《神经网络》

崔志勇

交通科学与工程学院

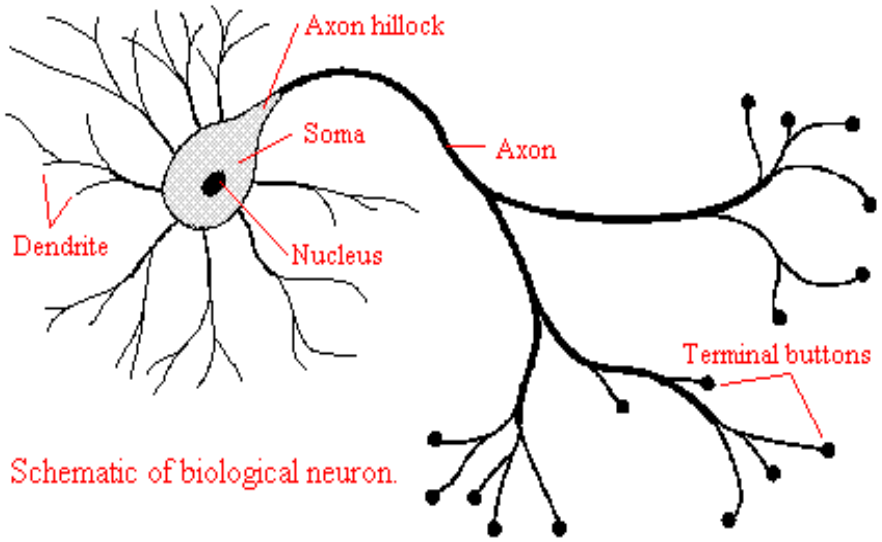
2024年5月

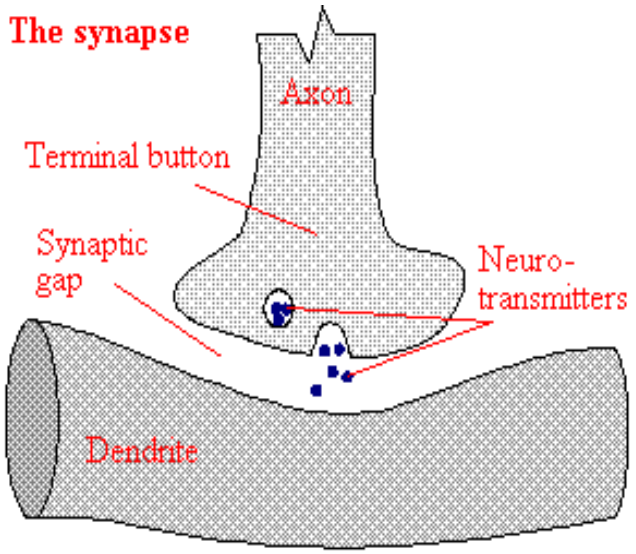
人工神经网络

人工神经网络的灵感来自于对人类和其他哺乳动物大脑工作方式的研究和模型。研究人员认为人脑是一个高度复杂、非线性和并行的计算机或信息处理系统，能够执行高度复杂的任务。

人工神经网络

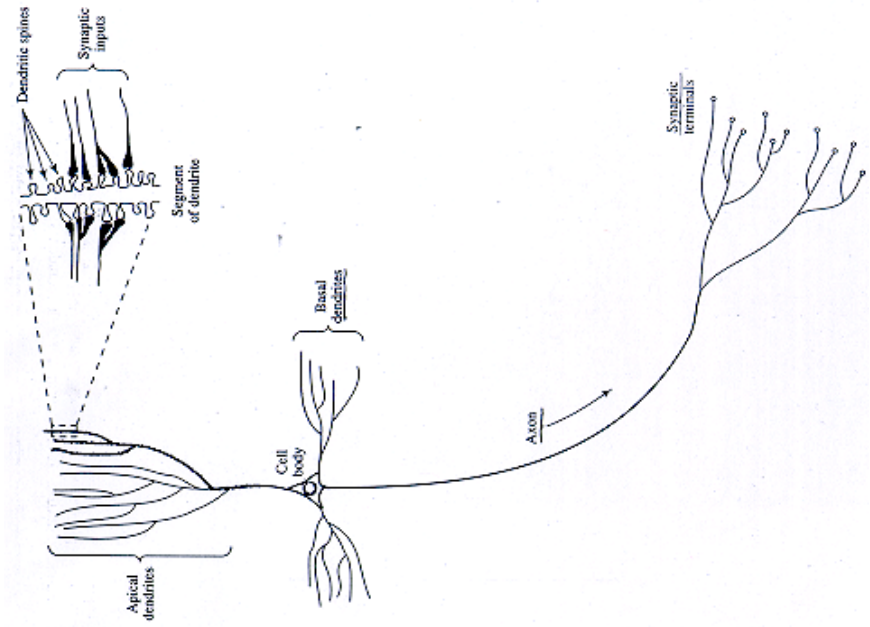
大脑是由大约 100 亿个相互连接的神经元组成的。每个神经元都是一个利用生化反应来接收、处理和传递信息的细胞。

