人工智能与大数据

第四章 深度学习

提纲

一、深度学习历史发展

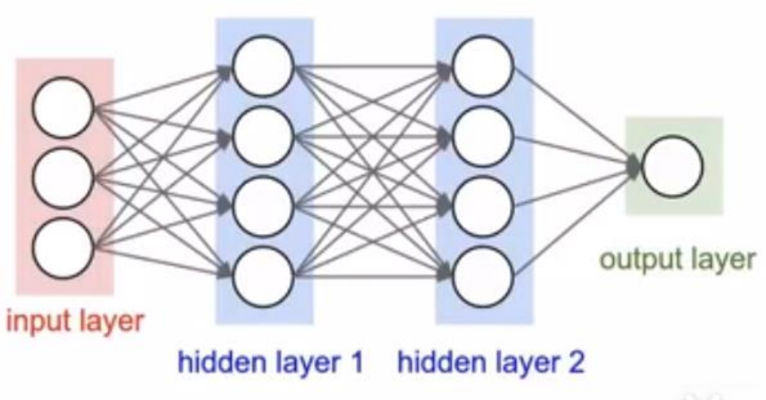
二、前馈神经网络

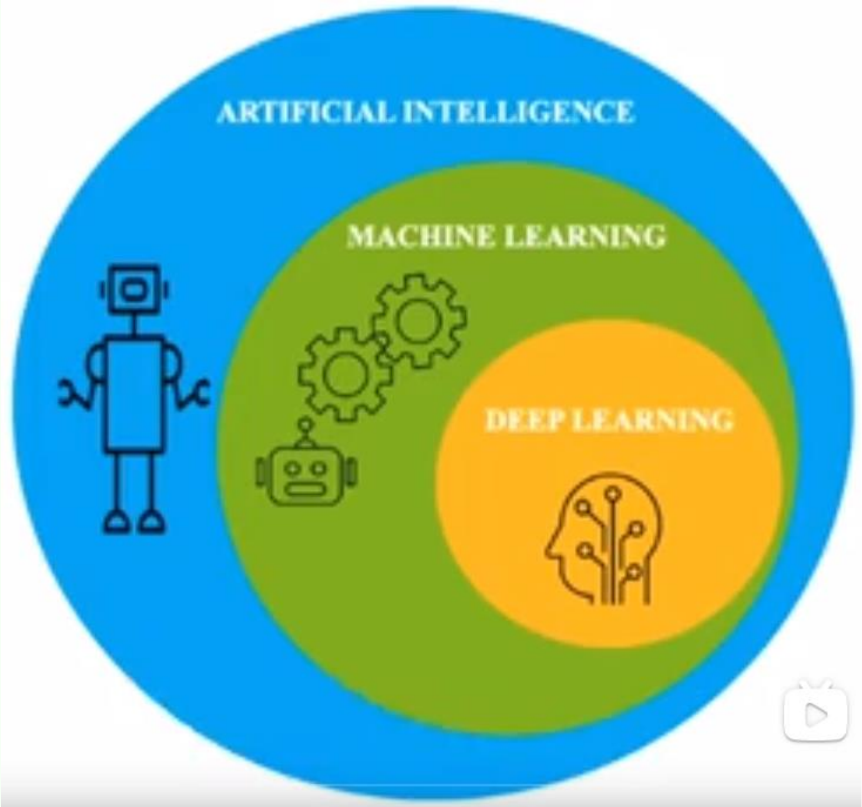
三、卷积神经网络

四、循环神经网络

人工智能：一个广义的术语，使计算机具有人类智慧。
机器学习：人工智能的一个分支，专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能。

深度学习：是机器学习领域中一个新的研究方向。包括卷积神经网络、深度置信网络等。





深度学习的历史发展

1943年，神经科学家Warren McCulloch和逻辑学家Walter Pitts合作提出了“McCulloch–Pitts (MCP) neuron”的思想。MCP可对输入信号线性加权组合，再用符号函数来输出线性加权组合结果，以模拟大脑复杂活动模式。MCP是最早的神经网络雏形。

“我们在科学史上第一次知道了‘我们是怎么知道的’（for the first time in the history of science, *we know how we know*)”

赫布理论(Hebbian theory)：
神经元之间持续重复经验刺激可导致突触传递效能增加.

Neurons that fire together, wire together.

“神经元之间突触的强弱变化是学习与记忆的生理学基础”这一理论为联结主义人工智能研究提供了认知神经心理学基础。

1958年，David Hubel和Torsten Wiesel在实验中发现小猫后脑皮层中不同视觉神经元与瞳孔所受刺激之间存在某种对应关系，由此发现了一种被称为“方向选择性细胞(orientation selective cell)”的神经元细胞，从而揭示了“视觉系统信息分层处理”这一机制。

MCP	Hebbian Theory	Orientation Selective Cell	MLP	Error Back-propagation	Deep Belief Network	○○○
1943	1949	1958	1969	1986	2006	

深度学习的历史发展

神经网络研究的突破来自于Frank Rosenblatt在20世纪50年代所提出的“感知机（perceptron）”模型。

由于**感知机**中没有包含非线性变换操作的隐藏层，因此感知机表达能力较弱（如无法解决异或问题）。

最早由Werbos提出[Werbos 1974][Werbos 1990]、并且由Rumelhar和Hinton等人[Rumelhart 1986]完善的**误差后向传播**（error backpropagation）算法解决了多层感知机中参数优化这一难题。

2006年，Hinton在《Science》等期刊上发表了论文，首次提出了“**深度信念网络**(deep belief network)”模型[Hinton 2006]，在相关分类任务上可取得性能超过了传统浅层学习模型（如支持向量机），使得深度架构引起了大家的关注。

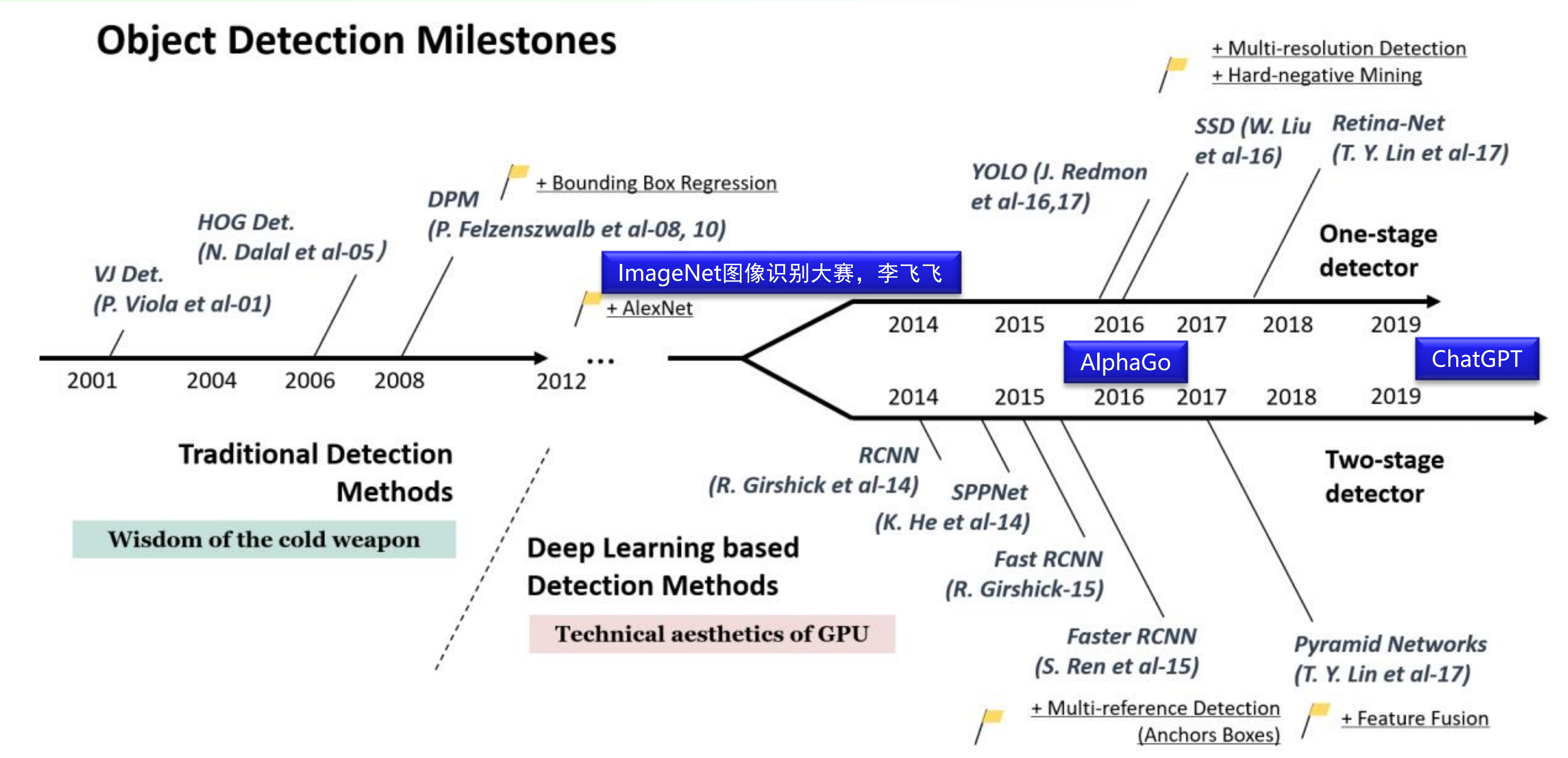
深度学习元年

MCP	Hebbian Theory	Orientation Selective Cell	MLP	Error Back-propagation	Deep Belief Network
1943	1949	1958	1969	1986	2006

深度学习的历史发展

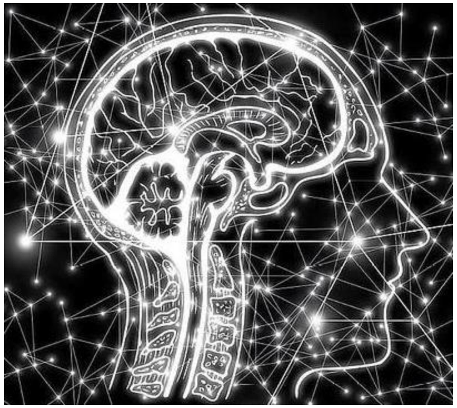
❑深度学习迅猛发展

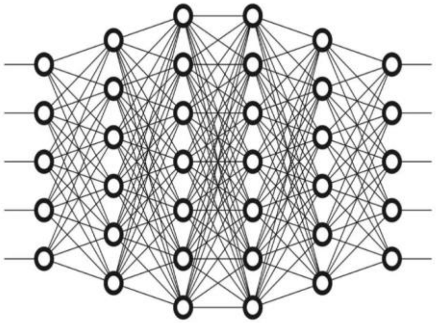
❖ RNN LSTM BERT RCNN YOLOV1-V6 Transformer



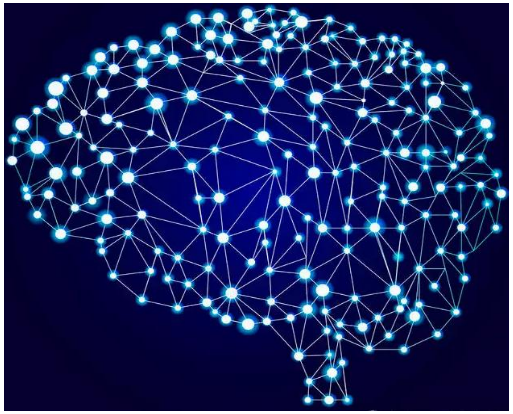
问题?

- 大脑是如何学习的?
  - 神经元传导
  - 神经键改变
- 为什么要模拟人脑?
  - 快速、智能
  - 稳定
- 我们如何模拟人脑的神经元?



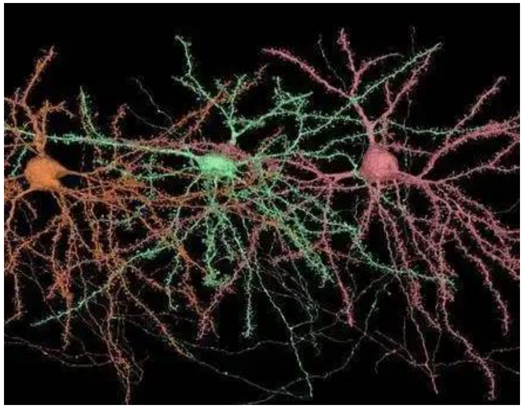


引言----生物神经元

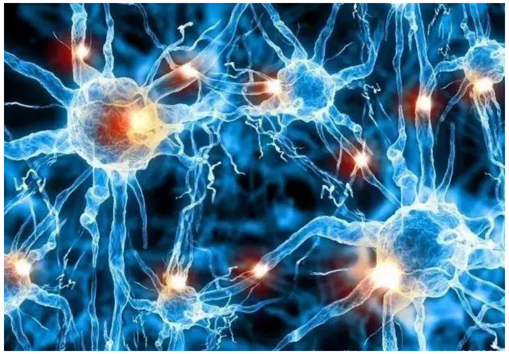


巨大的复杂网络:

