x 的增量 Δx , 就会保证损失误差下降最多、下降最快, 犹如从山峰处沿着最陡峭路径可快速走到山谷。在前面的推导中忽略了损失函数的二阶导数以及其高阶导数的取值, 因此在实际中引入步长 η , 用 $x - \eta \nabla f(x)$ 来更新 x (在具体实现时 η 可取一个定值)。

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。

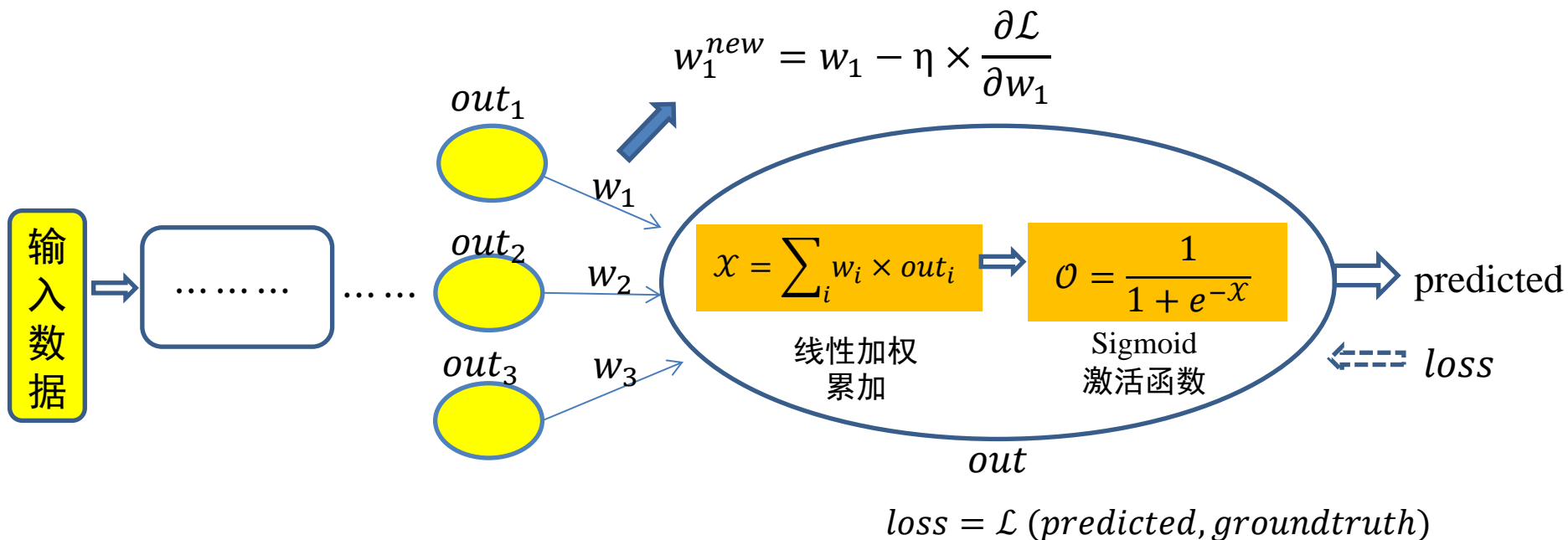


参数优化：误差反向传播 (error back propagation, BP)



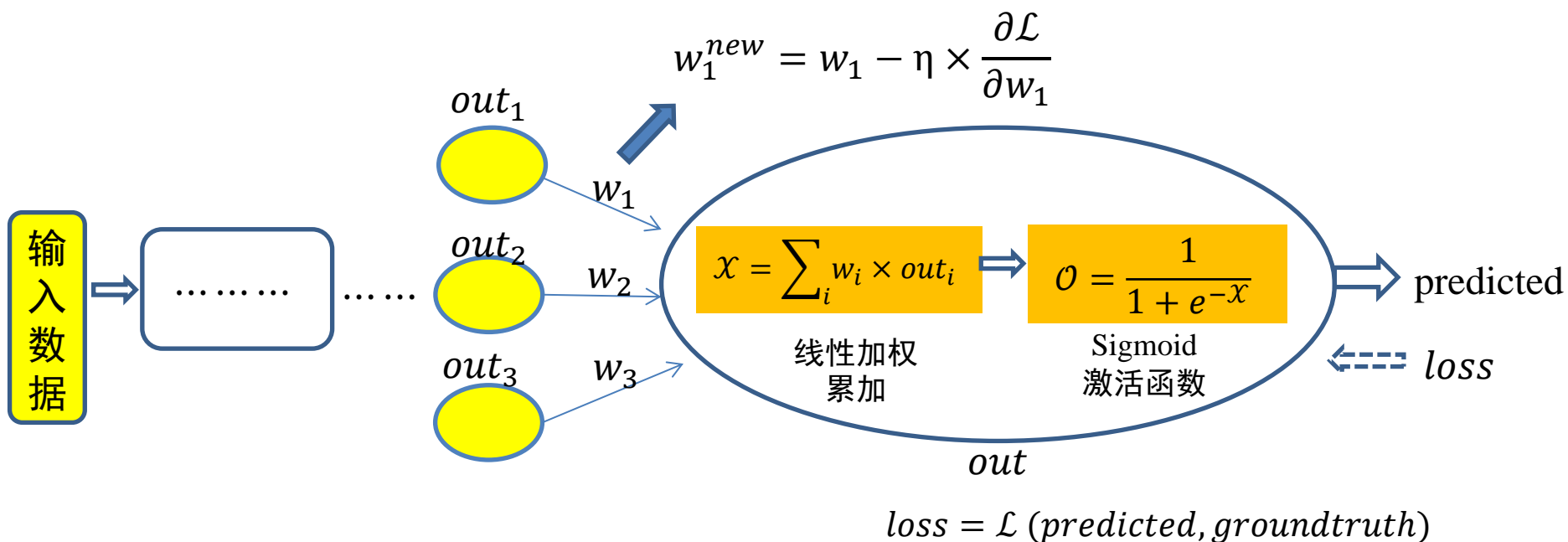
为了使损失函数 \mathcal{L} 取值减少（从而保证模型预测结果与实际结果之间的差距越来越小），需要求取损失函数 \mathcal{L} 相对于 w_1 的偏导，然后按照损失函数梯度的反方向选取一个微小的增量，来调整 w_1 的取值，就能够保证损失函数取值减少。即将 w_1 变为 $w_1 - \eta \frac{d\mathcal{L}}{dw_1}$ 后，能使得损失误差减少。这里 $\frac{d\mathcal{L}}{dw_1}$ 为损失函数 \mathcal{L} 对 w_1 的偏导。于是，需要求取损失函数 \mathcal{L} 对变量参数 w_1 的偏导。

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



由于 w_1 与加权累加函数 x 和sigmoid函数均有关，因此 $\frac{d\mathcal{L}}{dw_1} = \frac{d\mathcal{L}}{do} \frac{do}{dx} \frac{dx}{dw_1}$ 。在这个链式求导中： $\frac{d\mathcal{L}}{do}$ 与损失函数的定义有关； $\frac{do}{dx}$ 是对sigmoid函数求导，结果为 $\frac{1}{1+e^{-x}} \times (1 - \frac{1}{1+e^{-x}})$ ； $\frac{dx}{dw_1}$ 是加权累加函数 $w_1 \times out_1 + w_2 \times out_2 + w_3 \times out_3$ 对 w_1 求导，结果为 out_1 。链式求导实现了损失函数对某个自变量求偏导，好比将损失误差从输出端向输入端逐层传播，通过这个传播过程来更新该自变量取值。梯度下降法告诉我们，只要沿着损失函数梯度的反方向来更新参数，就可使得损失函数下降最快。

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

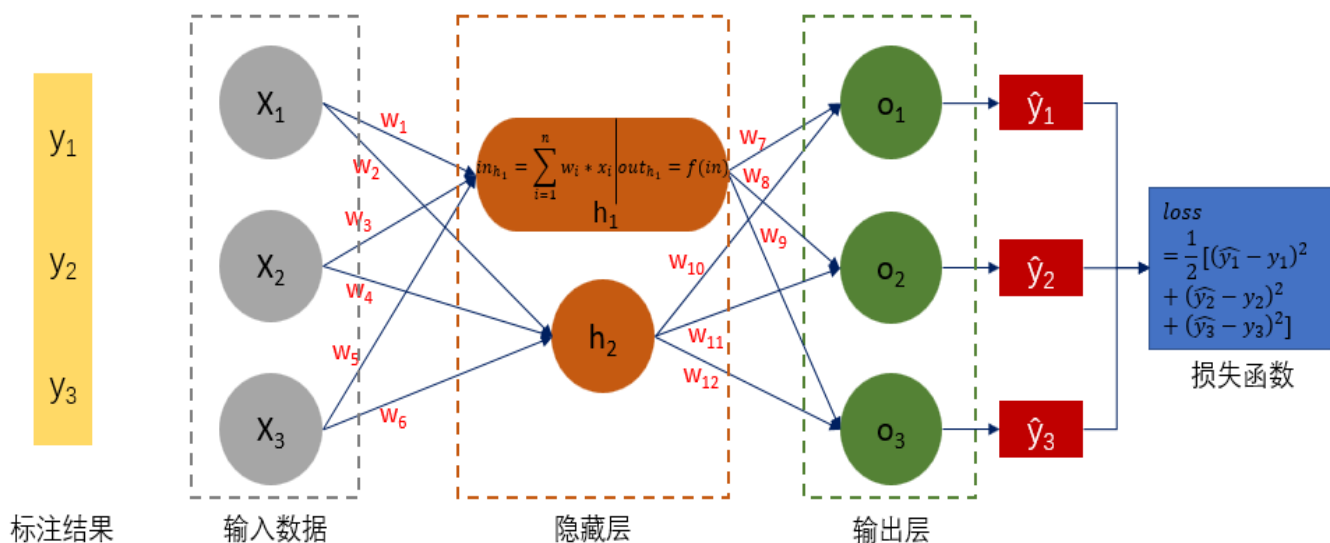


参数 w_1 在下一轮迭代中的取值被调整为： $w_1^{new} = w_1 - \eta \times \frac{d\mathcal{L}}{dw_1} = w_1 - \eta \times$