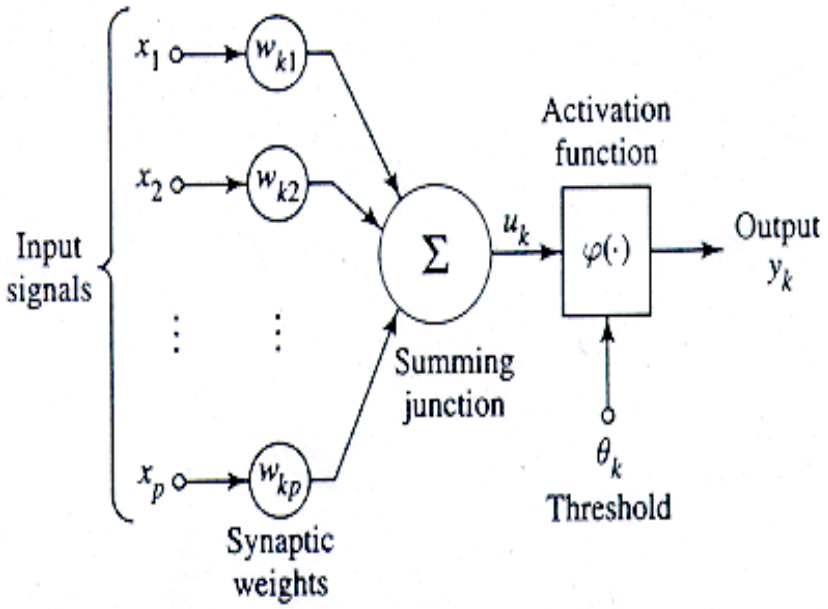




Dendrites（树突）
Soma（体细胞）
Axon（轴突）
Synapses （突触）

从生物神经元到人工神经元





Dendrite Cell Body Axon

人工神经网络 vs 生物神经元

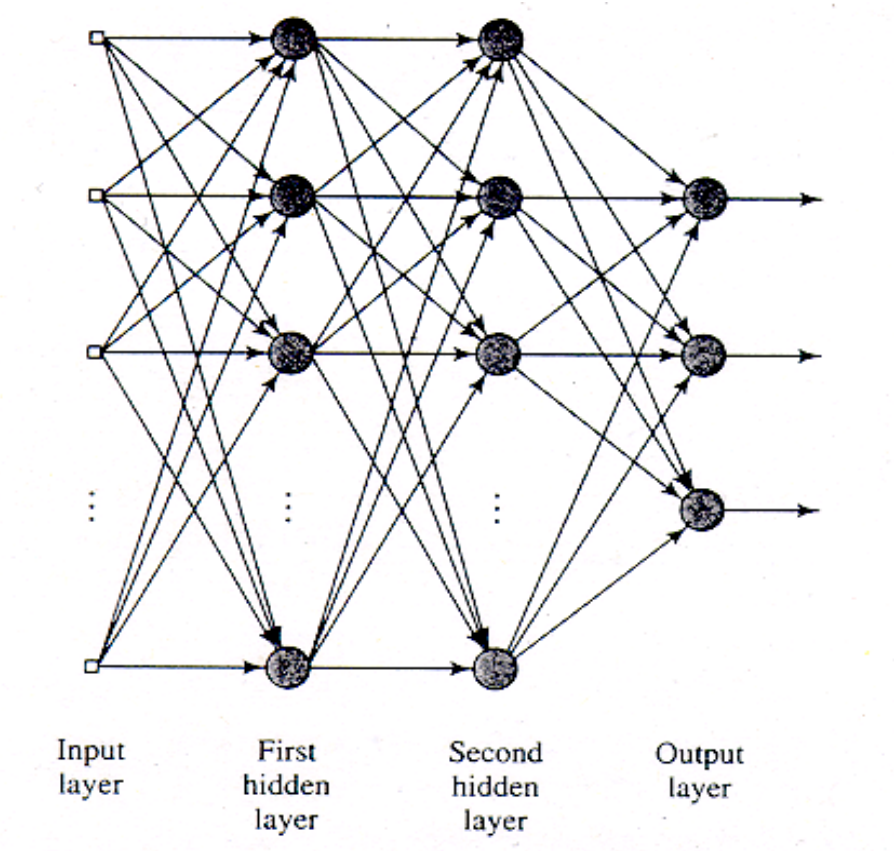
ANN	BNN
Soma (神经元胞体)	节点
Dendrites (树突)	输入
Synapse (突触)	权重或互连
Axon (轴突/神经纤维)	输出