- 860 亿个神经元
- 每个神经元有上千个突触和其他神经元相连接
- 神经连接的总长度可达数千公里
- #因特网逊色



- 神经元可以接收其他神经元的信息
- 也可以发送信息给其他神经元



- 神经元之间没有物理连接
- 两个“连接”的神经元之间留有20 纳米左右的缝隙
- 靠突触进行互联来传递信息
- 形成一个神经网络

- 单个神经元只有两种状态：兴奋和抑制
- 状态取决于从其他的神经细胞收到的输入信号量，以及突触的强度

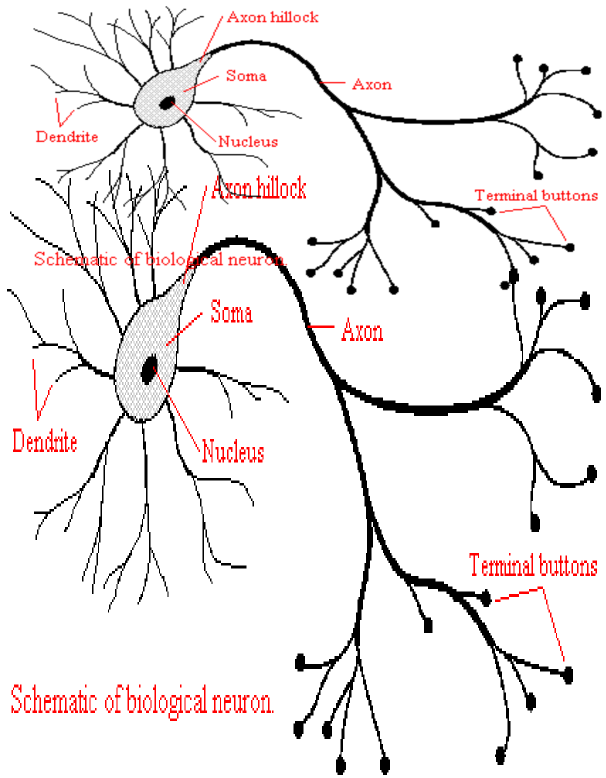
神经网络模型的基本组成

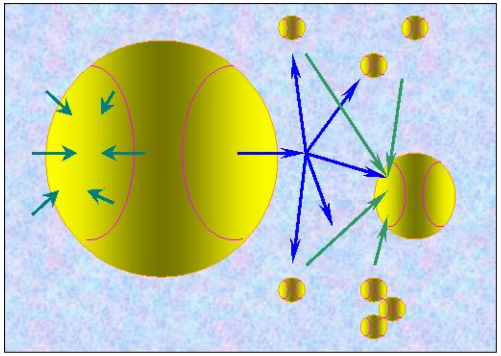
生物神经元的基本组成

- **细胞体**
- **突起**
  - 树突
  - 轴突

人工神经元的基本结构

- **处理单元**
- **连接**
  - 输入
  - 输出

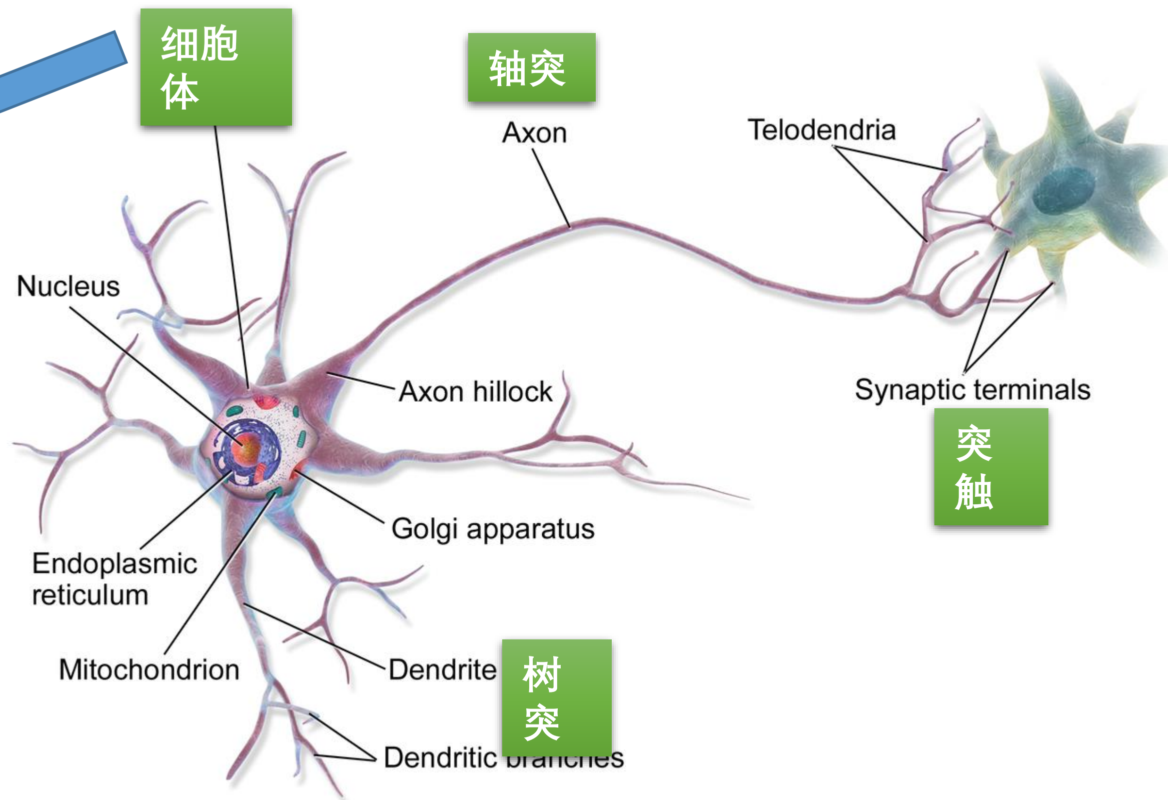




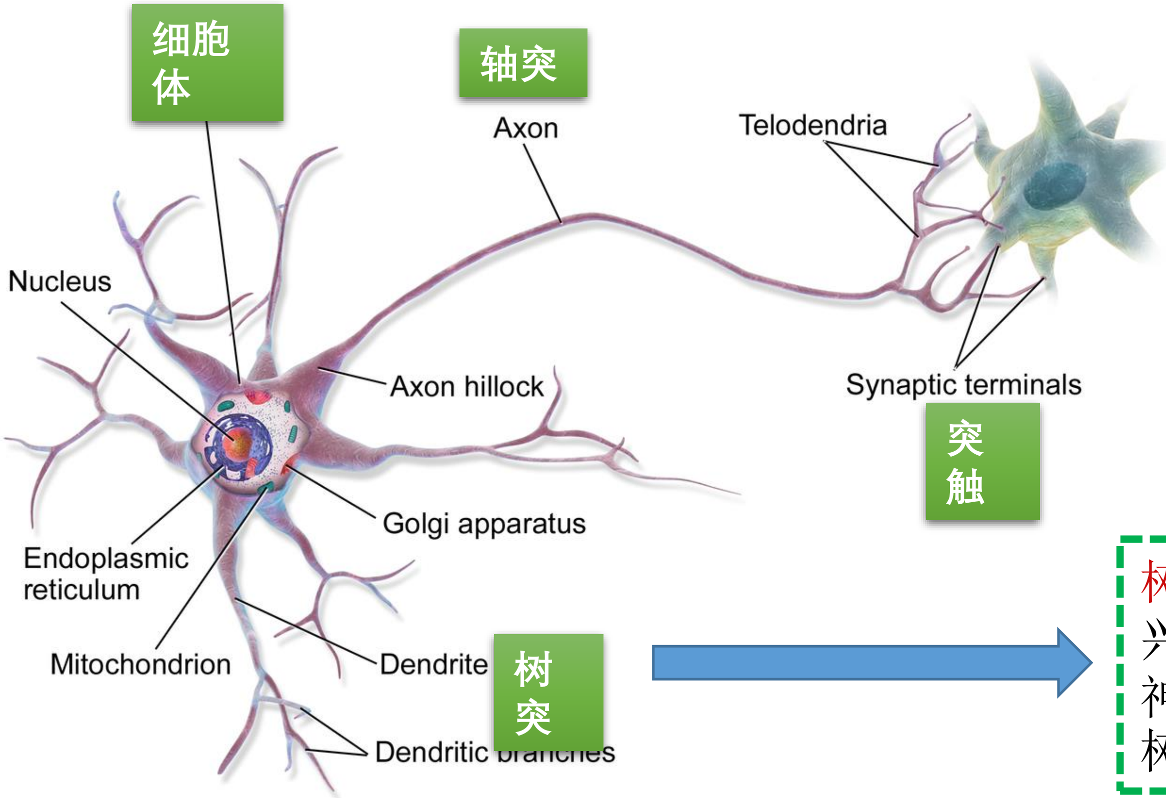
生物神经元结构

主体

细胞体中的神经细胞膜上有各种受体和离子通道，胞膜的受体可与相应的化学物质神经递质结合，引起离子通透性及膜内外电位差发生改变，产生相应的生理活动：兴奋或抑制.