$\frac{d\mathcal{L}}{d\sigma} \frac{d\sigma}{dx} \frac{dx}{dw_1} = w_1 - \eta \times \frac{d\mathcal{L}}{d\sigma} \times \frac{1}{1+e^{-x}} \times (1 - \frac{1}{1+e^{-x}}) \times out_1$ 。按照同样的方法，可调整 w_2 、

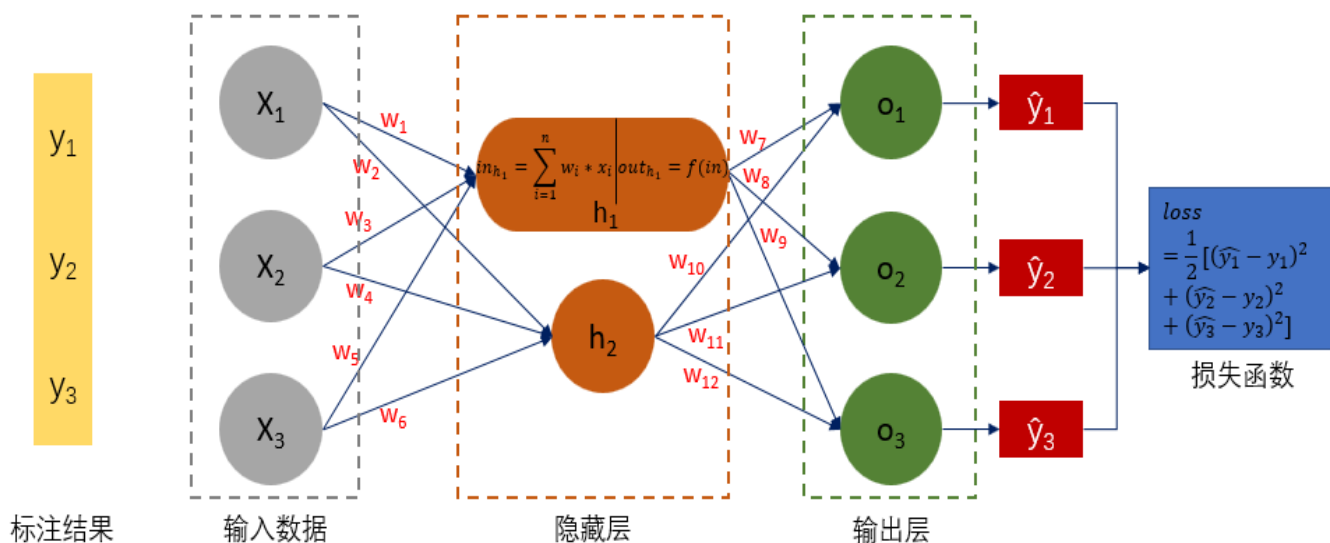
w_3 以及其它图中未显示参数的取值。经过这样的调整，在模型参数新的取值作用下，损失函数 $\mathcal{L}(\theta)$ 会以最快下降方式逐渐减少，直至减少到最小值（即全局最小值）。

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



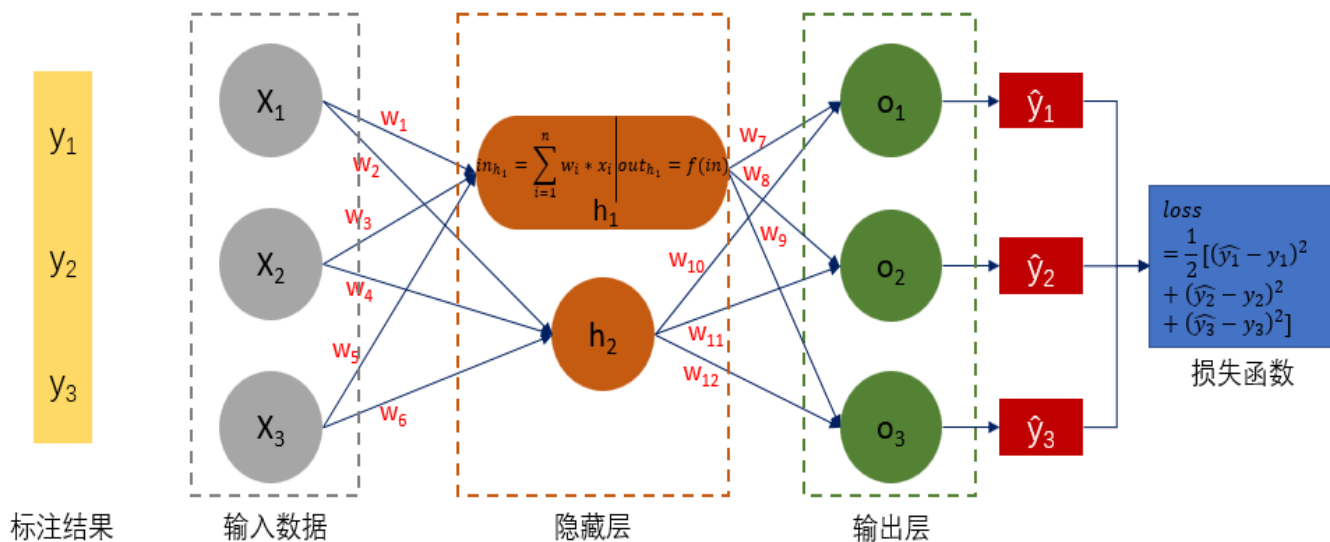
通过一个三类分类的具体例子来介绍神经网络中参数更新过程。给定一个包含输入层、一层隐藏层和输出层的多层感知机，其中隐藏层由两个神经元构成。网络使用Sigmoid函数作为神经元的激活函数，使用均方损失函数来计算网络输出值与实际值之间的误差。

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



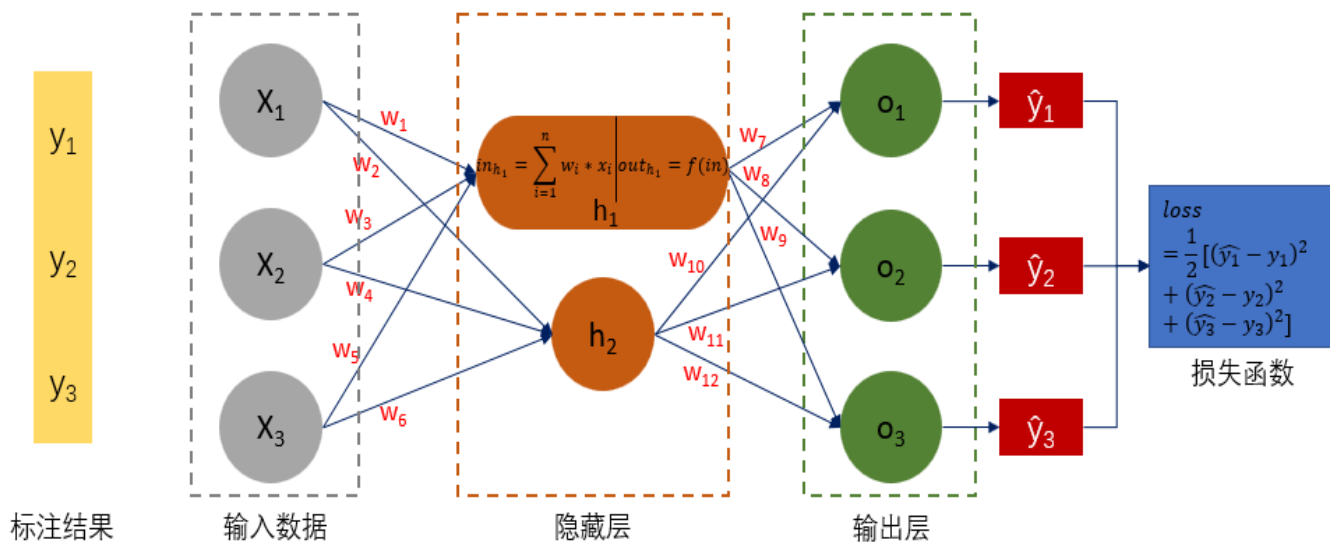
每个神经元完成如下两项任务：1) 对相邻前序层所传递信息进行线性加权累加；2) 对加权累加结果进行非线性变换。假设样本数据输入为 (x_1, x_2, x_3) ，其标注信息为 (y_1, y_2, y_3) 。在三类分类问题中，对于输入数据 (x_1, x_2, x_3) ， y_1 、 y_2 和 y_3 中只有一个取值为1、其余两个取值为0。