人工神经网络 vs 生物神经元

标准	BNN	ANN
处理	大规模并行，速度慢，但优于ANN	大规模并行，速度快但不如BNN
尺寸	10^{11} 个神经元和 10^{15}个互连	10^{2}到10^{4} 个节点
学习	可以容忍歧义	需要非常精确、结构化和格式化的数据来容忍歧义
容错	性能随部分损坏而下降	具有强大的性能，因此具有容错潜力
存储容量	将信息存储在突触中	将信息存储在连续的内存位置中

人工神经网络的特性

- 人工神经元网络



- 特征
  - 自适应地
  - 泛化能力
  - 容错（平缓降级）
  - 生物学类比
  - 非线性 I/O 映射

人工神经网络的类型

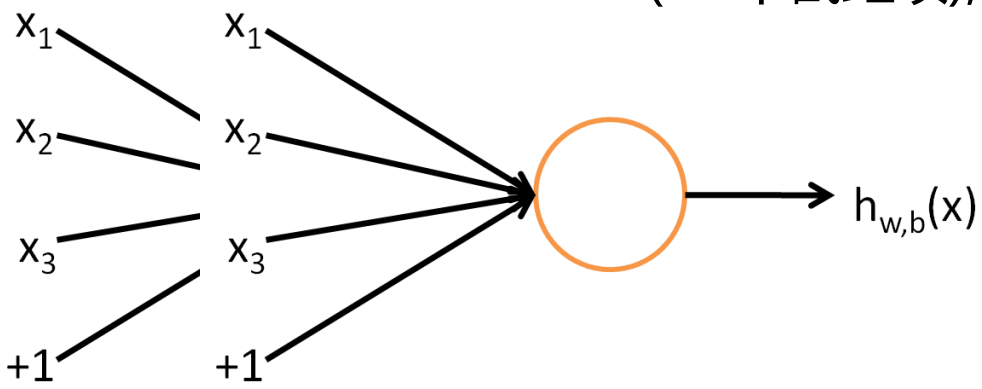
- 单层感知器
- 多层感知器 (MLPs)
- 径向基函数网络 (RBFs)
- 霍普菲尔德网络(Hopfield Network)
- 玻尔兹曼机(Boltzmann Machine)
- 自组织图(SOM)
- 模块化网络 (Committee Machines)

激活函数

为了描述神经网络，我们首先要描述最简单的神经网络，即由单个“神经元”组成的神经网络。我们将使用下图来表示单个神经元：

这个“神经元”是一个计算单元，输入为 x_1, x_2, x_3 (+1个截距项), 输出为 $h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b)$,