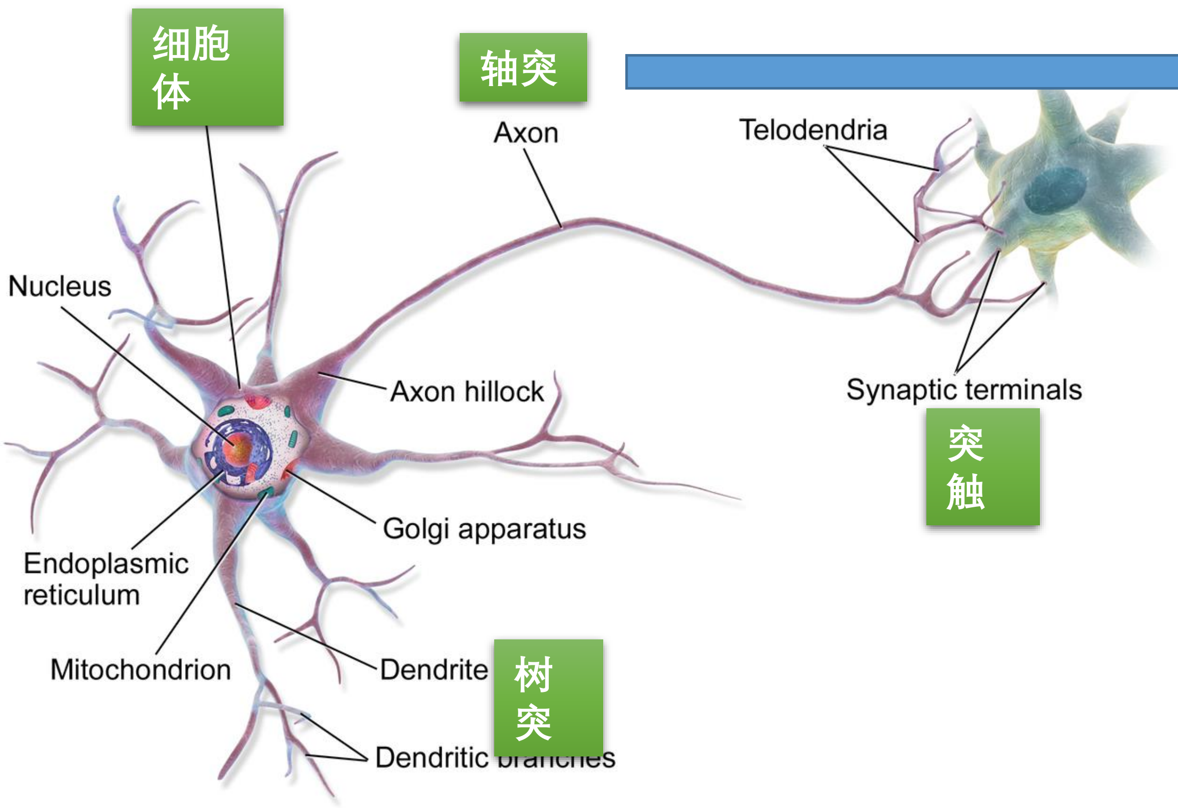
生物神经元结构



输入

树突可以接收刺激并将兴奋传入细胞体．每个神经元可以有一或多个树突．

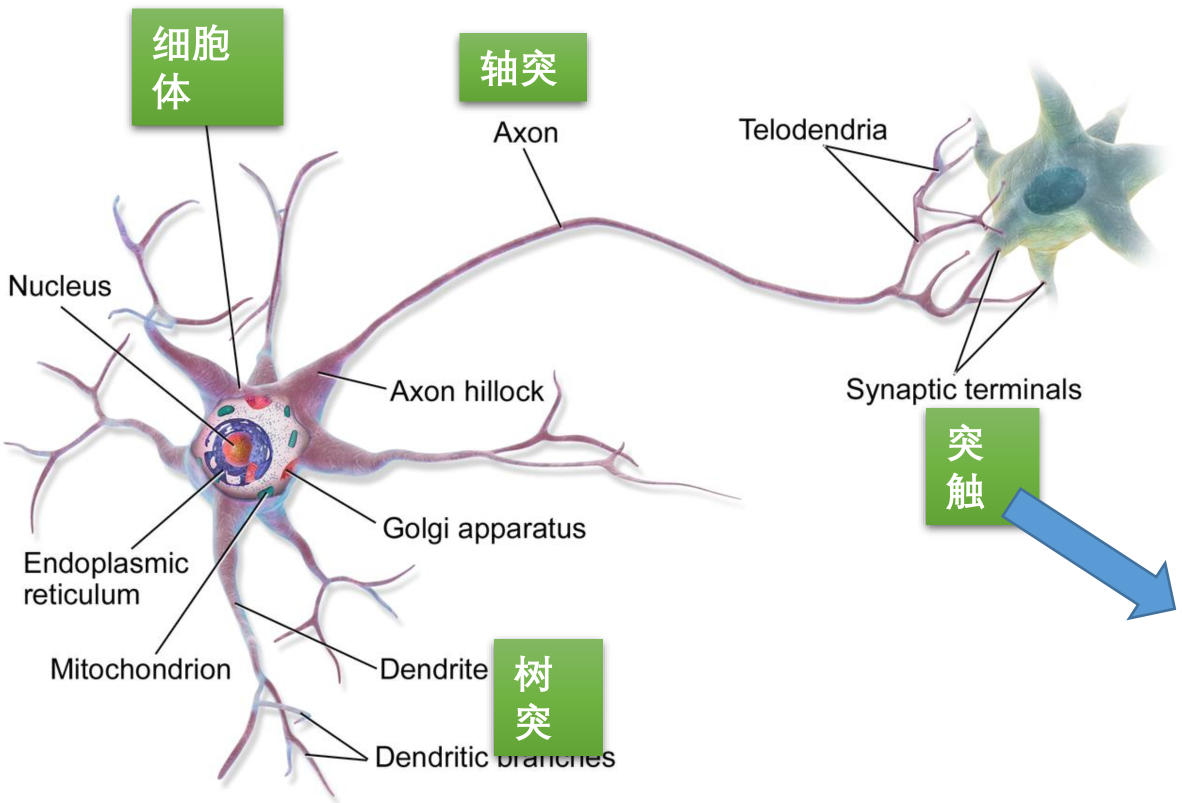
生物神经元结构



轴突可以把自身的兴奋状态从胞体传送到另一个神经元或其他组织．每个神经元只有一个轴突．

输出

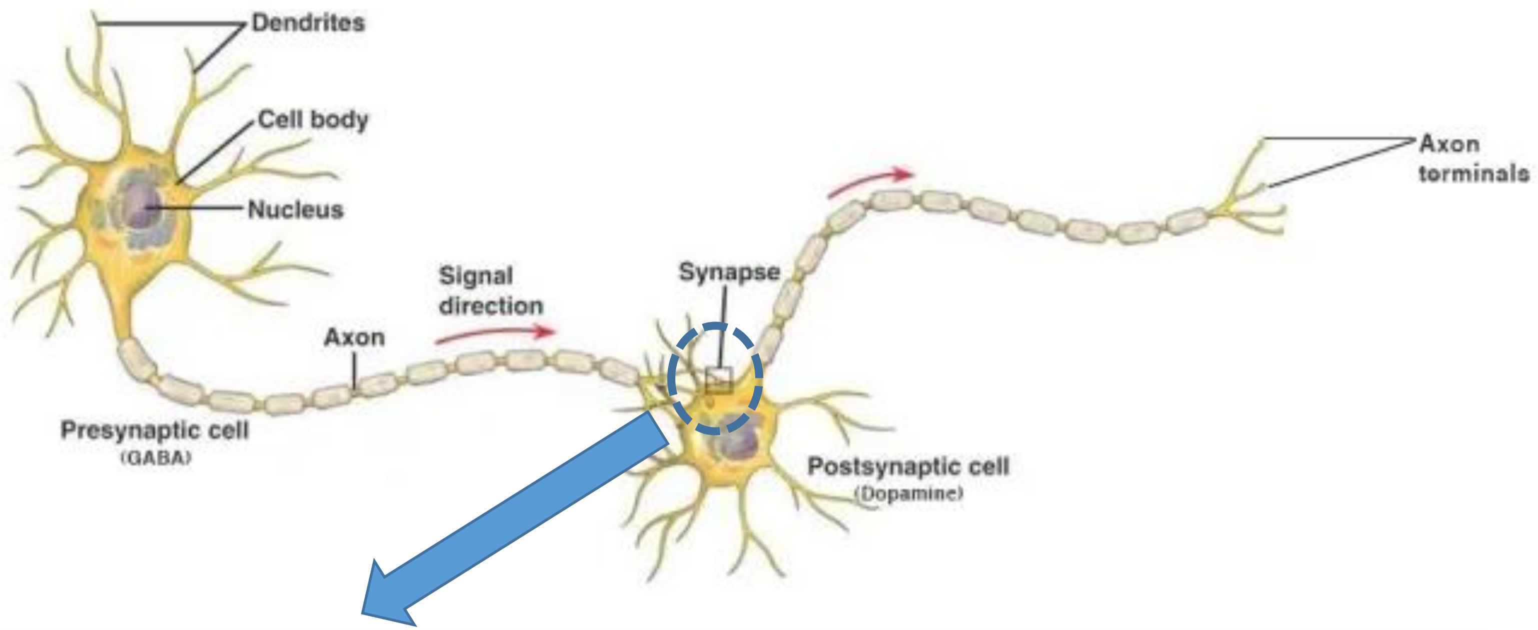
生物神经元结构



接口

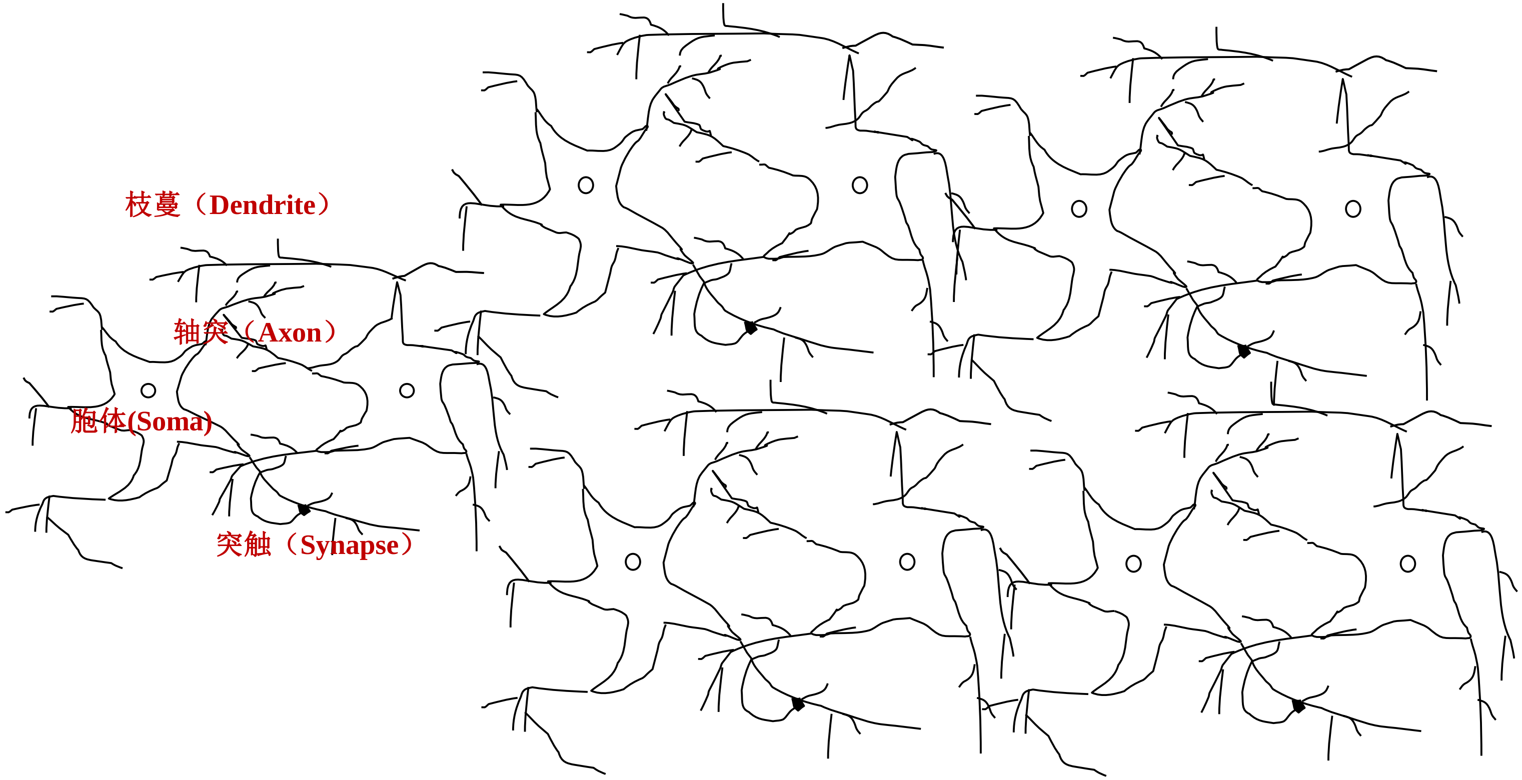
突触可以理解为神经元之间的连接“接口”，将一个神经元的兴奋状态传到另一个神经元.

两个生物神经元是如何"连接



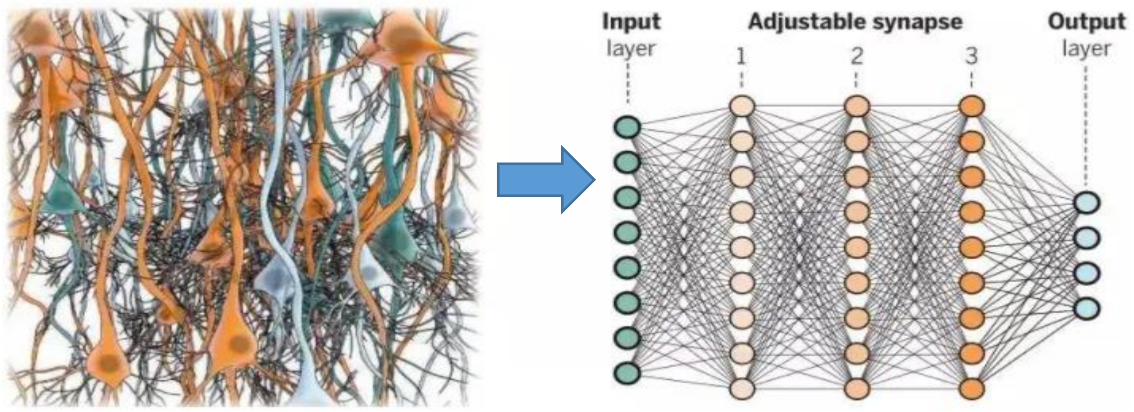
当信号量总和超过了某个阈值时，细胞体就会兴奋，产生电脉冲．电脉冲沿着轴突并通过突触传递到其他神经元

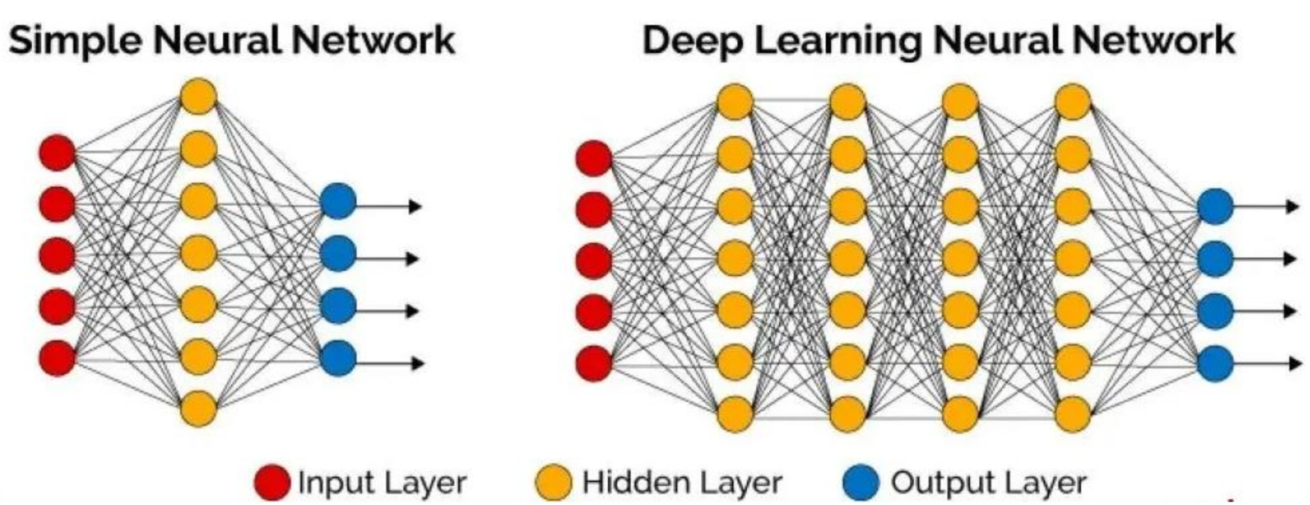
多个生物神经网连接



人工神经网络

➢ 人工神经网络的生物原型是拥有千亿数量级的生物神经元细胞的人脑生物神经网络，即人工神经网络是在“模仿”生物神经网络的基础上发展起来的机器学习算法。

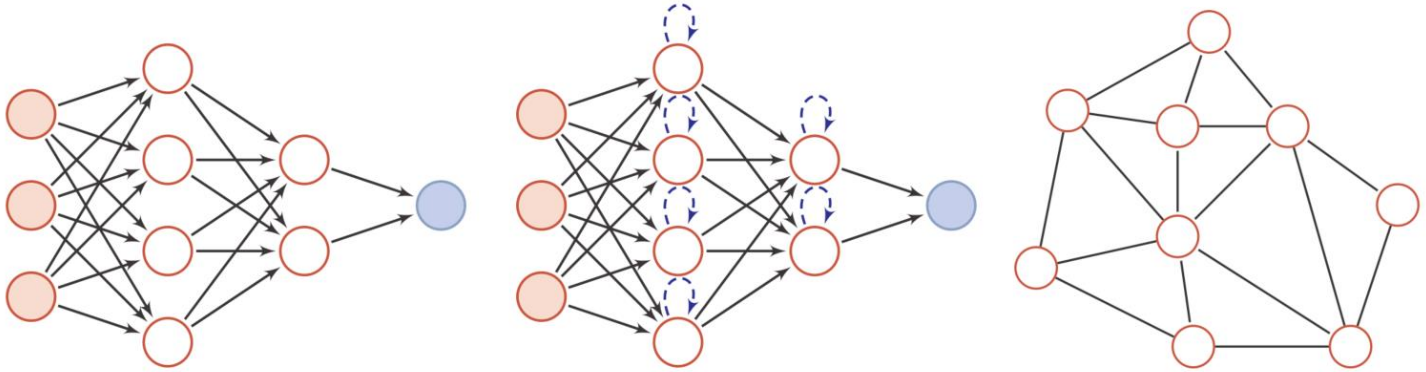




人工神经网络

人工神经网络主要由大量的神经元以及它们之间的有向连接构成。因此考虑三方面：

- 神经元的激活规则
  - 主要是指神经元输入到输出之间的映射关系，一般为非线性函数。
- 网络的拓扑结构
  - 不同神经元之间的连接关系。如单层、多层、前馈、反馈。
- 学习算法
  - 通过训练数据来学习神经网络的参数。如监督、无监督学习。



(a) 前馈网络　(b) 反馈网络　(c) 图网络

模拟单个神经元

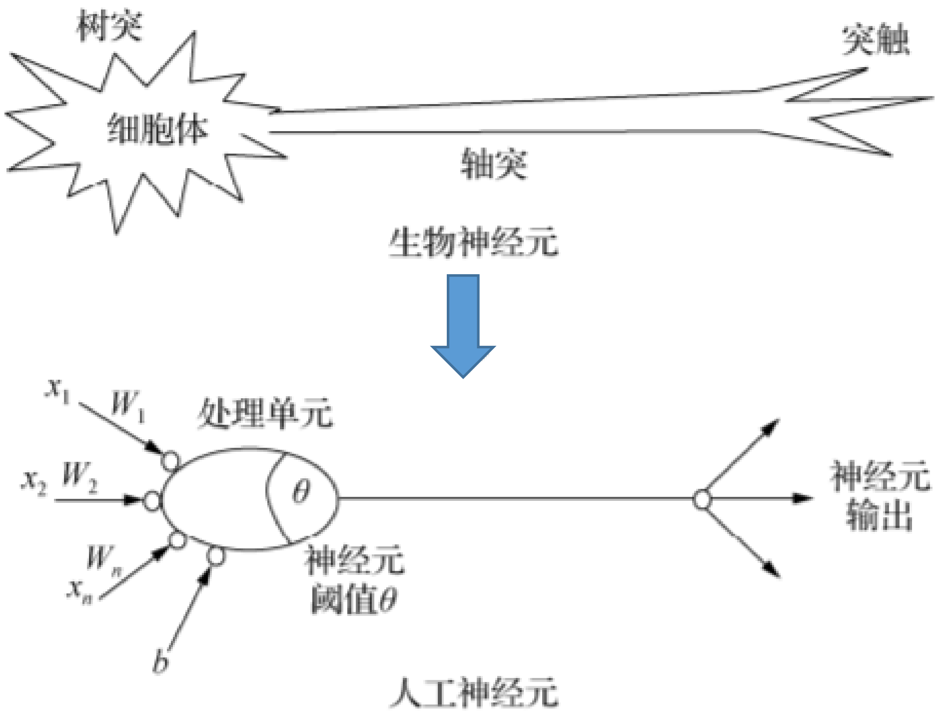


图 生物神经元演变成人工神经元

模拟单个神经元

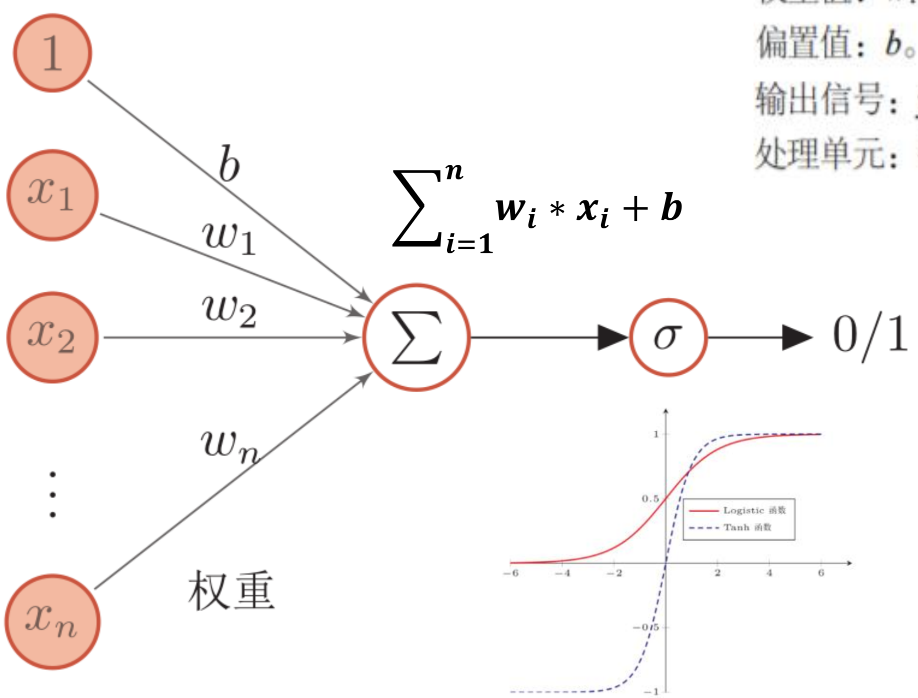
输入信号：x_1，x_2，…，x_n。

权重值：w_1，w_2，…，w_n。

偏置值：b。(类似线性方程的截距,可以平移直线,如分类,a是旋转角度)