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



| 单元 | 相邻前序神经元传递信息线性累加 | 非线性变换 |
|-------|---|---|
| h_1 | $In_{h_1} = w_1 * x_1 + w_3 * x_2 + w_5 * x_3$ | $Out_{h_1} = Sigmoid(In_{h_1})$ |
| h_2 | $In_{h_2} = w_2 * x_1 + w_4 * x_2 + w_6 * x_3$ | $Out_{h_2} = Sigmoid(In_{h_2})$ |
| o_1 | $In_{o_1} = w_7 * Out_{h_1} + w_{10} * Out_{h_2}$ | $\hat{y}_1 = Out_{o_1} = Sigmoid(In_{o_1})$ |
| o_2 | $In_{o_2} = w_8 * Out_{h_1} + w_{11} * Out_{h_2}$ | $\hat{y}_2 = Out_{o_2} = Sigmoid(In_{o_2})$ |
| o_3 | $In_{o_3} = w_9 * Out_{h_1} + w_{12} * Out_{h_2}$ | $\hat{y}_3 = Out_{o_3} = Sigmoid(In_{o_3})$ |

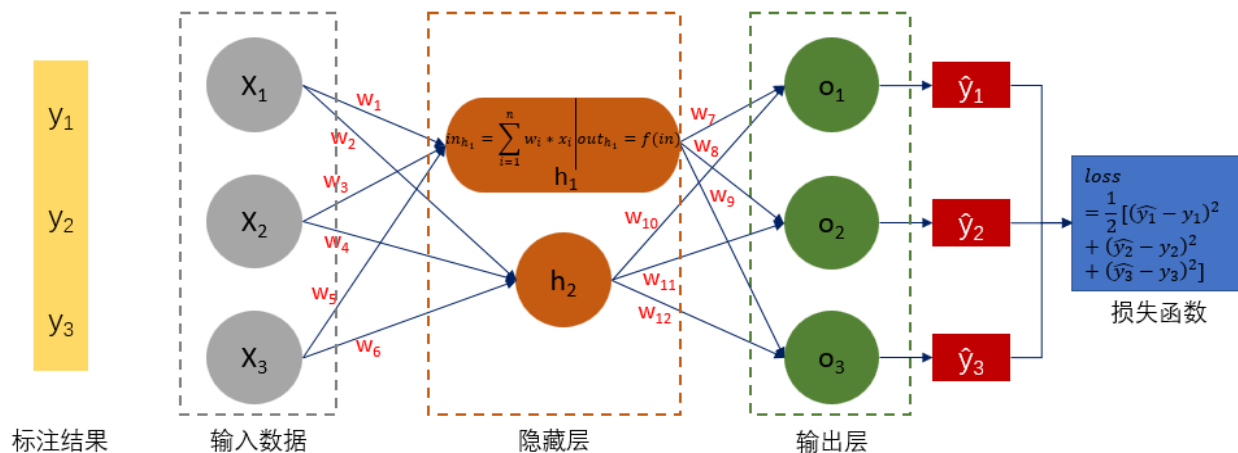
参数优化：误差反向传播 (error back propagation, BP)



一旦神经网络在当前参数下给出了预测结果 $(\hat{y}_1, \hat{y}_2, \hat{y}_3)$ 后, 通过均方误差损失函数来计算模型预测值与真实值 (y_1, y_2, y_3) 之间误差, 记为 $loss = \frac{1}{3} \sum_{i=1}^3 (\hat{y}_i - y_i)^2$ 。

接下来通过梯度下降和误差反向传播方法, 沿着损失函数梯度的反方向来更改参数变量取值, 使得损失函数快速下降到其最小值。由于损失函数对 $w_7 \sim w_{12}$ 的偏导计算相似, 在此以 w_7 为例来介绍如何更新 w_7 这一参数取值。记损失函数为 $\mathcal{L}(w)$, $\mathcal{L}(w) = \frac{1}{3} \sum_{i=1}^3 (\hat{y}_i - y_i)^2$ 。

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



损失函数 \mathcal{L} 对参数 w_7 的偏导可如下计算：

$$\delta_7 = \frac{\partial \mathcal{L}}{\partial w_7} = \frac{\partial \mathcal{L}}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial \ln_{o_1}} * \frac{\partial \ln_{o_1}}{\partial w_7}$$