其中 $f : \Re \to \Re$ 为激活函数



$$f(z) = \frac{1}{1+\exp(-z)}.$$

Sigmoid 函数

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}.$$

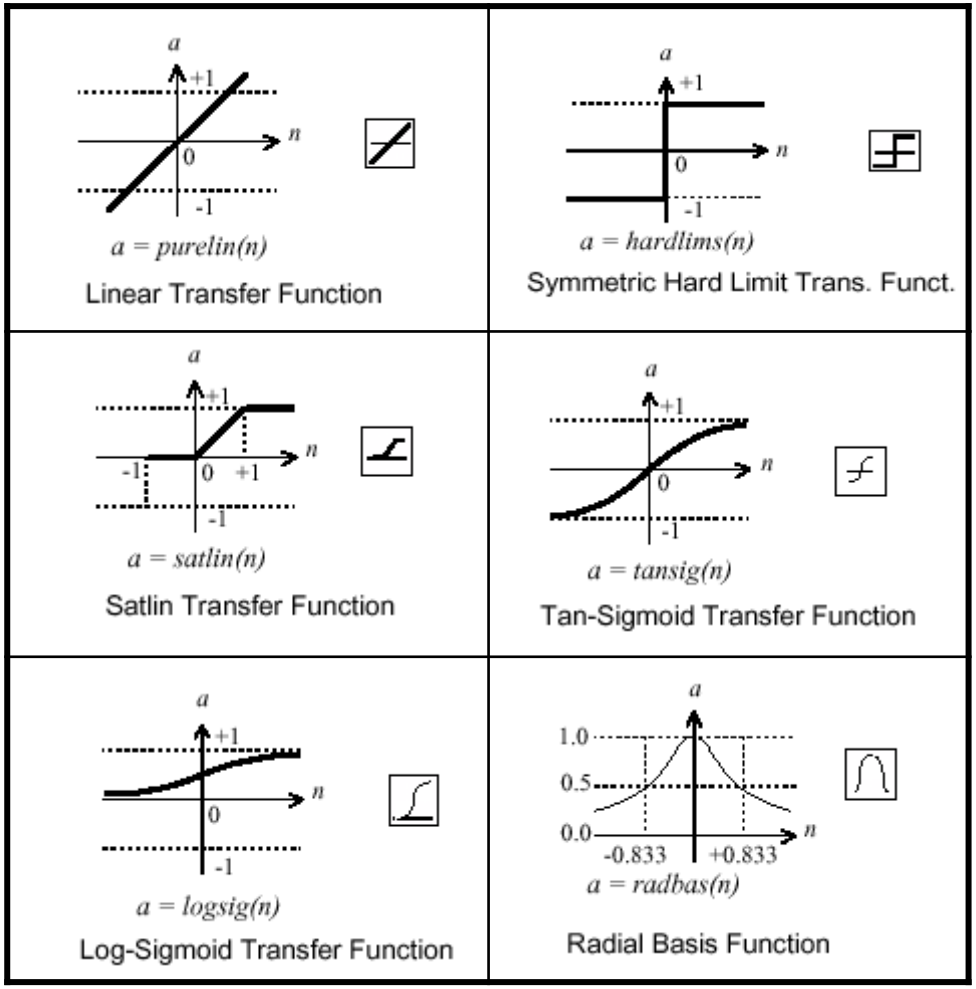
Tanh 函数

$$f(z) = \max(0, x).$$

线性整流函数

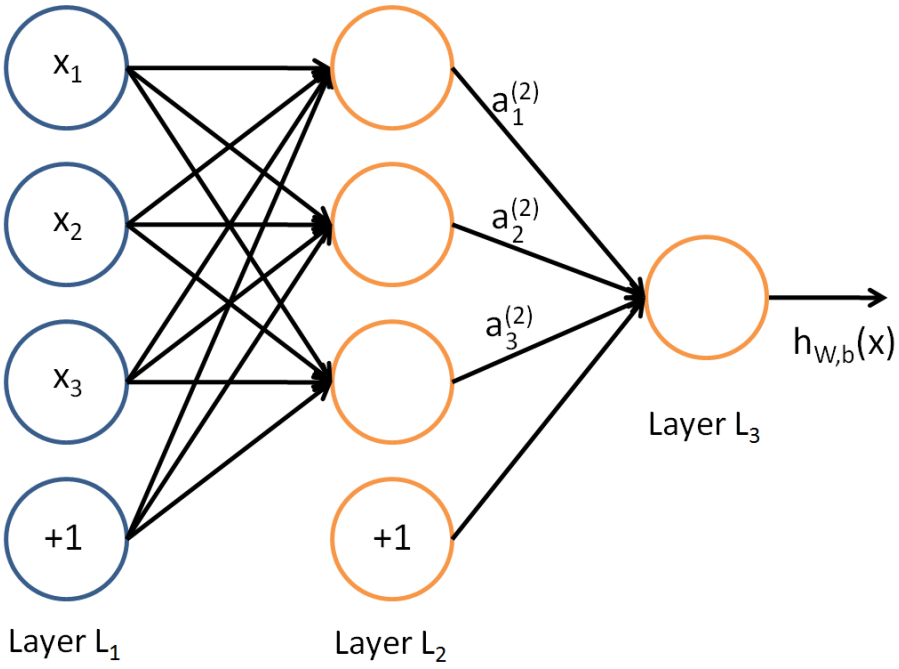
激活函数

激活函数通常是非线性的。线性函数有局限性，因为输出仅与输入成正比。



神经网络模型

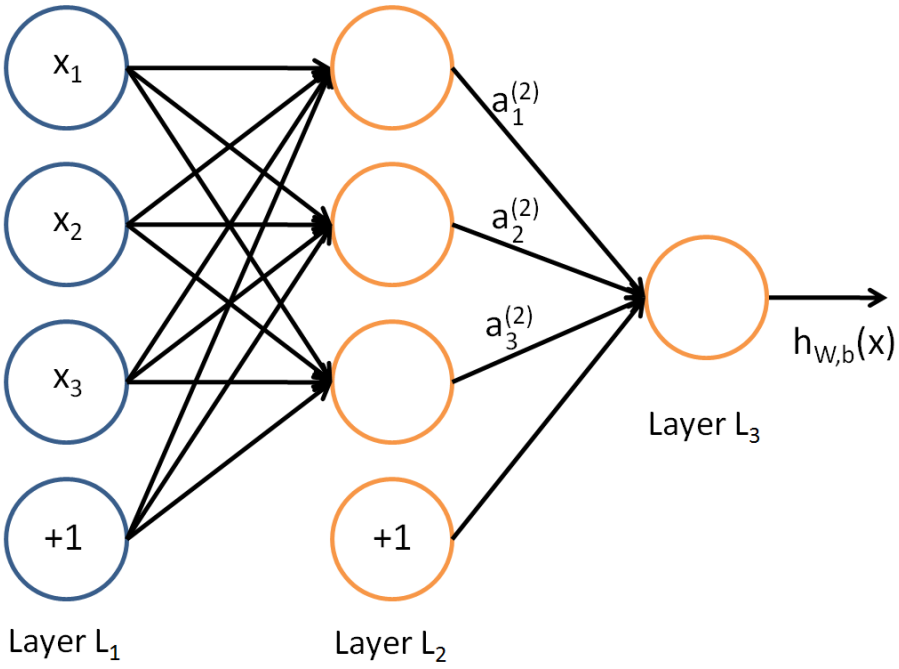
- 前馈网络



在此图中，我们使用圆圈表示网络的输入。标有“+1”的圆圈称为偏差单元，与截距项相对应。网络的最左边的层称为输入层，最右边的层称为输出层（在此示例中，输出层只有一个节点）。中间的节点层称为隐藏层，因为在训练集中未观察到其值。我们还说我们的示例神经网络有 3 个输入单元（不包括偏差单元）、3 个隐藏单元和 1 个输出单元。

神经网络模型

- 前馈网络



我们将用 n_l 表示我们网络中的层数；因此，在我们的例子中 $n_l = 3$。我们将第 l 层标记为 L_l，因此 L_1 层是输入层，L_{n_l} 层是输出层。我们的神经网络的参数为 $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$ 其中 $W_{ij}^{(l)}$ 表示第 l 层的单元 j 和第 l+1 层的单元 i 之间的连接参数（或权重）。(注意索引的顺序）。此外，$b_i^{(l)}$ 是与第 l+1 层的单元 i 相关的偏置项。因此，在我们的例子中，我们有 $W^{(1)} \in \Re^{3\times 3}$ 和 $W^{(2)} \in \Re^{1\times 3}$. 注意，偏置单元没有输入或连接进入它们，因为它们总是输出值 +1。我们还用 s_l 表示第 l 层的节点数（不包括偏置单元）。