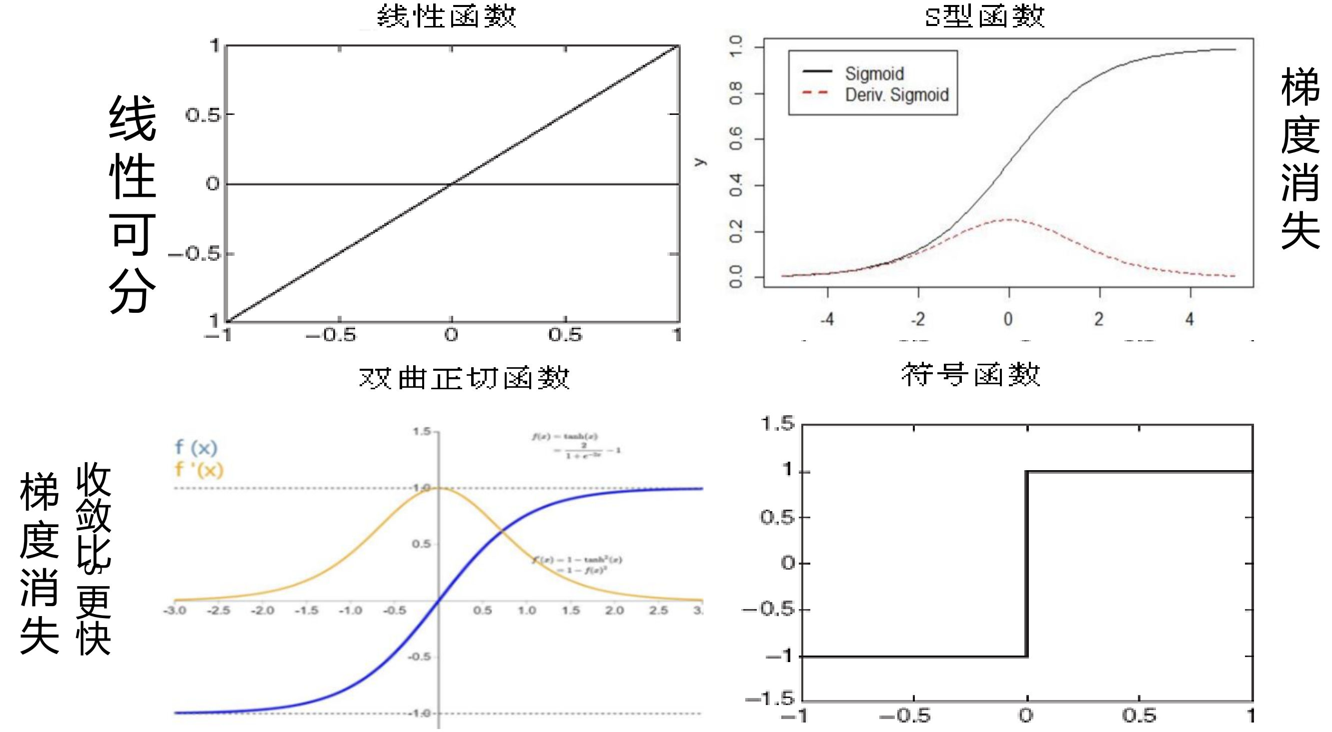
输出信号：$y = w_1 x_1 + w_2 x_2 + \ldots\ldots + w_n x_n + b$。

处理单元：每个输入 x_1，x_2…，x_n，都要乘上相应的权重值 w_1，w_2，…，w_n



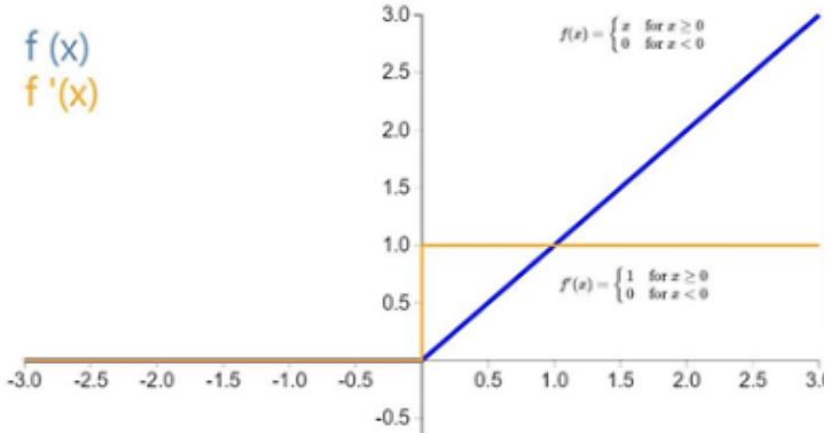
激活函数

激活函数可以是多种函数



梯度消失：如S型函数在[-5,5]外倒数接近0，梯度几乎不变，权值更新停滞，不利于寻优

激活函数



线性整流函数ReLU

x>0时，梯度为1，不存在梯度消失，而且没有复杂的梯度运算

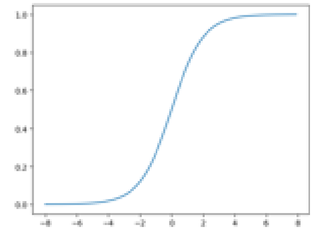
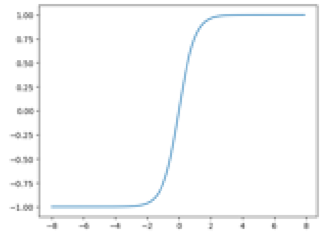
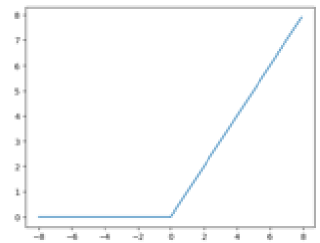
激活函数

解决了部分梯度消失问题

激活函数	函数	导数
Logistic 函数	$f(x)=\frac{1}{1+\exp(-x)}$	$f'(x)=f(x)(1-f(x))$
Tanh 函数	$f(x)=\frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$	$f'(x)=1-f(x)^2$
ReLU	$f(x)=\max(0,x)$	$f'(x)=I(x>0)$
ELU	$f(x)=\max(0,x)+\min\left(0,\gamma(\exp(x)-1)\right)$	$f'(x)=I(x>0)+I(x\leq 0)\cdot\gamma\exp(x)$
SoftPlus 函数	$f(x)=\log\left(1+\exp(x)\right)$	$f'(x)=\frac{1}{1+\exp(-x)}$

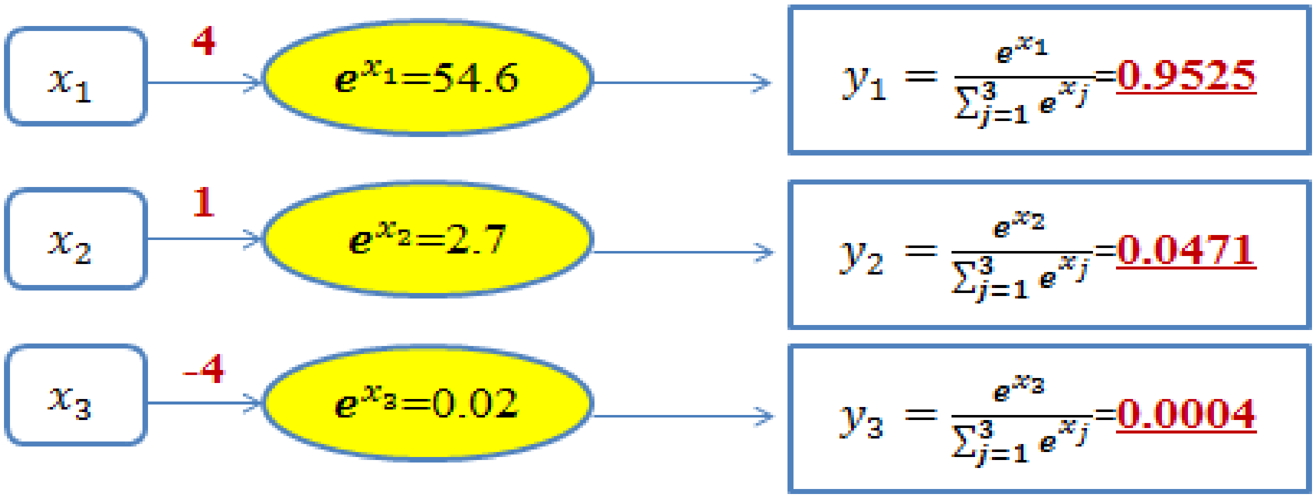
收敛速度比双面正切更快

表6.1 激活函数基本性质

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

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

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



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

- 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。

损失函数

损失函数（Loss Function）又称为代价函数（Cost Function），用来计算模型预测值与真实值之间的误差。损失函数是神经网络设计中的一个重要组成部分。通过定义与任务相关的良好损失函数，在训练过程中可根据损失函数来计算神经网络的误差大小，进而优化神经网络参数。

两种最常用损失函数：

- 均方误差损失函数
- 交叉熵损失函数

均方误差损失函数

均方误差损失函数通过计算预测值和实际值之间距离（即误差）的平方来衡量模型优劣。假设有n个训练数据x_i，每个训练数据x_i的真实输出为y_i，模型对x_i的预测值为$\hat{y}_i$。该模型在n个训练数据下所产生的均方误差损失可定义如下：

$$\mathrm{MSE}=\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

交叉熵损失函数

交叉熵（cross entropy）是信息论中的重要概念，主要用来度量两个概率分布间的差异。假定p和q是数据x的两个概率分布，通过q来表示p的交叉熵可如下计算：

$$\mathrm{H}(p,q) = -\sum_{x} p(x) * log q(x)$$

交叉熵刻画了两个概率分布之间的距离，旨在描绘通过概率分布q来表达概率分布p的困难程度。根据公式不难理解，交叉熵越小，两个概率分布p和q越接近。

对于每一个输入数据，神经网络所预测类别分布概率与实际类别分布概率之间的差距越小越好，即交叉熵越小越好。于是，可将交叉熵作为损失函数来训练神经网络。

交叉熵损失函数

假设数据x属于类别1。记数据x的类别分布概率为y，显然$y=(1,0,0)$代表数据x的实际类别分布概率。记$\hat{y}$代表模型预测所得类别分布概率。

那么对于数据x而言，其实际类别分布概率y和模型预测类别分布概率$\hat{y}$的交叉熵损失函数定义为：

$$\text{cross entropy} = -y \times \log(\hat{y})$$

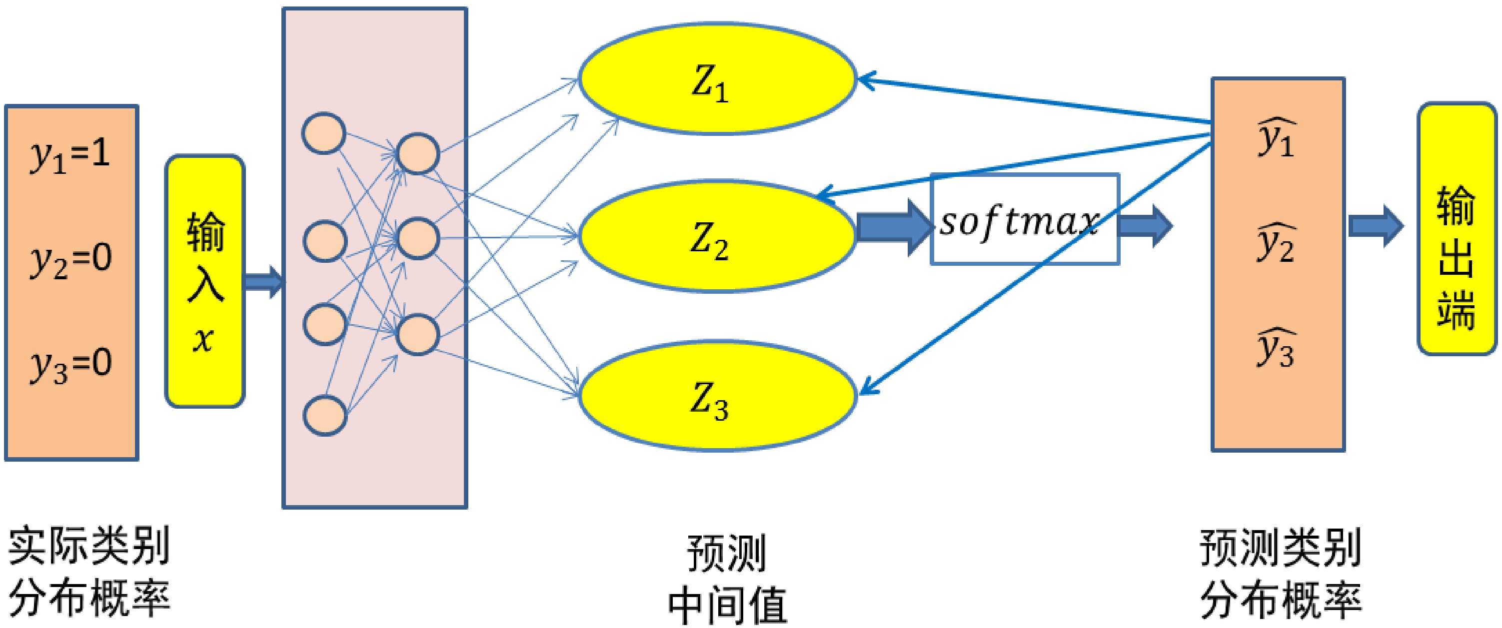


图6.3 三类分类问题中输入x的交叉熵损失示意图（x属于第一类）

交叉熵损失函数

图6.3给出了一个三个类别分类的例子。由于输入数据x属于类别1，因此其实际类别概率分布值为$\mathrm{y} = (y_1, y_2, y_3) = (1,0,0)$。经过神经网络的变换，得到了输入数据$x$相对于三个类别的预测中间值 (z_1, z_2, z_3)。然后，经过Softmax函数映射，得到神经网络所预测的输入数据x的类别分布概率$\hat{y} = (\widehat{y_1}, \widehat{y_2}, \widehat{y_3})$。根据前面的介绍，$\widehat{y_1}$、$\widehat{y_2}$和$\widehat{y_3}$为$(0,1)$范围之间的一个概率值。由于样本$x$属于第一个类别，因此希望神经网络所预测得到的$\widehat{y_1}$取值要远远大于$\widehat{y_2}$和$\widehat{y_3}$的取值。

交叉熵损失函数

训练中可利用如下交叉熵损失函数来对模型参数进行优化：

$$\text{cross entropy} = -(y_1 \times \log(\widehat{y_1}) + y_2 \times \log(\widehat{y_2}) + y_3 \times \log(\widehat{y_3}))$$

在上式中，y_2和y_3均为$\mathbf{0}$、y_1为$\mathbf{1}$，因此交叉熵损失函数简化为：$-y_1 \times \log(\widehat{y_1}) = -\log(\widehat{y_1})$。