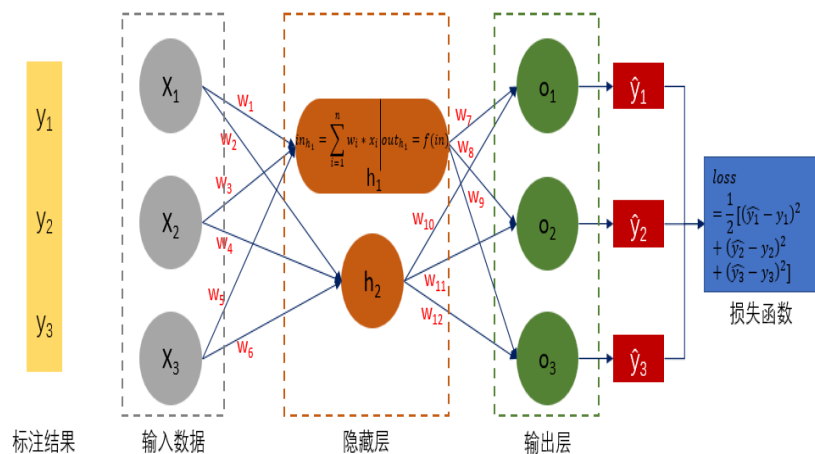
上述公式中每一项结果如下：

$$\frac{\partial \mathcal{L}}{\partial \hat{y}_1} = \hat{y}_1 - y_1$$

$$\frac{\partial \hat{y}_1}{\partial \ln_{o_1}} = \hat{y}_1 * (1 - \hat{y}_1)$$

$$\frac{\partial \ln_{o_1}}{\partial w_7} = \text{out}_{h_1}$$

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

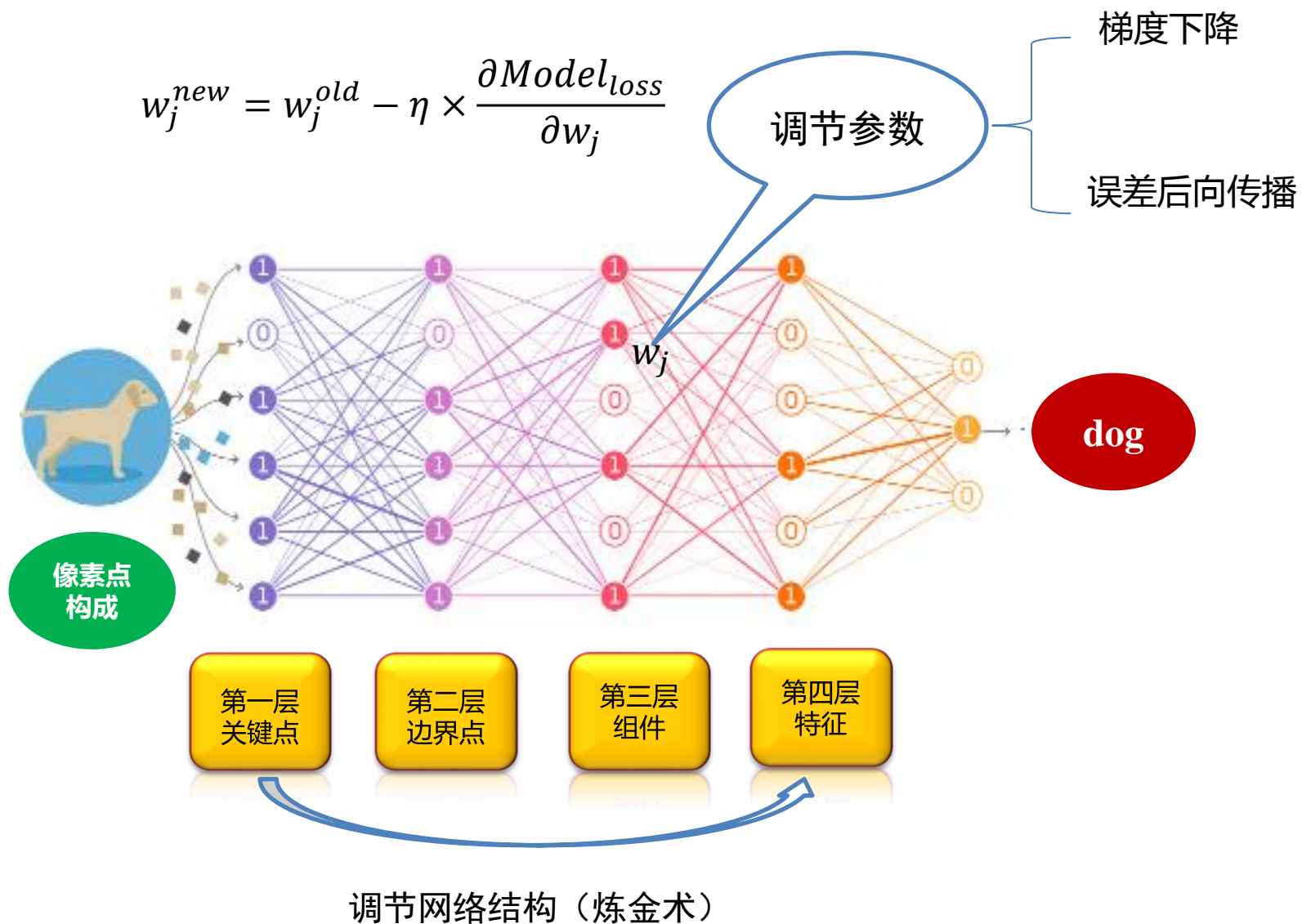


下面以 w_1 为例介绍损失函数 \mathcal{L} 对 w_1 的偏导 δ_1 。

$$\begin{aligned}
 \delta_1 &= \frac{\partial \mathcal{L}}{\partial w_1} \\
 &= \frac{\partial \mathcal{L}}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial w_1} + \frac{\partial \mathcal{L}}{\partial \hat{y}_2} * \frac{\partial \hat{y}_2}{\partial w_1} + \frac{\partial \mathcal{L}}{\partial \hat{y}_3} * \frac{\partial \hat{y}_3}{\partial w_1} \\
 &= \frac{\partial \mathcal{L}}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial \ln_{o_1}} * \frac{\partial \ln_{o_1}}{\partial \text{Out}_{h_1}} * \frac{\partial \text{Out}_{h_1}}{\partial \ln_{h_1}} * \frac{\partial \ln_{h_1}}{\partial w_1} + \frac{\partial \mathcal{L}}{\partial \hat{y}_2} * \frac{\partial \hat{y}_2}{\partial \ln_{o_2}} * \frac{\partial \ln_{o_2}}{\partial \text{Out}_{h_1}} * \frac{\partial \text{Out}_{h_1}}{\partial \ln_{h_1}} * \frac{\partial \ln_{h_1}}{\partial w_1} \\
 &\quad + \frac{\partial \mathcal{L}}{\partial \hat{y}_3} * \frac{\partial \hat{y}_3}{\partial \ln_{o_3}} * \frac{\partial \ln_{o_3}}{\partial \text{Out}_{h_1}} * \frac{\partial \text{Out}_{h_1}}{\partial \ln_{h_1}} * \frac{\partial \ln_{h_1}}{\partial w_1} \\
 &= \left(\frac{\partial \mathcal{L}}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial \ln_{o_1}} * \frac{\partial \ln_{o_1}}{\partial \text{Out}_{h_1}} + \frac{\partial \mathcal{L}}{\partial \hat{y}_2} * \frac{\partial \hat{y}_2}{\partial \ln_{o_2}} * \frac{\partial \ln_{o_2}}{\partial \text{Out}_{h_1}} + \frac{\partial \mathcal{L}}{\partial \hat{y}_3} * \frac{\partial \hat{y}_3}{\partial \ln_{o_3}} * \frac{\partial \ln_{o_3}}{\partial \text{Out}_{h_1}} \right) * \frac{\partial \text{Out}_{h_1}}{\partial \ln_{h_1}} * \frac{\partial \ln_{h_1}}{\partial w_1} \\
 &= (\delta_7 + \delta_8 + \delta_9) * \frac{\partial \text{Out}_{h_1}}{\partial \ln_{h_1}} * \frac{\partial \ln_{h_1}}{\partial w_1}
 \end{aligned}$$

在计算得到所有参数相对于损失函数 \mathcal{L} 的偏导结果后，利用梯度下降算法，通过 $w_i^{new} = w_i - \eta * \delta_i$ 来更新参数取值。然后不断迭代，直至模型参数收敛，此时损失函数减少到其最小值。