前向传播算法

请注意，这样更紧凑的表示法很容易实现。具体来说，如果我们将激活函数 $f(\cdot)$ 扩展为逐元素应用于向量（即 $f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]$），那么我们可以将上面的方程更紧凑地写成：

$$z^{(2)} = W^{(1)}x + b^{(1)}$$

$$a^{(2)} = f(z^{(2)})$$

$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$$

$$h_{W,b}(x) = a^{(3)} = f(z^{(3)})$$

我们将这一步称为前向传播。更一般地说，

$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}$$
$$a^{(l+1)} = f(z^{(l+1)})$$

单选题 1分

设置

一个具有两个隐藏层的全连接神经网络模型用于二分类问题：

第 i层可以表示为：$h_i = \sigma(W_{i-1}h_{i-1} + b_{i-1})$。

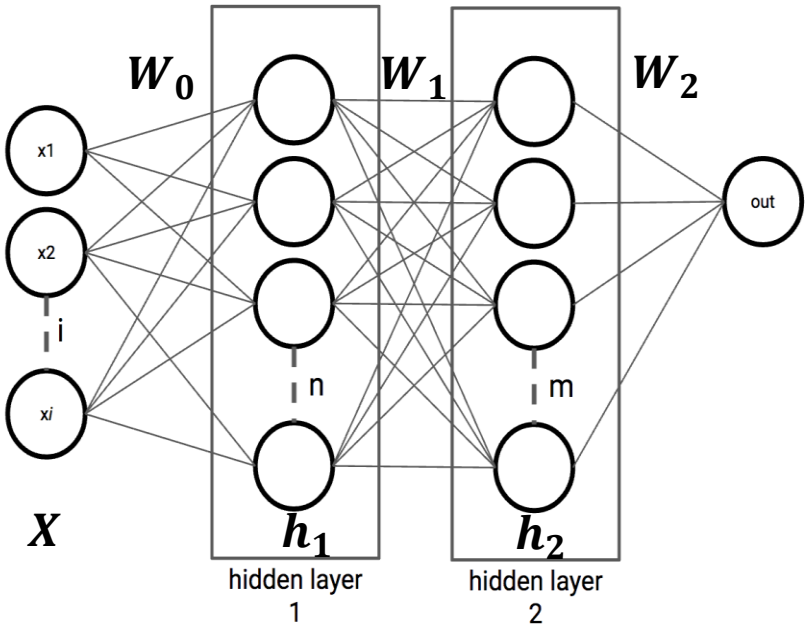
如果输入层$\boldsymbol{h_0}$ 即 $\boldsymbol{X} \in \mathbb{R}^{16\times1}$，第1个隐藏层的输出为$\boldsymbol{h_1} \in \mathbb{R}^{32\times1}$，第2个隐藏层的输出为 $\boldsymbol{h_2} \in \mathbb{R}^{16\times1}$，以下说法**错误**的是