在神经网络训练中，要将输入数据实际的类别概率分布与模型预测的类别概率分布之间的误差（即损失）从输出端向输入端传递，以便来优化模型参数。下面简单介绍根据交叉熵计算得到的误差从$\widehat{y_1}$传递给z_1和z_2（z_3的推导与z_2相同）的情况。

交叉熵损失函数

$$\frac{\partial \widehat{y_1}}{\partial z_1} = \frac{\partial}{\partial z_1}\left(\frac{e^{z_1}}{\sum_k e^{z_k}}\right) = \frac{(e^{z_1})' \times \sum_k e^{z_k} - e^{z_1} \times e^{z_1}}{(\sum_k e^{z_k})^2}$$

$$= \frac{e^{z_1}}{\sum_k e^{z_k}} - \frac{e^{z_1}}{\sum_k e^{z_k}} \times \frac{e^{z_1}}{\sum_k e^{z_k}} = \widehat{y_1}(1 - \widehat{y_1})$$

由于交叉熵损失函数$-\log(\widehat{y_1})$对$\widehat{y_1}$求导的结果为$-\frac{1}{\widehat{y_1}}$，$\widehat{y_1}(1-\widehat{y_1})$与$-\frac{1}{\widehat{y_1}}$相乘为$\widehat{y_1}-1$。这说明一旦得到模型预测输出$\widehat{y_1}$，将该输出减去1就是交叉损失函数相对于$z_1$的偏导结果。

$$\frac{\partial \widehat{y_1}}{\partial z_2} = \frac{\partial}{\partial z_2}\left(\frac{e^{z_1}}{\sum_k e^{z_k}}\right) = \frac{0 \times \sum_k e^{z_k} - e^{z_1} \times e^{z_2}}{(\sum_k e^{z_k})^2} = -\frac{e^{z_1}}{\sum_k e^{z_k}} \times \frac{e^{z_2}}{\sum_k e^{z_k}} = -\widehat{y_1}\widehat{y_2}$$

同理，交叉熵损失函数导数为$-\frac{1}{\widehat{y_1}}$，$-\widehat{y_1}\widehat{y_2}$与$-\frac{1}{\widehat{y_1}}$相乘结果为$\widehat{y_2}$。这意味对于除第一个输出节点以外的节点进行偏导，在得到模型预测输出后，只要将其保存，就是交叉损失函数相对于其他节点的偏导结果。在z_1、z_2和z_3得到偏导结果后，再通过链式法则（后续介绍）将损失误差继续往输入端传递即可。

交叉熵损失函数

- 在上面的例子中，假设所预测中间值(z_1, z_2, z_3)经过Softmax映射后所得结果为(0.34,0.46,0.20)。由于已知输入数据x属于第一类，显然这个输出不理想而需要对模型参数进行优化。如果选择交叉熵损失函数来优化模型，则 (z_1, z_2, z_3)这一层的偏导值为 $(0.34-1, 0.46, 0.20) = (-0.66, 0.46, 0.20)$。

- 可以看出，Softmax和交叉熵损失函数相互结合，为偏导计算带来了极大便利。偏导计算使得损失误差从输出端向输入端传递，来对模型参数进行优化。在这里，交叉熵与Softmax函数结合在一起，因此也叫Softmax损失（Softmax with cross-entropy loss）。

提纲

一、深度学习历史发展

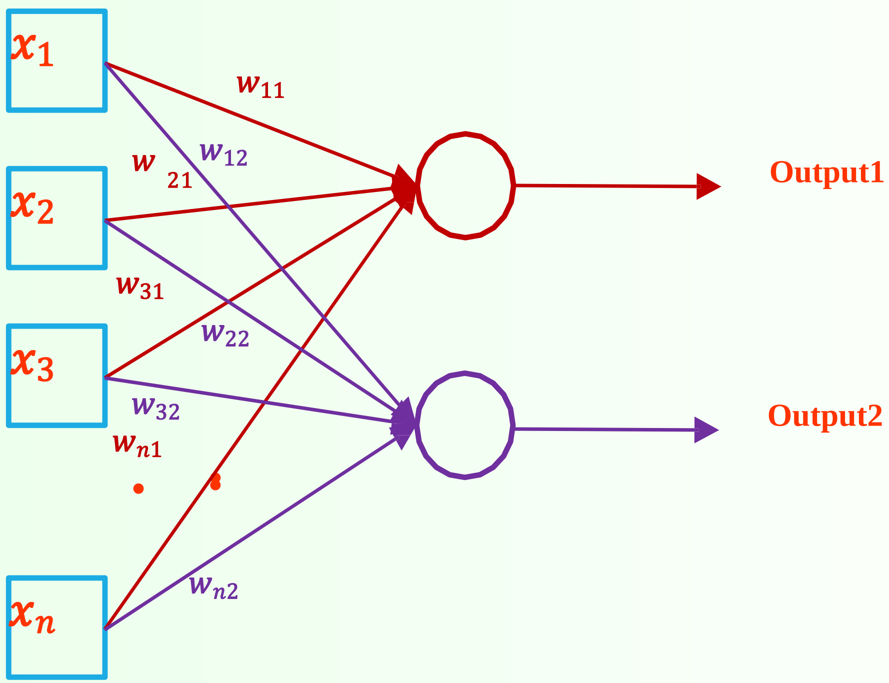
二、前馈神经网络

三、卷积神经网络

四、循环神经网络

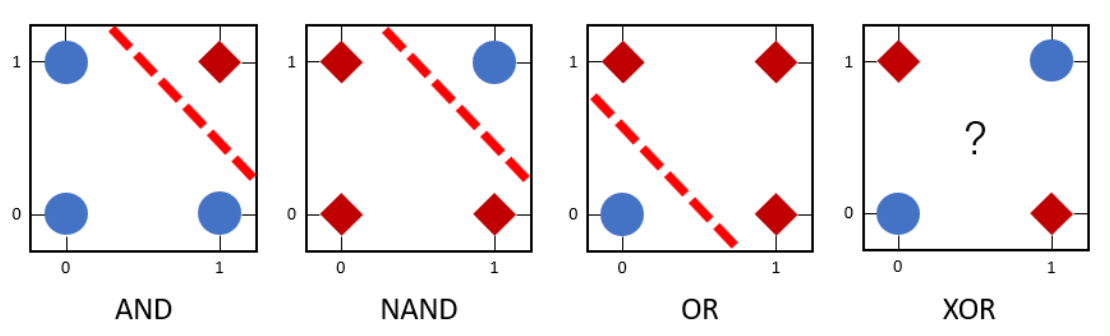
感知机模型：单层感知机

早期的感知机结构和MCP模型相似，由一个输入层和一个输出层构成，因此也被称为“单层感知机”。感知机的输入层负责接收实数值的输入向量，输出层则能输出1或-1两个值。



感知机模型：单层感知机

单层感知机可被用来区分线性可分数据。在图6.5中，逻辑与(AND)、逻辑与非(NAND)和逻辑或(OR)为线性可分函数，所以可利用单层感知机来模拟这些逻辑函数。但是，由于逻辑异或（XOR）是非线性可分的逻辑函数，因此单层感知机无法模拟逻辑异或函数的功能。

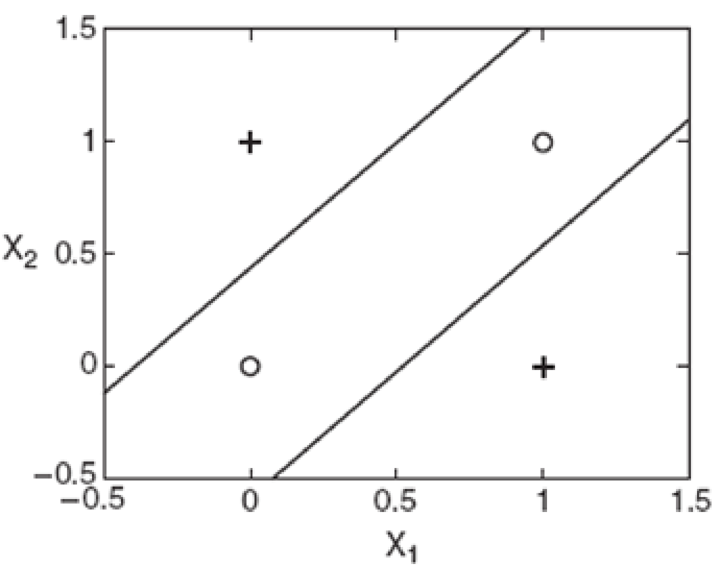


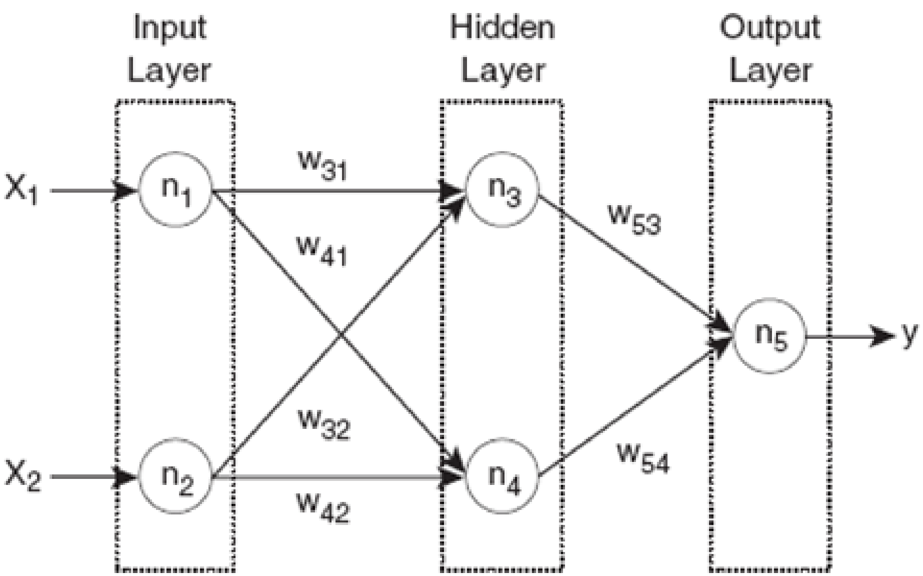
- F. Rosenblatt证明了对“线性可分”的问题
- Marvin Minsky，证明了感知器无法执行“异或问题”

图6.5 单层感知机模拟不同逻辑函数功能的示意图

多层人工神经网络-解决"异或"问题

- 例：用两层前馈神经网络解决异或问题



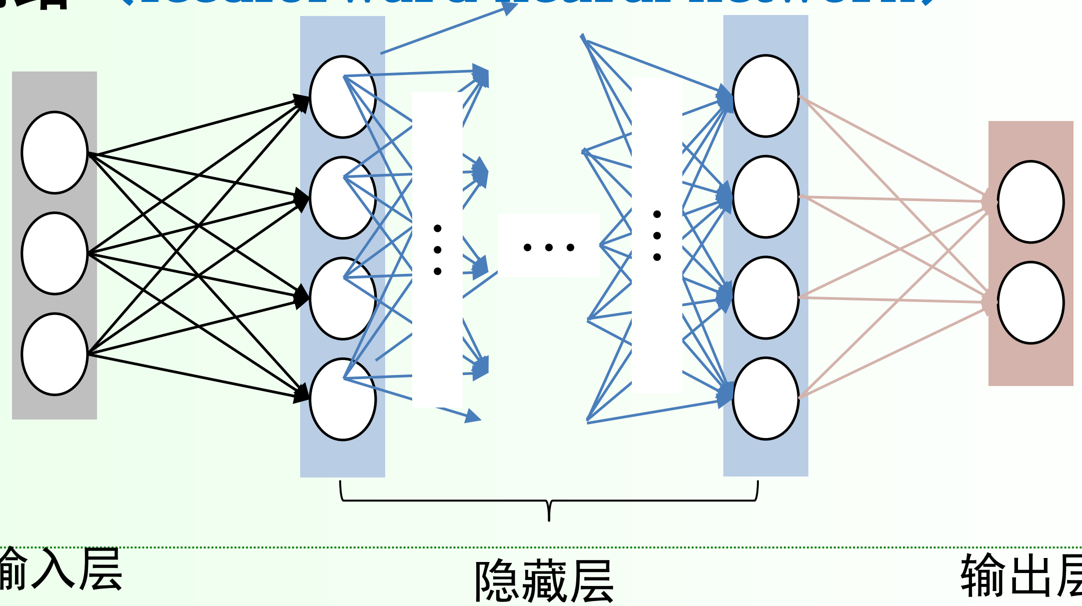


决策边界

神经网络拓扑结构

感知机模型：多层感知机

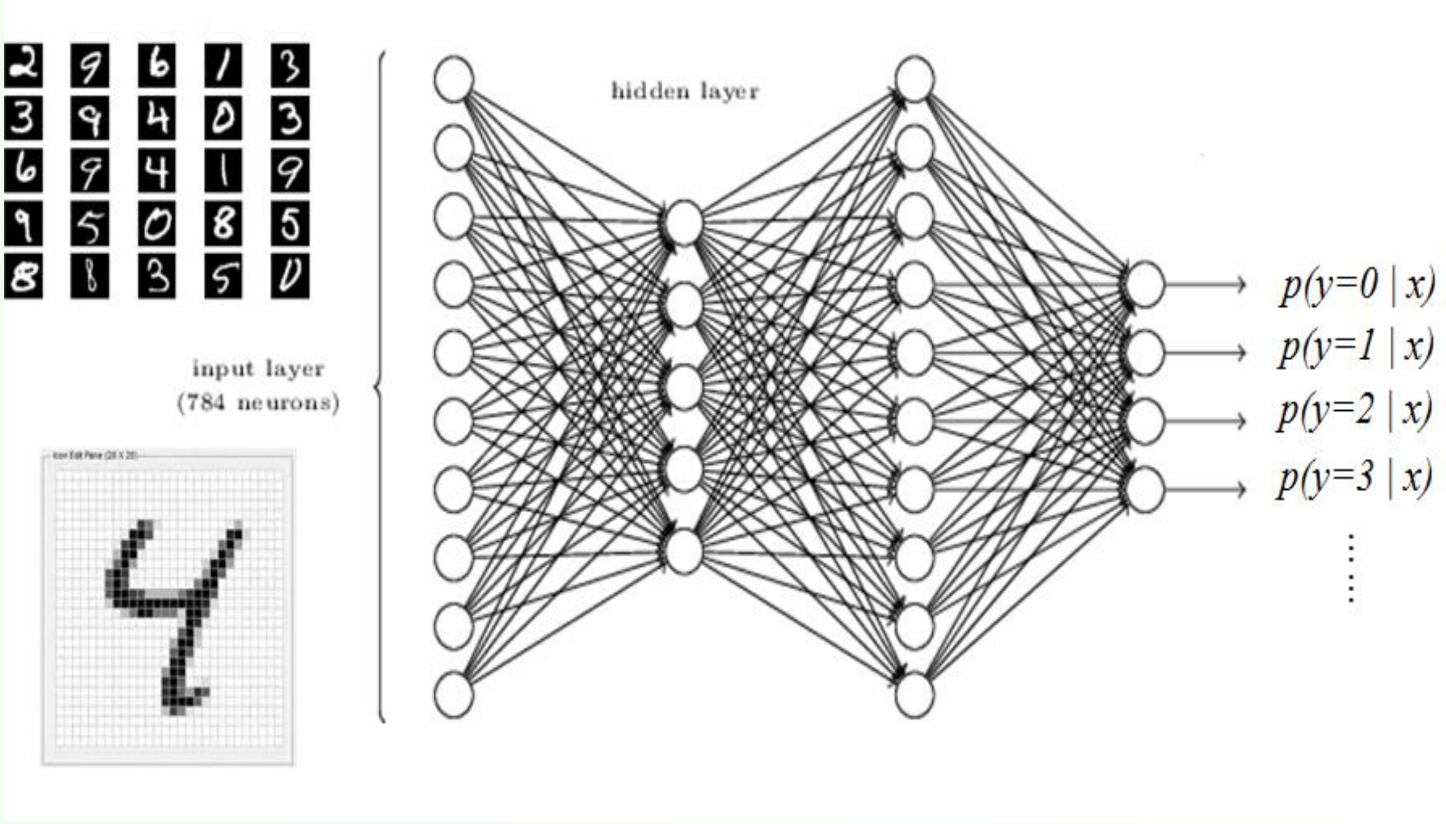
- 多层感知机由输入层、输出层和至少一层的隐藏层构成。网络中各个隐藏层中神经元可接收相邻前序隐藏层中所有神经元传递而来的信息，经过加工处理后将信息输出给相邻后续隐藏层中所有神经元。
- 各个神经元接受前一级的输入，并输出到下一级，模型中没有反馈
- 层与层之间通过“全连接”进行链接，即两个相邻层之间的神经元完全成对连接，但层内的神经元不相互连接。
- **前馈神经网络**（**feedforward neural network**）