深度神经网络的拟合优化



提纲

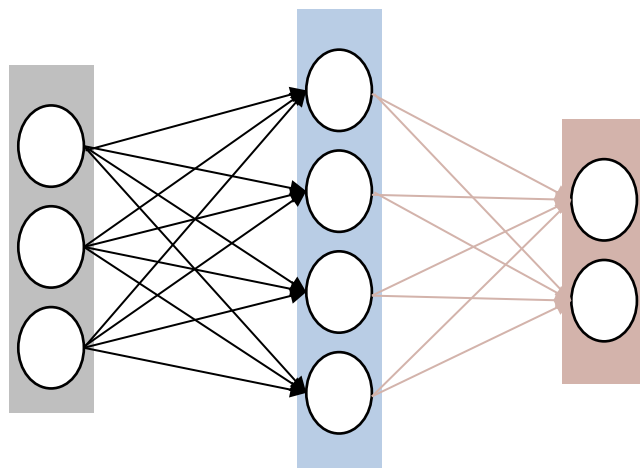
一、前馈神经网络

二、卷积神经网络

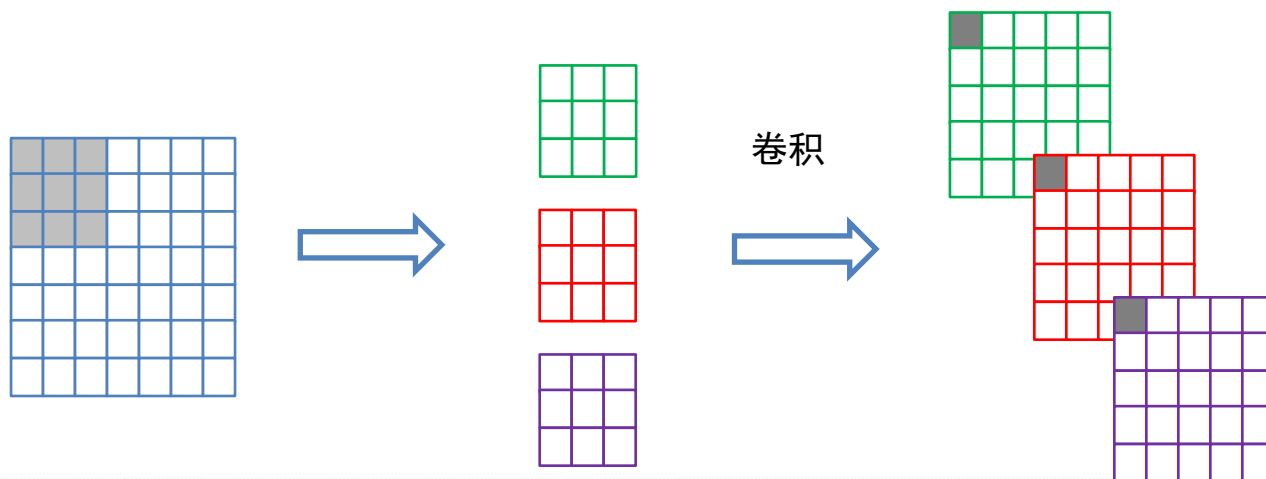
三、若干应用

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

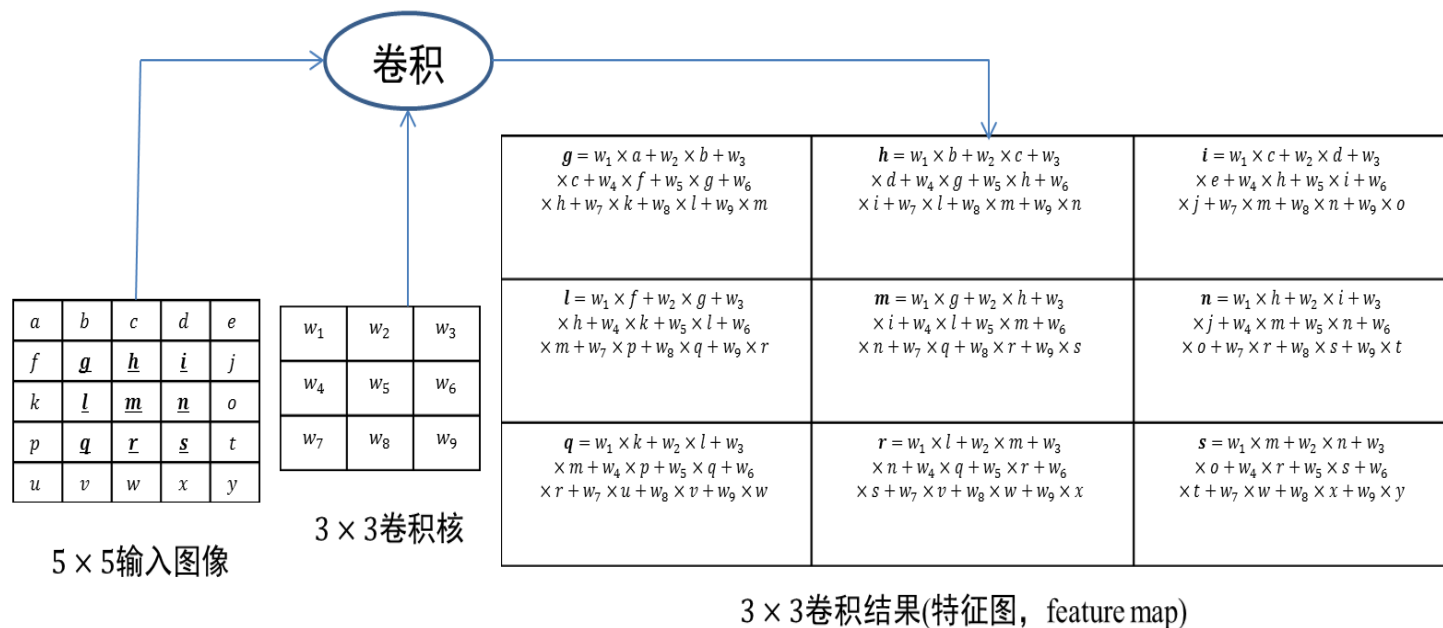
三层前馈神经网络



卷积神经网络

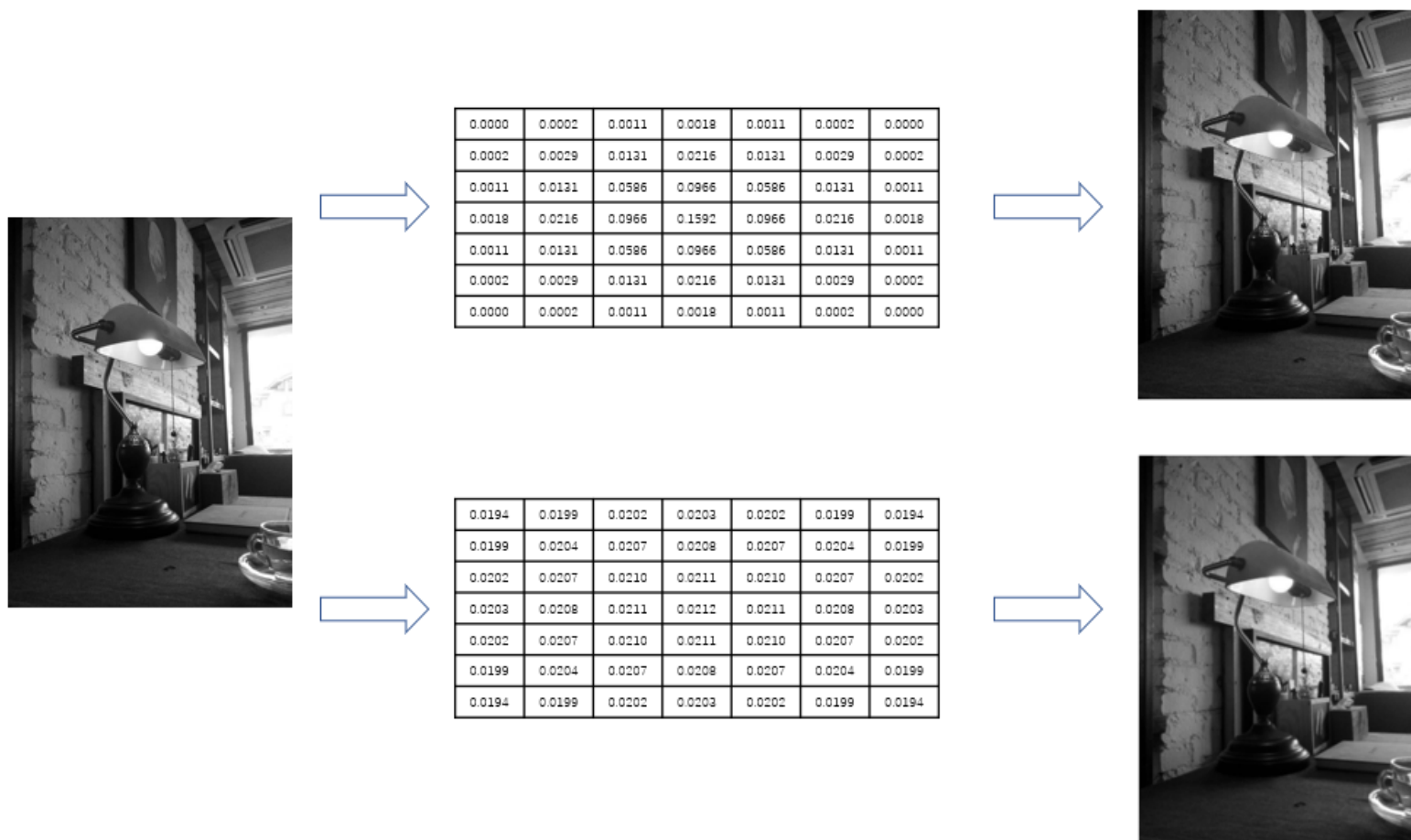


卷积神经网络：卷积操作



给定一个权重分别为 w_i ($1 \leq i \leq 9$)、大小为 3×3 的卷积核以及一个 5×5 大小灰度图像。该卷积核对图像的卷积操作就是分别以图像中 g 、 h 、 i 、 l 、 m 、 n 、 q 、 r 和 s 所在像素点位置为中心，形成一个 3×3 大小的图像子块区域，然后用卷积核所定义权重 w_i 对图像子块区域内每个像素点进行加权累加，所得结果即为图像子块区域中心像素点被卷积后（即滤波后）的结果。滤波也可视为在给定卷积核权重前提下，记住了邻域像素点之间的若干特定空间模式、忽略了某些空间模式。

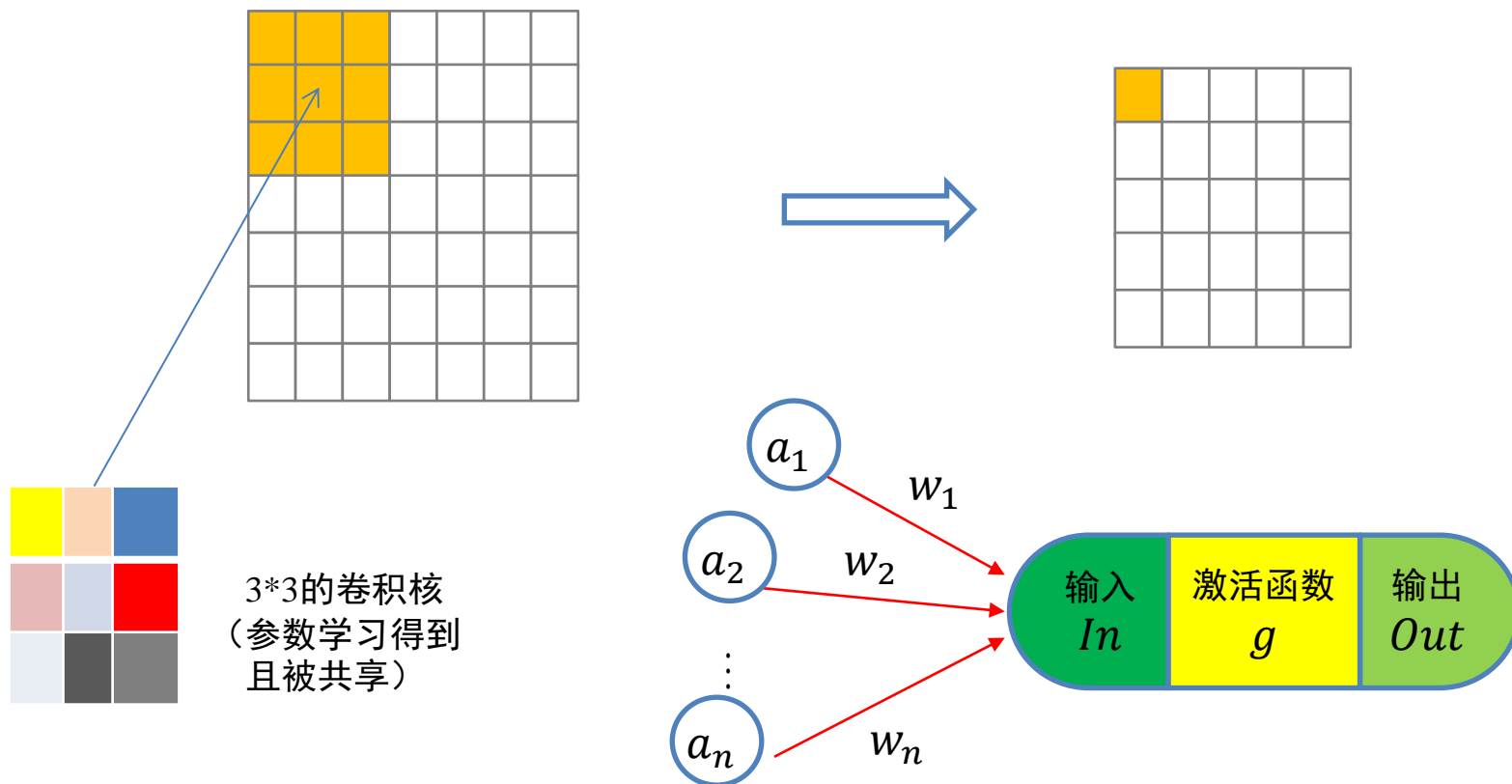
卷积神经网络：卷积操作



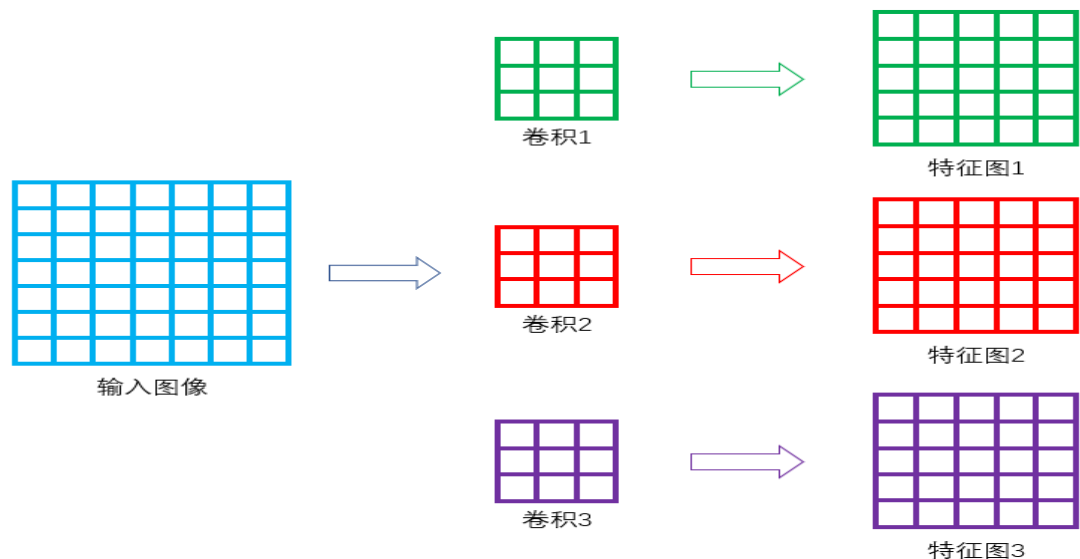
如果卷积核中心位置的权重系数越小且与其它卷积权重系数差别越小，则卷积所得到图像滤波结果越模糊，这被称为图像平滑操作