A W_0 的形状是 $\mathbb{R}^{32\times16}$

B b_0 的形状是 $\mathbb{R}^{32\times1}$

C W_1 的形状是 $\mathbb{R}^{32\times16}$

D W_2 的形状是 $\mathbb{R}^{1\times16}$



提交

代价函数

我们将针对该单个示例的代价函数定义为：

$$J(W, b; x, y) = \frac{1}{2} \left\| h_{W,b}(x) - y \right\|^2.$$

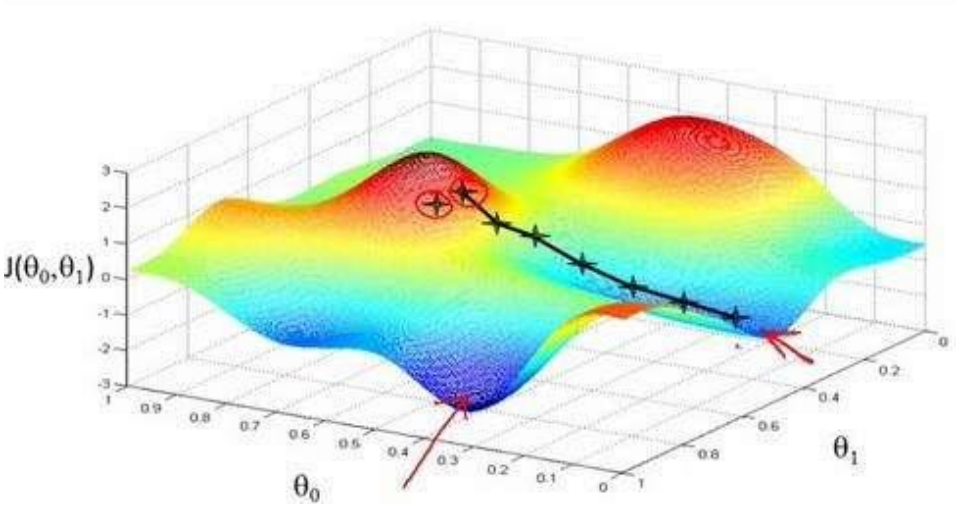
给定一个包含 m 个示例的训练集，我们将总体代价函数定义为：

$$\begin{aligned} J(W,b) &= \left[\frac{1}{m} \sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)}) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{(l)} \right)^2 \\ &= \left[\frac{1}{m} \sum_{i=1}^{m} \left(\frac{1}{2} \left\| h_{W,b}(x^{(i)}) - y^{(i)} \right\|^2 \right) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{(l)} \right)^2 \end{aligned}$$

第二项是正则化项（也称为权重衰减项），它倾向于降低权重的大小，并有助于防止过度拟合。

ANN 优化：梯度下降

误差度量: $E = \frac{1}{N}\sum_{t=1}^{N}(F(x_t;W)-y_t)^2$

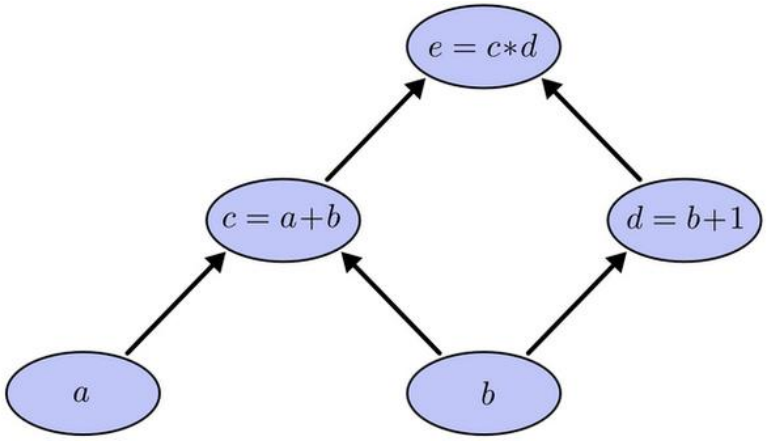


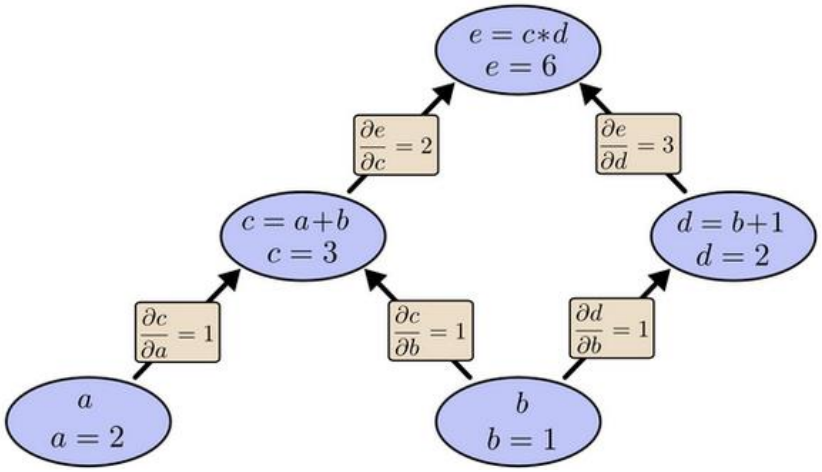
改变突触权重的规则:

$$\Delta w_i^j = -c \cdot \frac{\partial E}{\partial w_i^j}(W)$$

$$w_i^{j,new} = w_i^j + \Delta w_i^j$$

ANN 训练：反向传播算法





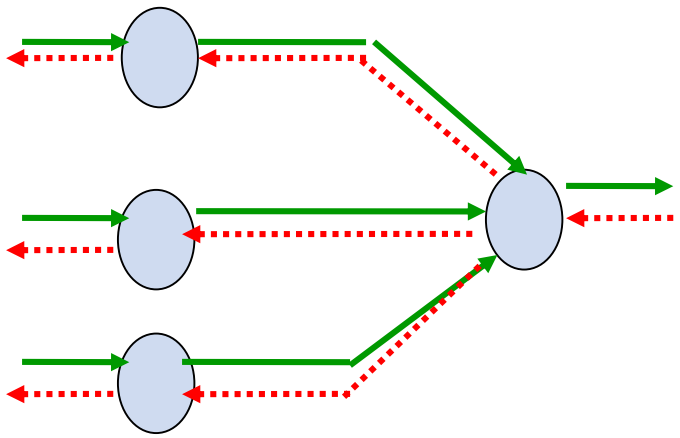
计算 e 和 a、b之间的偏导数, a=2, b=1:

$$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial c} \cdot \frac{\partial c}{\partial a} \qquad \frac{\partial e}{\partial b} = \frac{\partial e}{\partial c} \cdot \frac{\partial c}{\partial b} + \frac{\partial e}{\partial d} \cdot \frac{\partial d}{\partial b}$$

多条路径被重复访问（a-c-e，b-c-e）->BP算法解决了这个问题!

ANN 训练：反向传播算法

- 反向传播训练算法



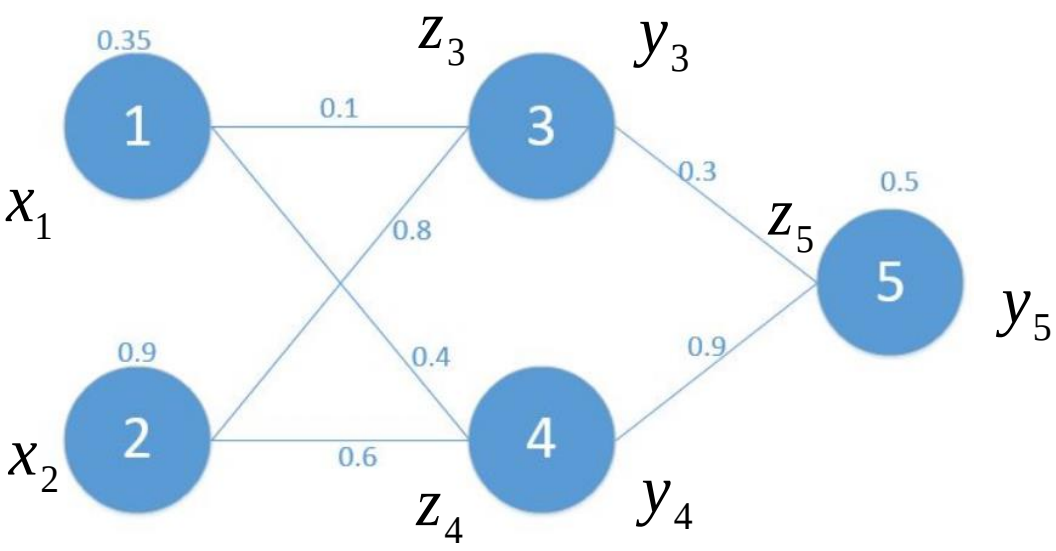
Network activation Forward Step

Error propagation Backward Step

- 反向传播调整 NN 的权重，以最小化网络总均方误差。

Example

一个三层的神经网络



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

$$X = Z_0 = \begin{bmatrix} 0.35 \\ 0.9 \end{bmatrix}$$

$$y_{out} = 0.5$$

$$W_0 = \begin{bmatrix} w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.1 & 0.8 \\ 0.4 & 0.6 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} w_{53} & w_{54} \end{bmatrix} = \begin{bmatrix} 0.3 & 0.9 \end{bmatrix}$$