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



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：

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

- 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，因此是损失函数求解的方向

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
  - 泰勒展开 (Δx是微小的增量):

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

多元泰勒展开:

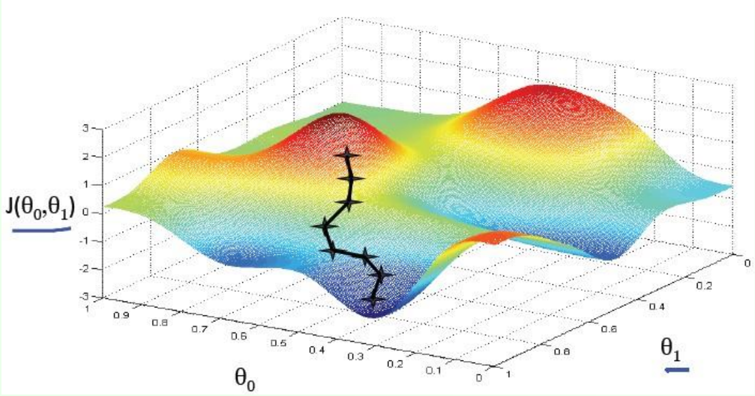
$$f(x+\Delta x)-f(x)\approx(\nabla f(x))^T\Delta x$$

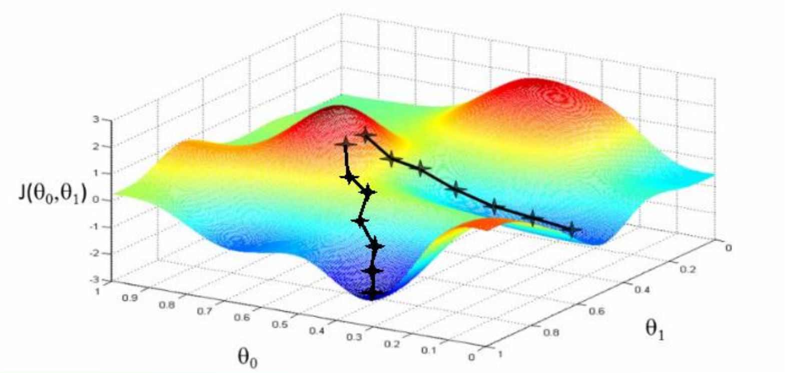
- 最小化损失函数$f(x)$，则$f(x+\Delta x)<f(x)$，即 $(\nabla f(x))^T\Delta x<0$
- 在 $(\nabla f(x))^T\Delta x=\|\nabla f(x)\|\|\Delta x\|\cos\theta$中，$\|\nabla f(x)\|$和$\|\Delta x\|$分别为损失函数梯度的模和下一轮迭代中$x$取值增量的模，两者均为正数。为了保证损失误差减少，只要保证$\cos\theta<0$。当$\theta=180°$时，$\cos\theta=-1$，这时 $(\nabla f(x))^T\Delta x$取到最小。

梯度下降（Gradient Descent)

当 $\theta = 180°$ 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$\Delta x$，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）。

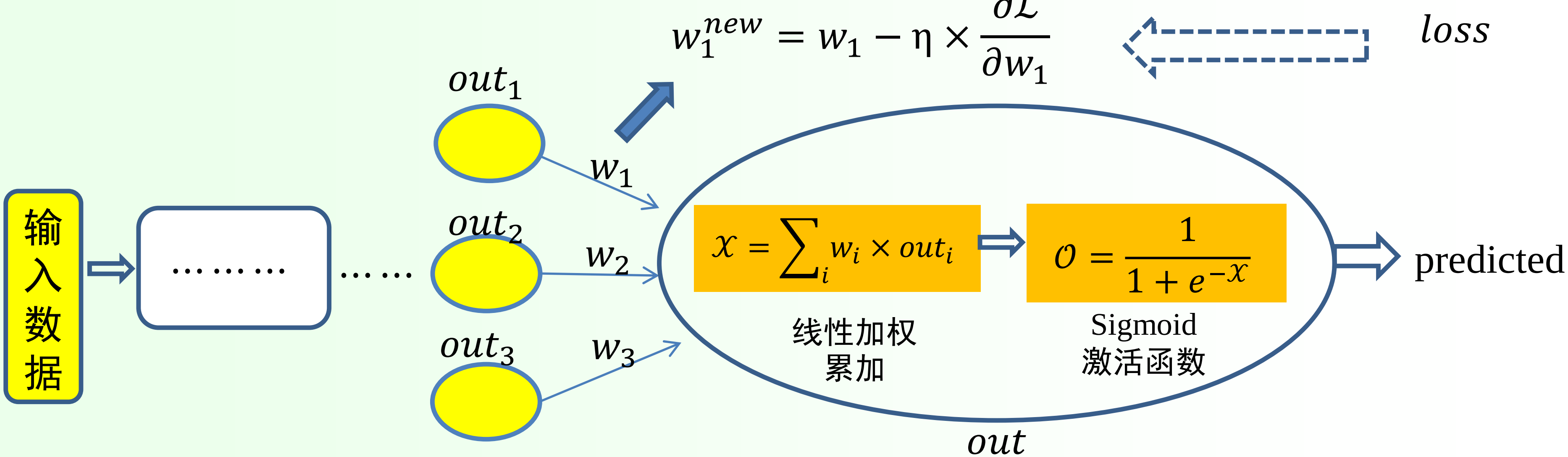




参数优化：误差反向传播 (error back propagation, BP)

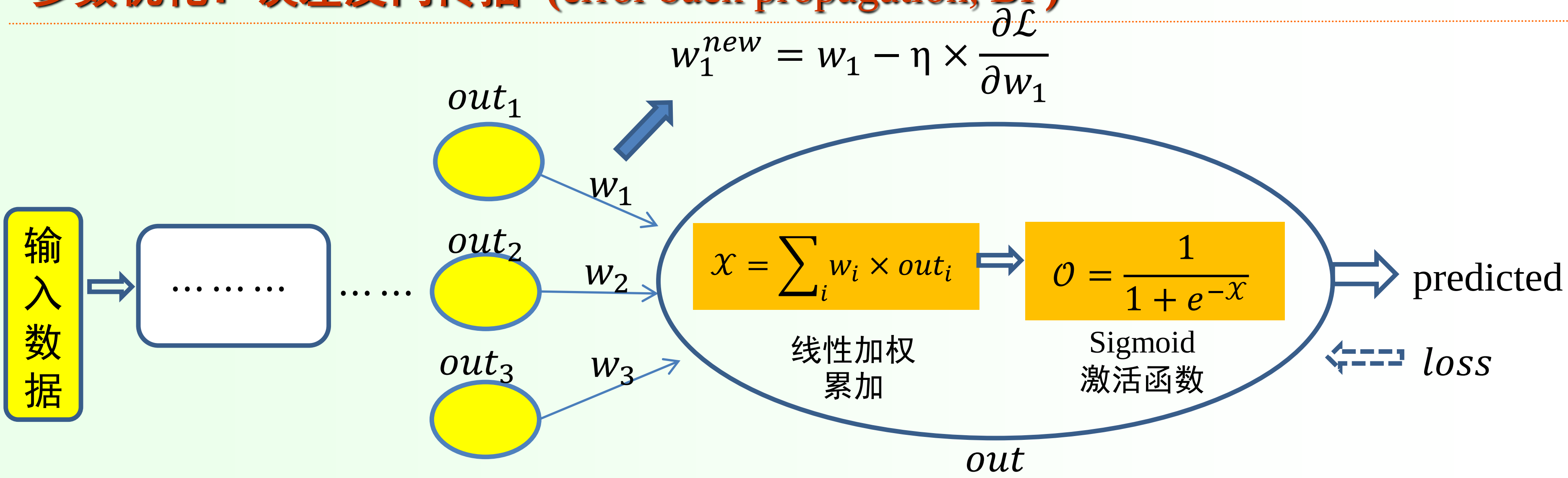
- **BP算法: 将输出层误差反向传播给隐藏层进行参数更新的方法。**
- **将误差从后向前传递，将误差分摊给各层所有单元，从而获得各层单元所产生的误差，进而依据这个误差来让各层单元负起各自责任、修正各单元参数。**

$$w_1^{new} = w_1 - \eta \times \frac{\partial \mathcal{L}}{\partial w_1}$$



$$loss = \mathcal{L}\,(predicted, groundtruth)$$

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

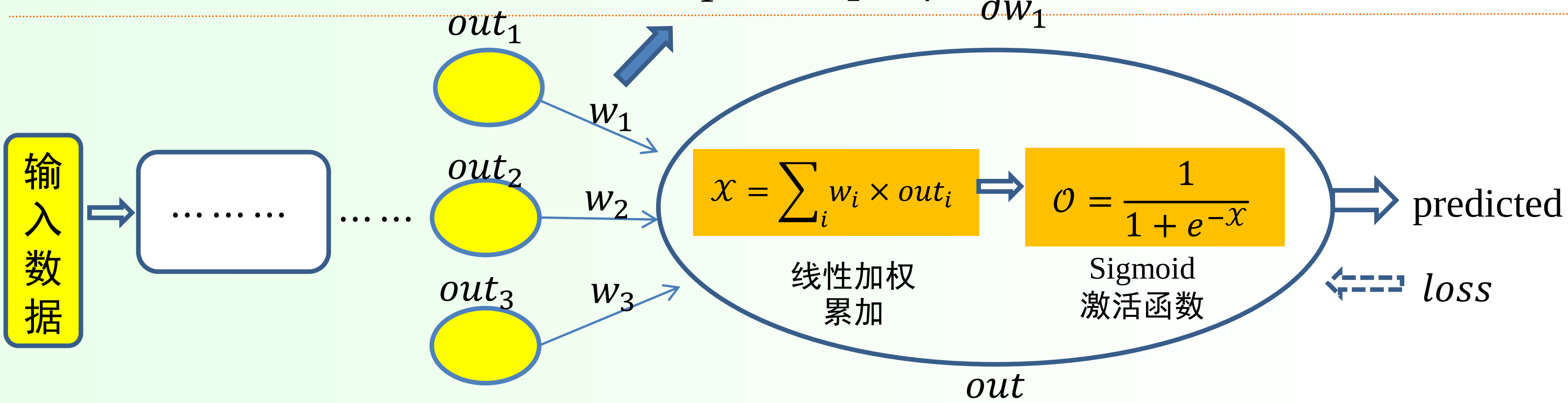


$loss = \mathcal{L}\,(predicted, groundtruth)$

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。

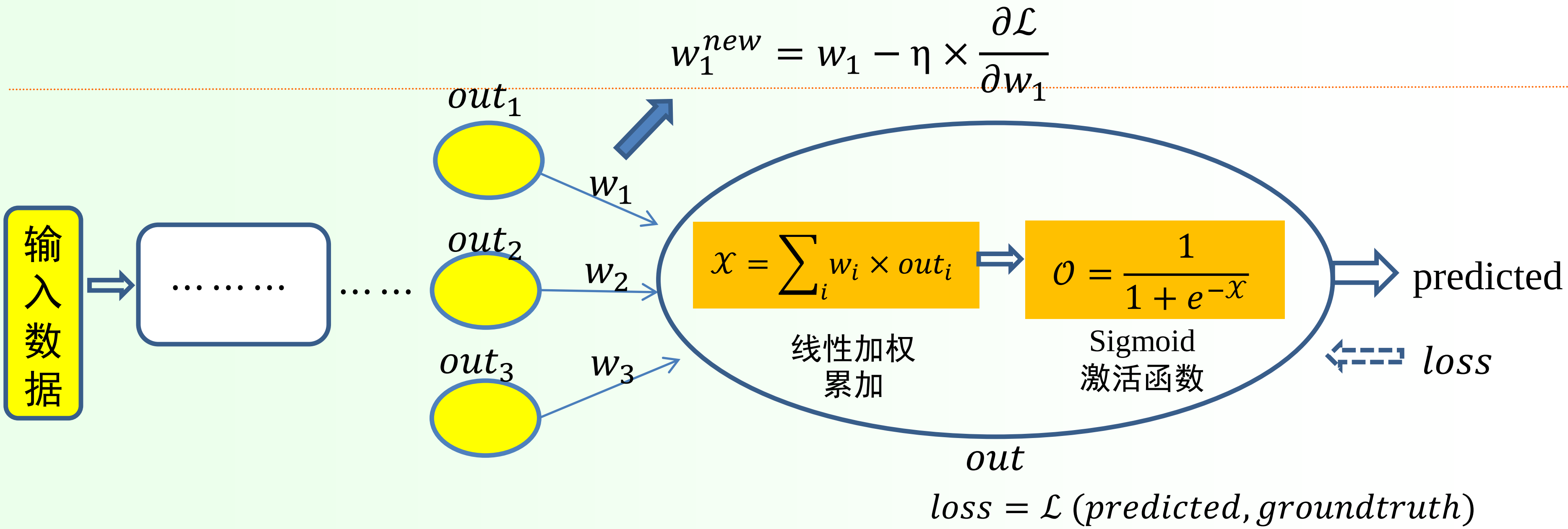
即将w_1变为$w_1 - \eta\frac{\partial \mathcal{L}}{\partial w_1}$后，能使得损失误差减少。