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

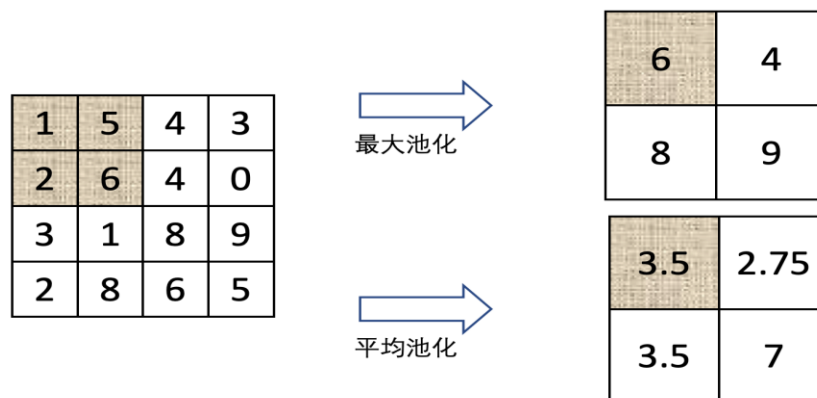


卷积神经网络：卷积操作



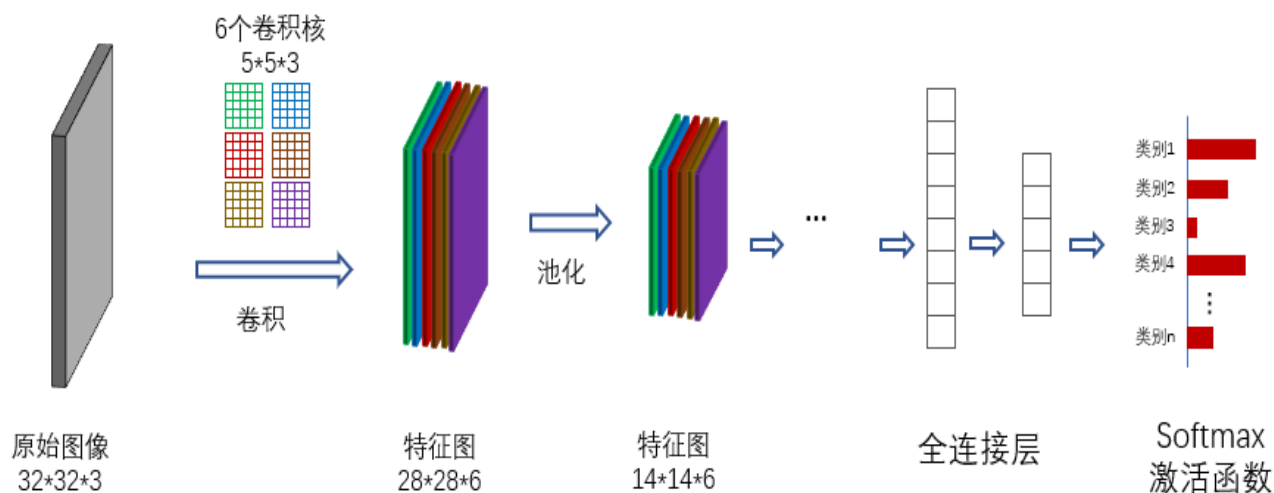
神经科学家发现，人的视觉神经细胞对不同的视觉模式具有特征选择性（Feature Selectivity），即不同视觉神经细胞对边界、运动和颜色等不同信息具有强弱不同的选择性。因此，不同卷积核可被用来刻画视觉神经细胞对外界信息感受时的不同选择性。同时也可以看到，卷积所得结果中，每个输出点的取值仅依赖于其在输入图像中该点及其邻域区域点的取值，与这个区域之外的其他点取值均无关，该区域被称为感受野（receptive field），正所谓“管中窥豹、见微知著”。在卷积神经网络中，感受野是卷积神经网络每一层输出的特征图（feature map）上的像素点在输入图像上映射的区域大小。也就是说，感受野是特征图上一个点对应输入图像上的区域。

卷积神经网络：池化操作



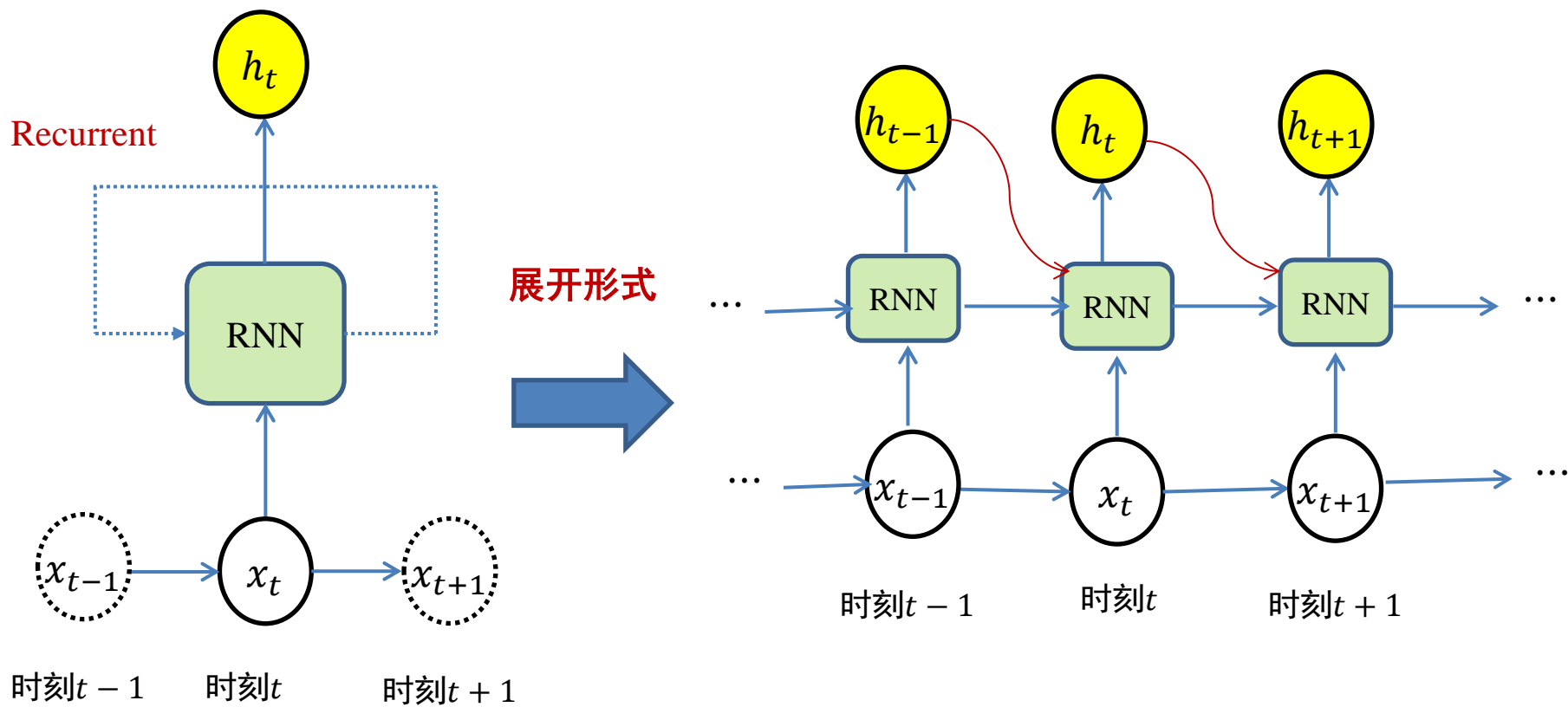
由于图像中存在较多冗余，在图像处理中，可用某一区域子块的统计信息（如最大值或均值等）来刻画该区域中所有像素点呈现的空间分布模式，以替代区域子块中所有像素点取值，这就是卷积神经网络中池化(pooling)操作。池化操作对卷积结果特征图进行约减，实现了下采样，同时保留了特征图中主要信息。