$$w_1^{new} = w_1 - \eta \times \frac{\partial \mathcal{L}}{\partial w_1}$$



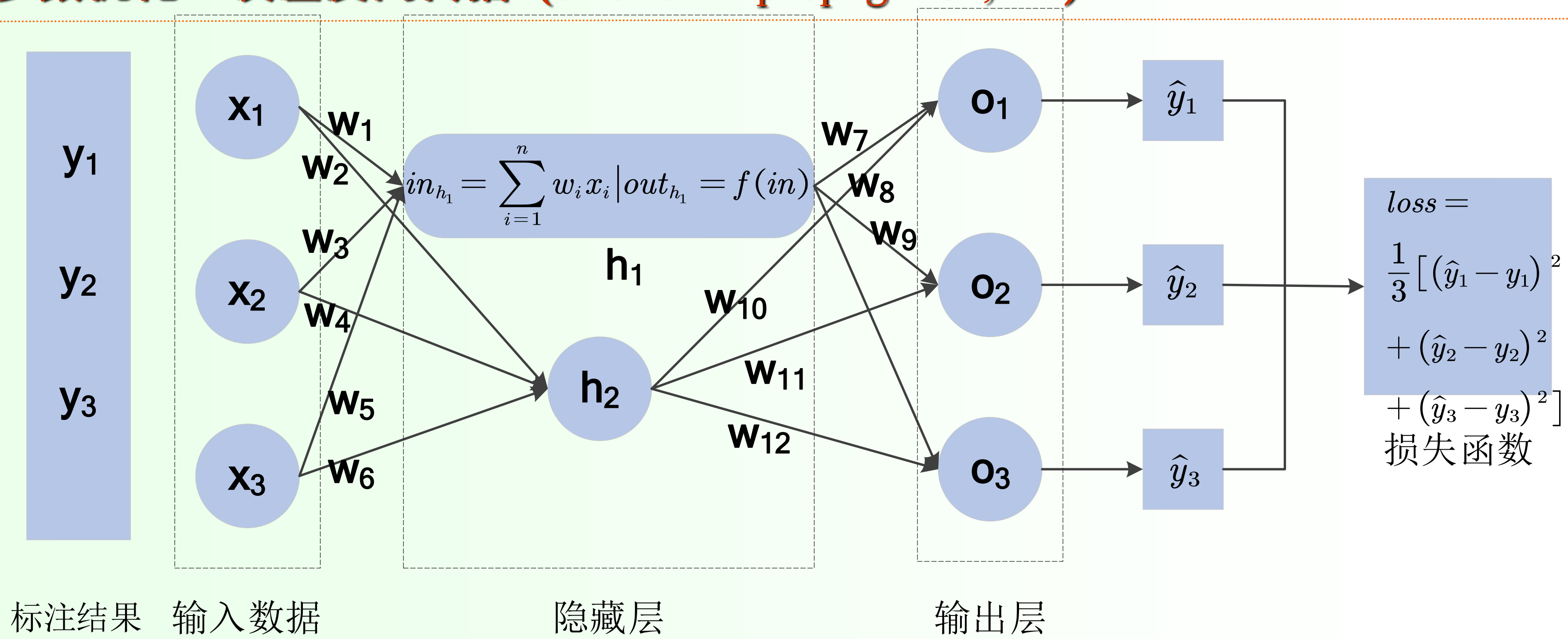
$$loss = \mathcal{L}\,(predicted, groundtruth)$$

- **由于w_1与加权累加函数$\mathcal{X}$和sigmoid函数均有关，因此$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial \mathcal{O}}\frac{\partial \mathcal{O}}{\partial \mathcal{X}}\frac{\partial \mathcal{X}}{\partial w_1}$。在这个链式求导中：$\frac{\partial \mathcal{L}}{\partial \mathcal{O}}$与损失函数的定义有关；$\frac{d\mathcal{O}}{d\mathcal{X}}$是对sigmoid函数求导，结果为$\frac{1}{1+e^{-\mathcal{X}}} \times (1 - \frac{1}{1+e^{-\mathcal{X}}})$；$\frac{\partial \mathcal{X}}{\partial w_1}$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。**



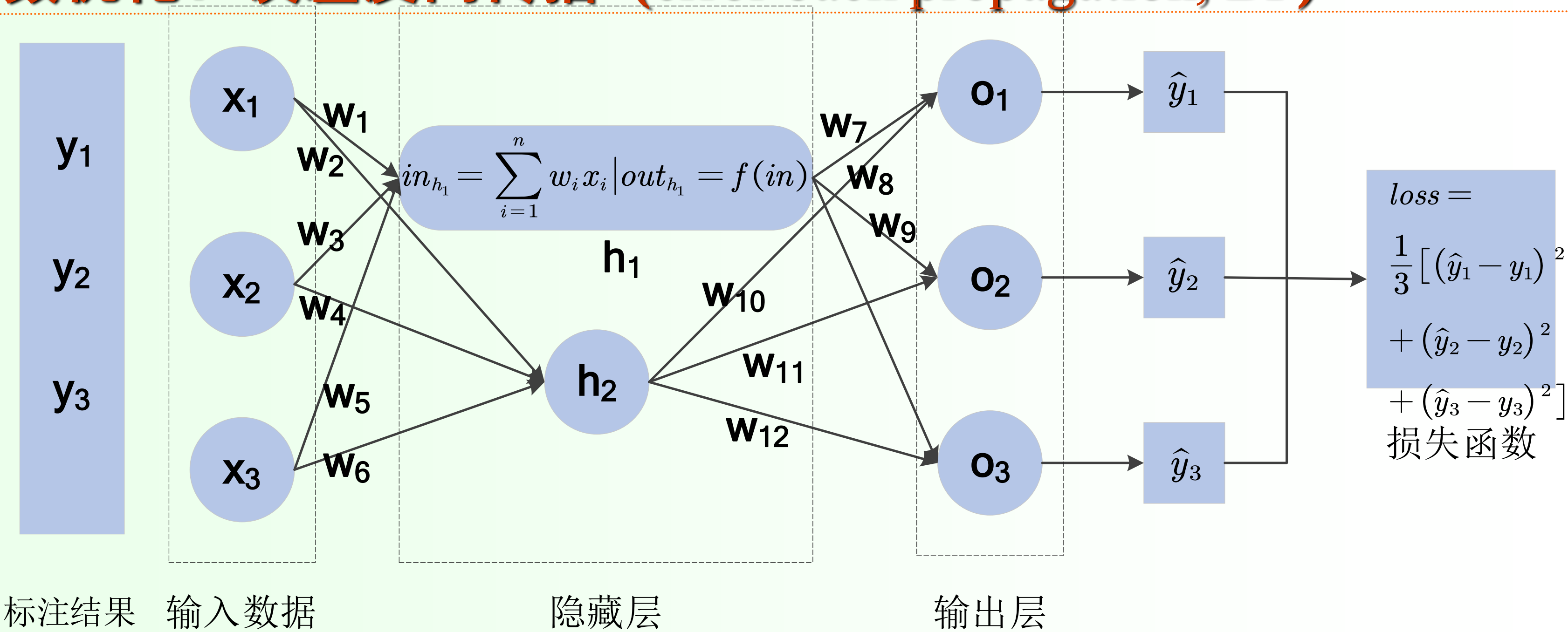
- 参数w_1在下一轮迭代中的取值被调整为：$w_1^{new} = w_1 - \eta \times \frac{\partial \mathcal{L}}{\partial w_1} = w_1 - \eta \times \frac{\partial \mathcal{L}}{\partial O}\frac{\partial O}{\partial x}\frac{\partial x}{\partial w_1} = w_1 - \eta \times \frac{\partial \mathcal{L}}{\partial O} \times \frac{1}{1+e^{-x}} \times (1 - \frac{1}{1+e^{-x}}) \times out_1$。
- 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)



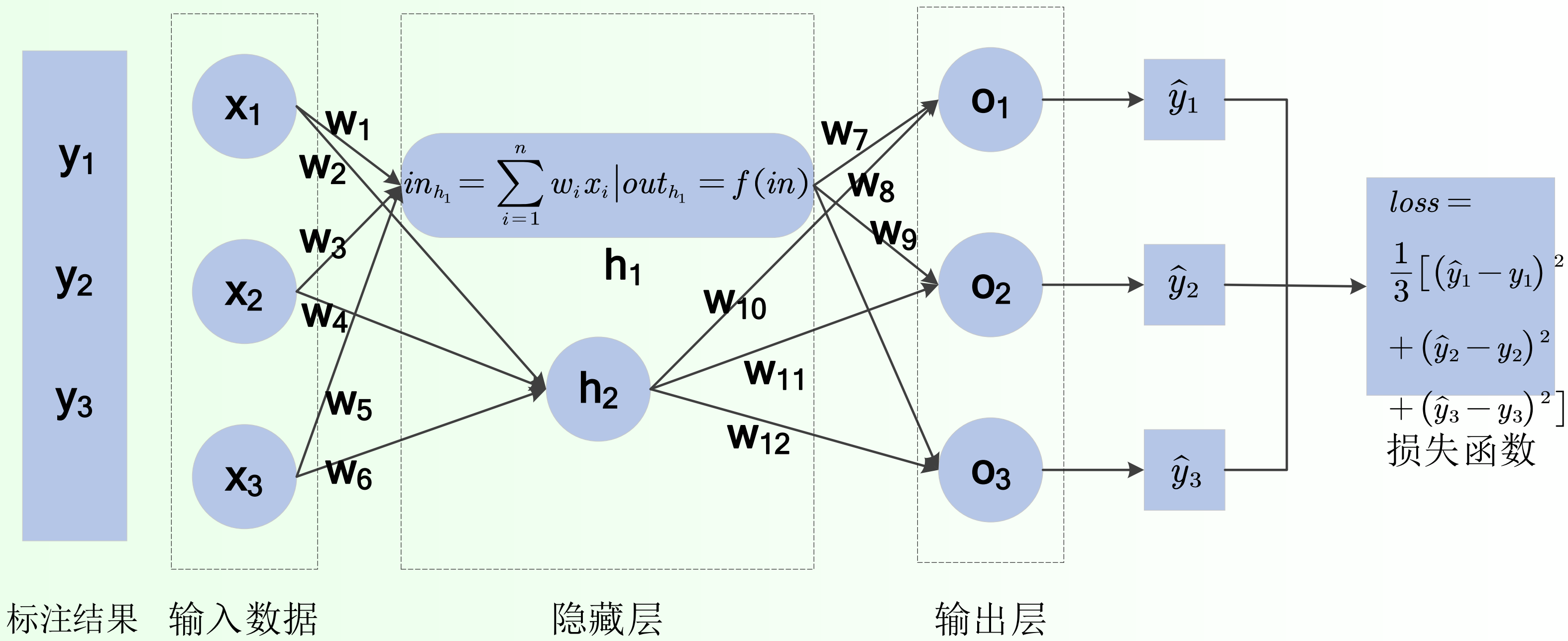
- 通过一个三类分类的具体例子来介绍神经网络中参数更新过程。给定一个包含输入层、一层隐藏层和输出层的多层感知机，其中隐藏层由两个神经元构成。
- 网络使用Sigmoid函数作为神经元的激活函数，使用均方损失函数来计算网络输出值与实际值之间的误差。

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



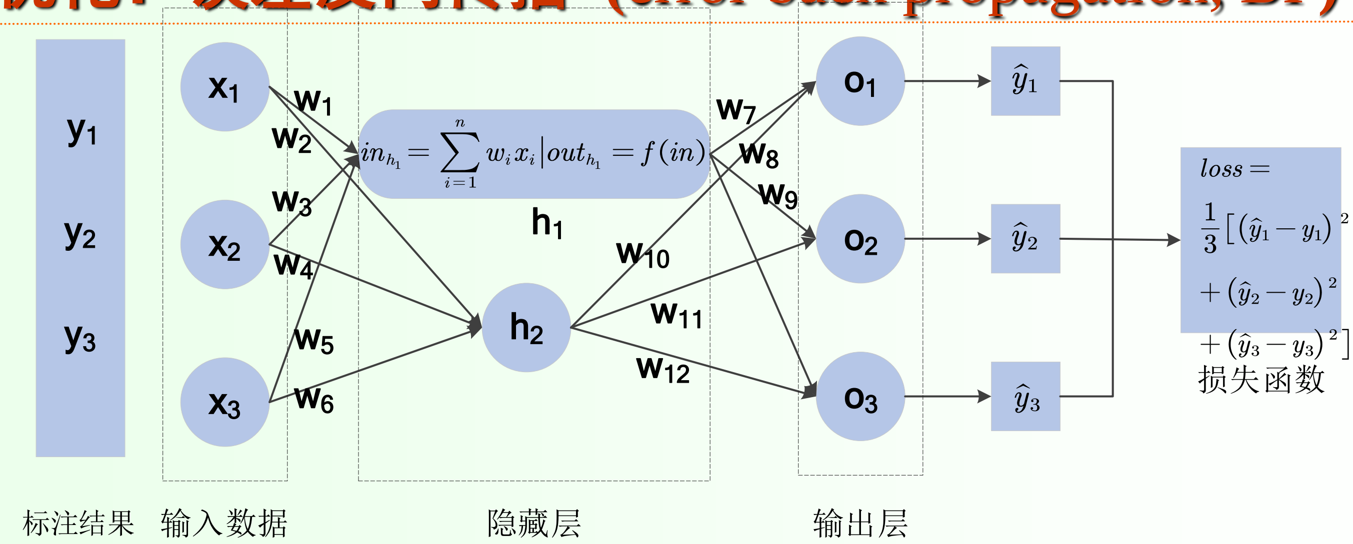
每个神经元完成如下两项任务：1）对相邻前序层所传递信息进行线性加权累加；2）对加权累加结果进行非线性变换。假设样本数据输入为(x_1, x_2, x_3)，其标注信息为(y_1, y_2, y_3)。在三类分类问题中，对于输入数据(x_1, x_2, x_3)，y_1、y_2和y_3中只有一个取值为1、其余两个取值为0。

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 $(\widehat{y_1}, \widehat{y_2}, \widehat{y_3})$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$loss = \sum_{i=1}^{3}(\hat{y}_i - y_i)^2$。

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~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$\mathcal{L}(w)$

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。

$$\delta_1=\frac{\partial\mathcal{L}}{\partial w_1}$$

$$=\frac{\partial\mathcal{L}}{\partial\widehat{y_1}}*\frac{\partial\widehat{y_1}}{\partial w_1}+\frac{\partial\mathcal{L}}{\partial\widehat{y_2}}*\frac{\partial\widehat{y_2}}{\partial w_1}+\frac{\partial\mathcal{L}}{\partial\widehat{y_3}}*\frac{\partial\widehat{y_3}}{\partial w_1}$$

$$=\frac{\partial\mathcal{L}}{\partial\widehat{y_1}}*\frac{\partial\widehat{y_1}}{\partial In_{o_1}}*\frac{\partial In_{o_1}}{\partial Out_{h_1}}*\frac{\partial Out_{h_1}}{\partial In_{h_1}}*\frac{\partial In_{h_1}}{\partial w_1}+\frac{\partial\mathcal{L}}{\partial\widehat{y_2}}*\frac{\partial\widehat{y_2}}{\partial In_{o_2}}*\frac{\partial In_{o_2}}{\partial Out_{h_1}}$$

$$*\frac{\partial Out_{h_1}}{\partial In_{h_1}}*\frac{\partial In_{h_1}}{\partial w_1}+\frac{\partial\mathcal{L}}{\partial\widehat{y_3}}*\frac{\partial\widehat{y_3}}{\partial In_{o_3}}*\frac{\partial In_{o_3}}{\partial Out_{h_1}}*\frac{\partial Out_{h_1}}{\partial In_{h_1}}*\frac{\partial In_{h_1}}{\partial w_1}$$

$$=\left(\frac{\partial\mathcal{L}}{\partial\widehat{y_1}}*\frac{\partial\widehat{y_1}}{\partial In_{o_1}}*\frac{\partial In_{o_1}}{\partial Out_{h_1}}+\frac{\partial\mathcal{L}}{\partial\widehat{y_2}}*\frac{\partial\widehat{y_2}}{\partial In_{o_2}}*\frac{\partial In_{o_2}}{\partial Out_{h_1}}+\frac{\partial\mathcal{L}}{\partial\widehat{y_3}}*\frac{\partial\widehat{y_3}}{\partial In_{o_3}}\right.$$

$$\left.*\frac{\partial In_{o_3}}{\partial Out_{h_1}}\right)*\frac{\partial Out_{h_1}}{\partial In_{h_1}}*\frac{\partial In_{h_1}}{\partial w_1}$$

$$=(\delta_7+\delta_8+\delta_9)*\frac{\partial Out_{h_1}}{\partial In_{h_1}}*\frac{\partial In_{h_1}}{\partial w_1}$$

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

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。

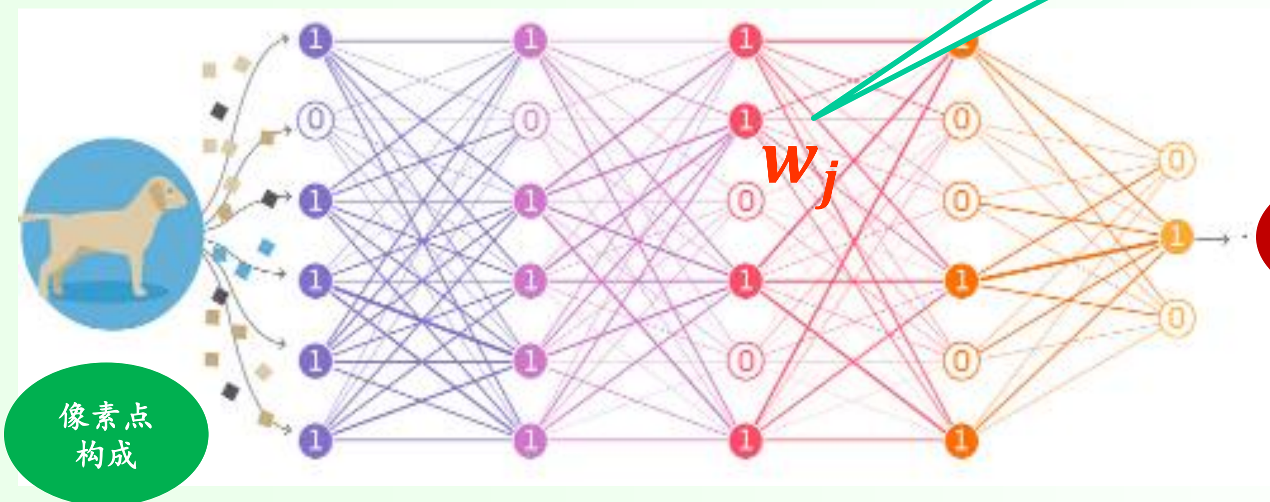
机器学习的能力在于拟合和优化

$$w_j^{new} = w_j^{old} - \eta \times \frac{\partial Model_{loss}}{\partial w_j}$$

调节参数

梯度下降

误差后向传播



dog

第一层 关键点

第二层 边界点

第三层 组件

第四层 特征

调节网络结构（炼金术）

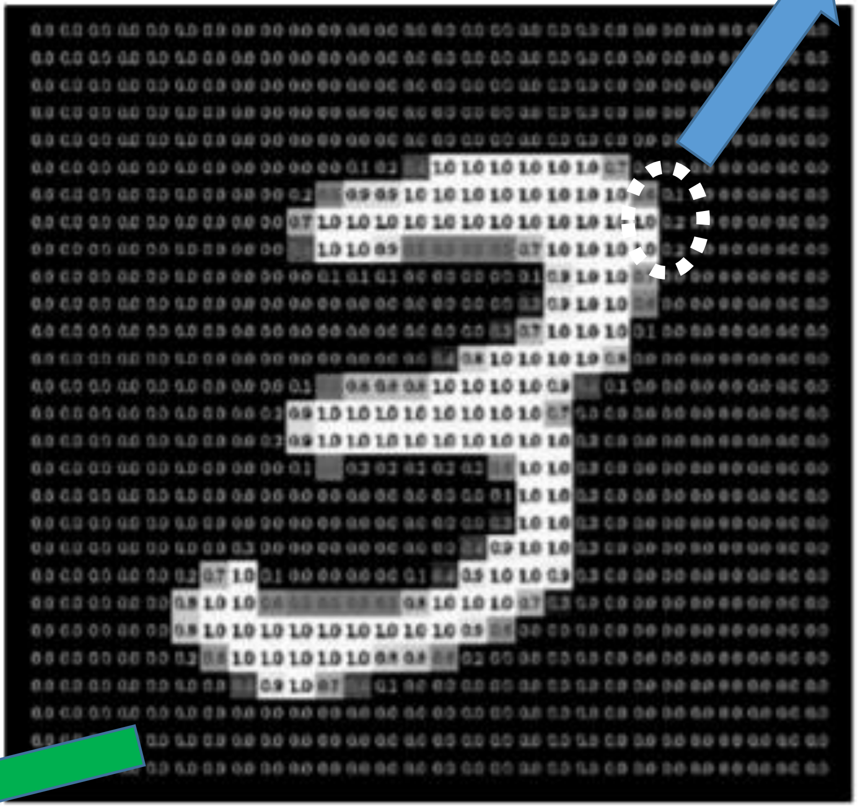
举例：神经网络构建

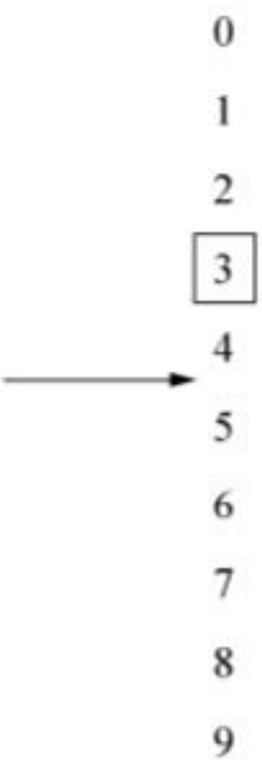
周边

0.1
0.1
0.2

（1）认识数据集的本质。

➢ 上述实例中用到的 MNIST 手写数字数据集（60000 个训练集和标签，10000 个测试 集和标签），就是每个数字的不同的写法。其中，每一个样本为代表 0～9 中的一个数字灰度图片，对应一个所代表数字的标签，图片大小 28 像素×28 像素（分辨率，28 行 28 列），且数字出现在图片正中间。





理想：白色部分为1，黑色部分为0

举例：神经网络构建

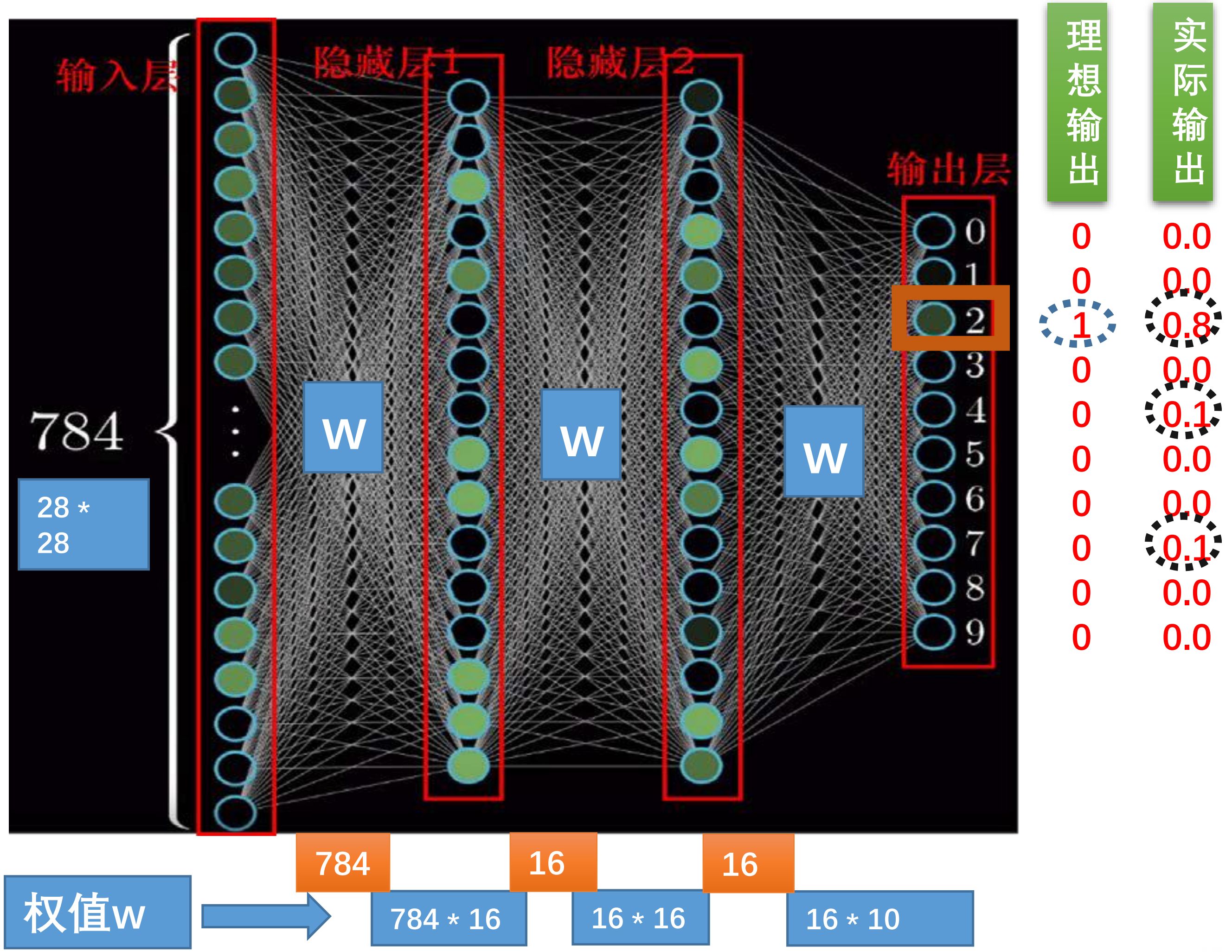


图 人工神经网络预测数字“2”的过程

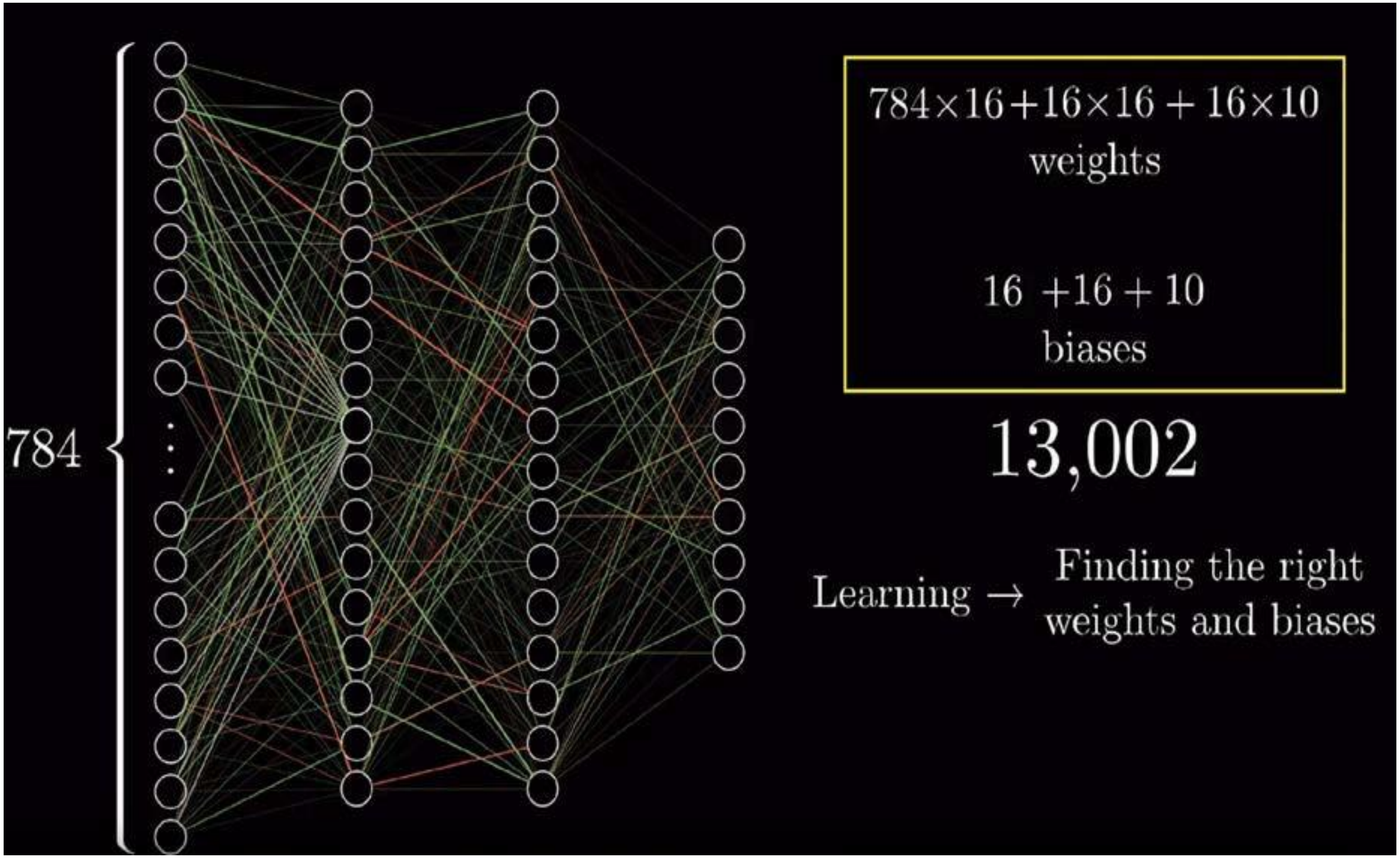


图 数字识别神经网络参数数量

ChatGPT

GPT(Generative Pre-trained Transformer（生成型预训练变换模型）)

OpenAI家族

OpenAI总部位于旧金山，由特斯拉的**马斯克、Sam Altman及其他投资者**在2015年共同创立，目标是开发造福全人类的AI技术。而马斯克则在2018年时因公司发展方向分歧而离开。

此前，OpenAI 因推出 GPT系列自然语言处理模型而闻名。从2018年起，OpenAI就开始发布生成式预训练语言模型GPT ，可用于生成文章、代码、机器翻译、问答等各类内容。每一代GPT模型的参数量都爆炸式增长，堪称“越大越好”。2019年2月发布的GPT-2参数量为15亿，而2020年5月的GPT-3，参数量达到了1750亿。

模型	发布时间	参数量	预训练数据量
GPT-1	2018年6月	1.17亿	约5GB
GPT-2	2019年2月	15亿	40G
GPT-3	2020年5月	1750亿	45TB
GPT-4	2023年7月	1.8万亿	13万亿

相当于阅读了1亿本书

训练1次6300万美元