卷积神经网络：池化操作

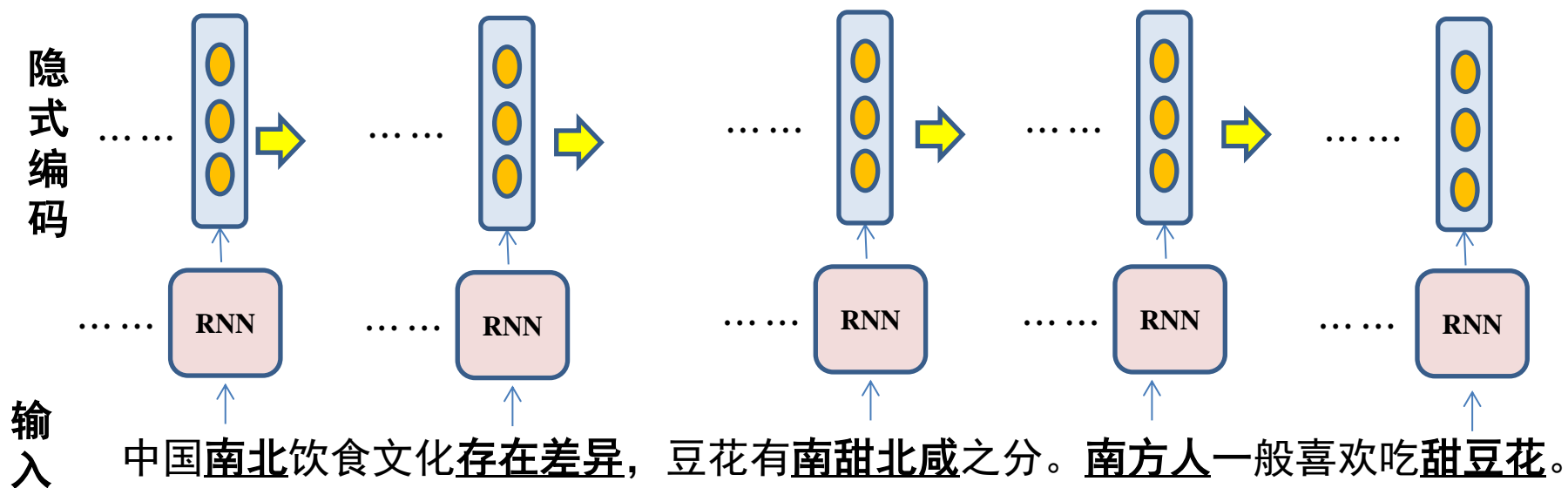


对于输入的海量标注数据，通过多次迭代训练，卷积神经网络在若干次卷积操作、接着对卷积所得结果进行激活函数操作和池化操作下，最后通过全连接层来学习得到输入数据的特征表达，即分布式向量表达(distributed vector representation)。

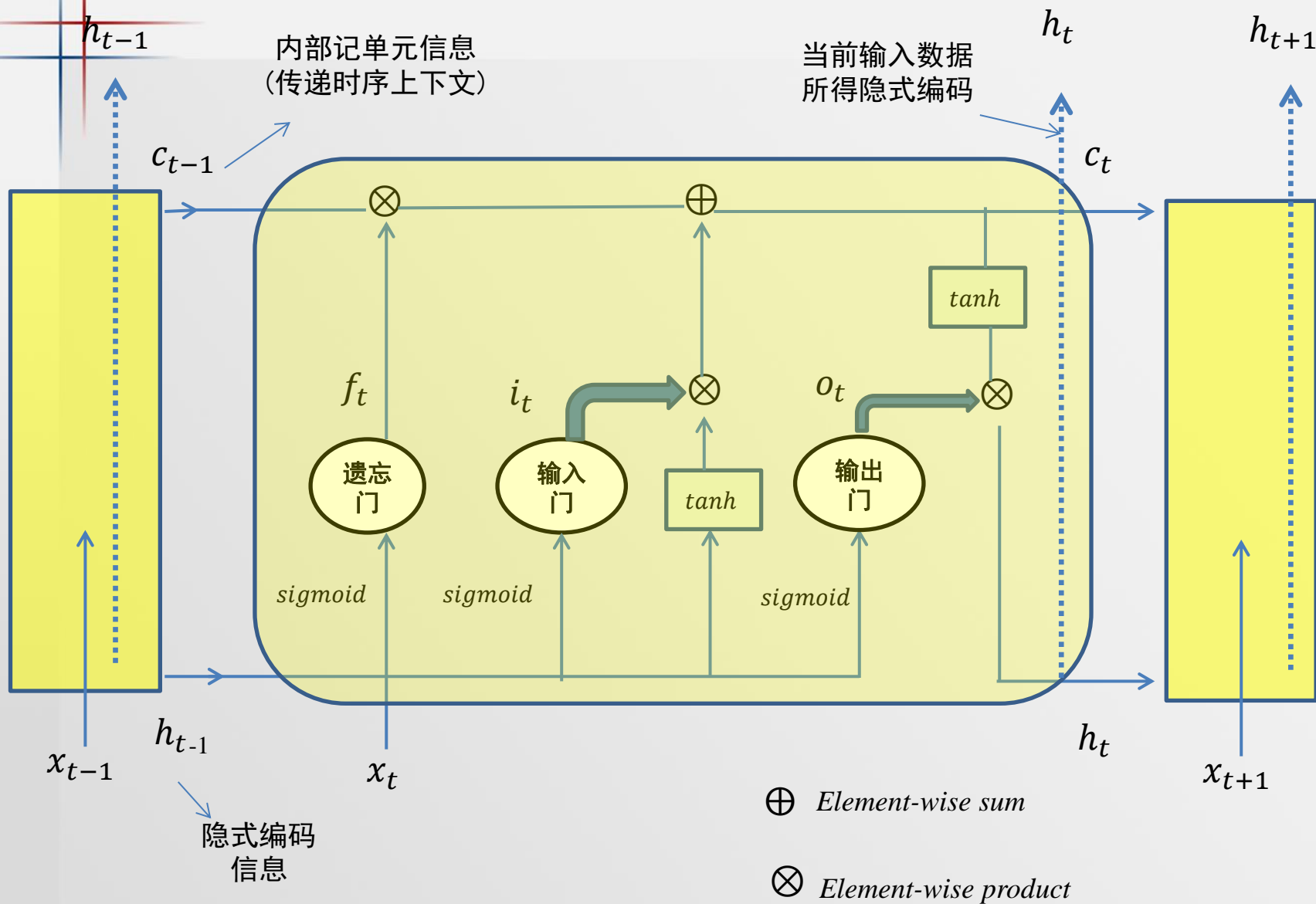
卷积神经网络：循环神经网络



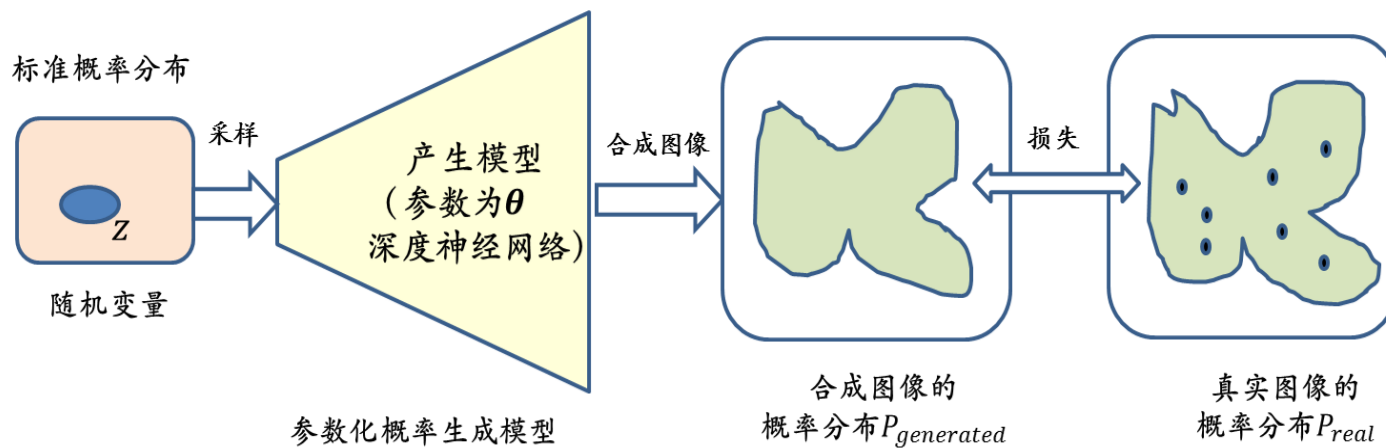
卷积神经网络：循环神经网络



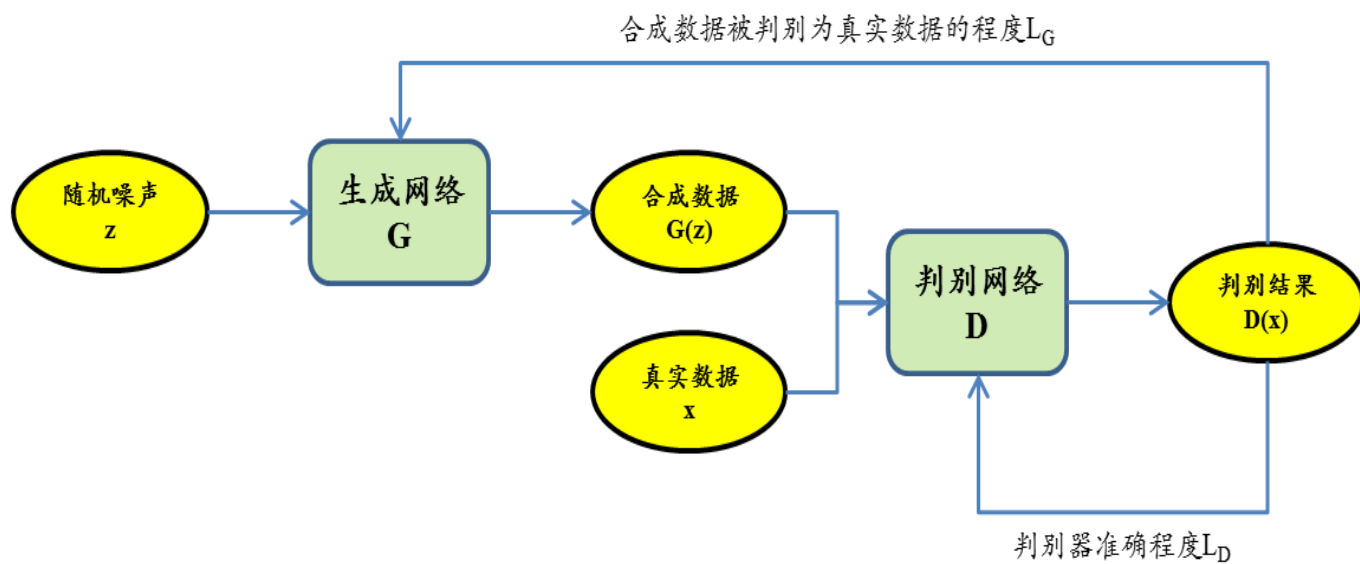
长短记忆网络模型



深度生成学习模型 (deep generative learning model)



生成对抗网络 (deep generative learning model)



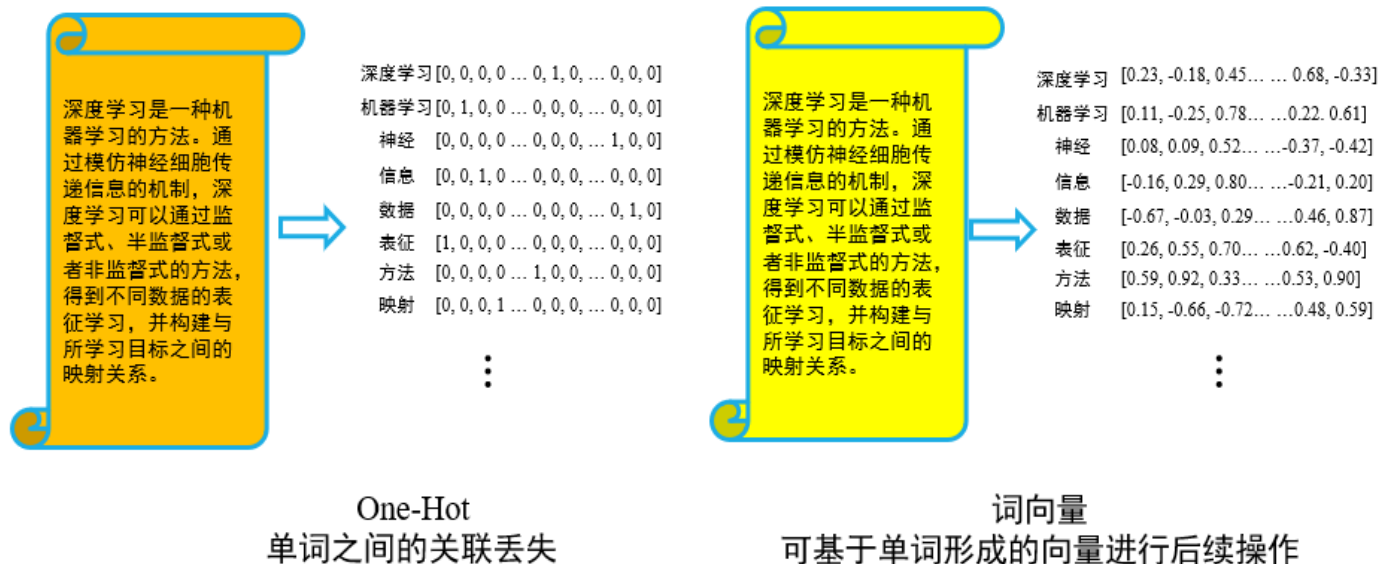
提纲

一、前馈神经网络

二、卷积神经网络

三、若干应用

自然语言中词向量生成



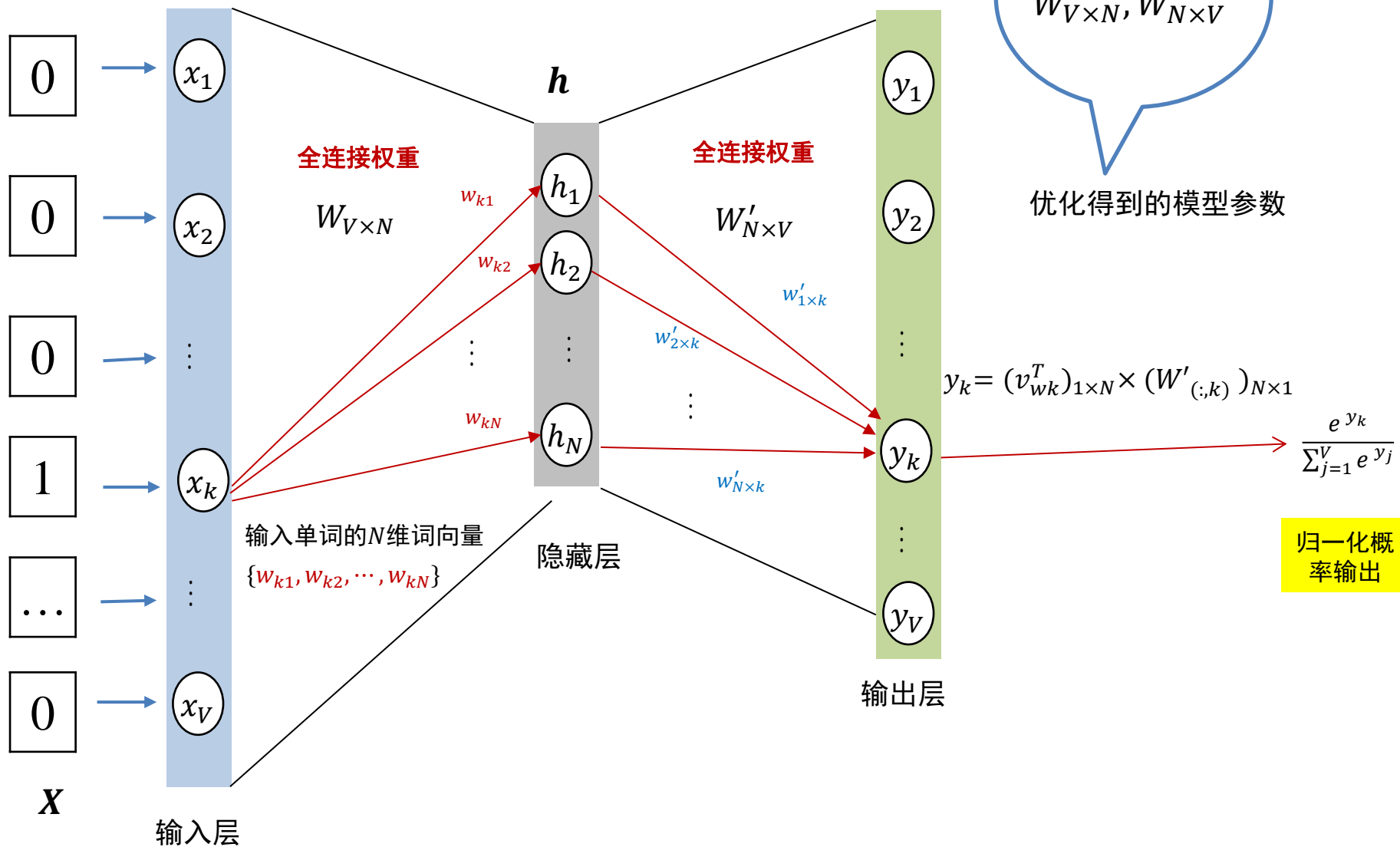
为了表达文本，“词袋”（Bag of Words）模型被采用，但是这一模型忽略了文本单词之间的依赖关系，仅仅将文本看作是单词的集合（忽略了单词之间的先后次序）。在词袋模型中，一个单词按照词典序被表示为一个词典维数大小的向量（被称为one-hot vector），向量中该单词所对应位置按照其在文档中出现与否取值为1或0。1表示该单词在文档中出现、0表示没有出现。每个单词被表达为单词向量后，一些与自然语言相关工作（如聚类、同义词计算、主题挖掘等）可以依次开展。

Word2Vec 模型

one-hot
表达

输入单词的 N 维隐式表达

$$\mathbf{h} = \mathbf{W}^T \times \mathbf{X} = v_{wk}^T$$



图像分类和目标定位