Example

前向传播

$$Z_1 = \begin{bmatrix} z_3 \\ z_4 \end{bmatrix} = W_0 X = \begin{bmatrix} w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.1*0.35+0.8*0.9 \\ 0.4*0.35+0.6*0.9 \end{bmatrix} = \begin{bmatrix} 0.755 \\ 0.68 \end{bmatrix}$$

Then,

$$Y_1 = f(Z_1) = \begin{bmatrix} 0.68 \\ 0.663 \end{bmatrix}$$

$$Z_2 = z_5 = W_1 Y_1 = \begin{bmatrix} w_{53} & w_{54} \end{bmatrix} \begin{bmatrix} y_3 \\ y_4 \end{bmatrix}$$

$$= [0.801]$$

Then,

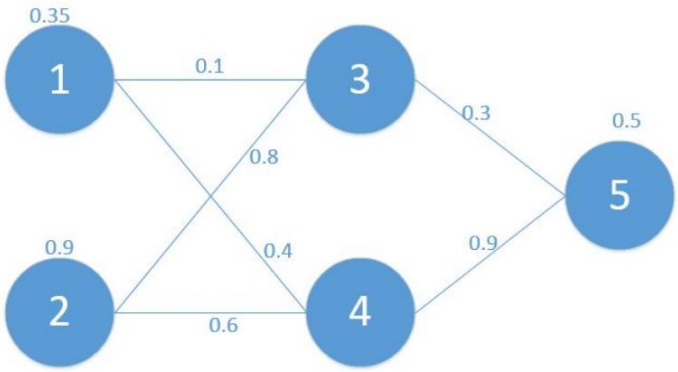
$$Y_2 = y_5 = f(Z_2) = [0.690]$$

Cost Value $$C = \frac{1}{2}(0.690 - 0.5)^2 = 0.01805$$

Example

反向传播

$$\begin{cases} C = \frac{1}{2}(y_5 - y_{out})^2 \\ y_5 = f(z_5) \\ z_5 = w_{53}y_3 + w_{54}y_4 \end{cases}$$



计算 C 对 w 的导数

$$\frac{\partial C}{\partial w_{53}} = \frac{\partial C}{\partial y_5}\frac{\partial y_5}{\partial z_5}\frac{\partial z_5}{\partial w_{53}}$$

$$= (y_5 - y_{out}) f'(z_5) y_3$$

why?

$$= (y_5 - y_{out}) f(z_5)(1 - f(z_5)) y_3$$

$$= (0.69 - 0.5) \times 0.69 \times (1 - 0.69) * 0.68$$

$$= 0.02763$$

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

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

Similarly, $\frac{\partial C}{\partial w_{54}} = \frac{\partial C}{\partial y_5}\frac{\partial y_5}{\partial z_5}\frac{\partial z_5}{\partial w_{54}} = (y_5 - y_{out}) f(z_5)(1 - f(z_5)) y_4$

$$= 0.02711$$

Example

我们也需要知道 $w_{31}, w_{32}, w_{41}, w_{42}$

$$
\begin{cases}
C = \frac{1}{2}(y_5 - y_{out})^2 \\
y_5 = f(z_5) \\
z_5 = w_{53}y_3 + w_{54}y_4 \\
y_3 = f(z_3) \\
z_3 = w_{31}x_1 + w_{32}x_2
\end{cases}
$$

计算 C 对 w 的导数

$$
\frac{\partial C}{\partial w_{31}} = \frac{\partial C}{\partial y_5}\frac{\partial y_5}{\partial z_5}\frac{\partial z_5}{\partial y_3}\frac{\partial y_3}{\partial z_3}\frac{\partial z_3}{\partial w_{31}}
$$

$$
= (y_5 - y_{out}) f(z_5)(1 - f(z_5)) w_{53} f(z_3)(1 - f(z_3)) x_1
$$

$$
\begin{cases}
w_{31} = w_{31} - \eta \frac{\partial C}{\partial w_{31}} = 0.096 \\
w_{32} = w_{32} - \eta \frac{\partial C}{\partial w_{32}} = 0.798 \\
w_{41} = w_{41} - \eta \frac{\partial C}{\partial w_{41}} = 0.396 \\
w_{42} = w_{42} - \eta \frac{\partial C}{\partial w_{42}} = 0.589
\end{cases}
$$

使用新的权重值进行前向传播。
New cost value:0.165

Example

我们可以运行100次，cost值=5.92e-07。**最终的权重值**：

$$W_0 = \begin{bmatrix} w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.0993309 & 0.6425254 \\ 0.3993309 & 0.4425254 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} w_{53} & w_{54} \end{bmatrix} = \begin{bmatrix} -0.30032342 & 0.31508797 \end{bmatrix}$$

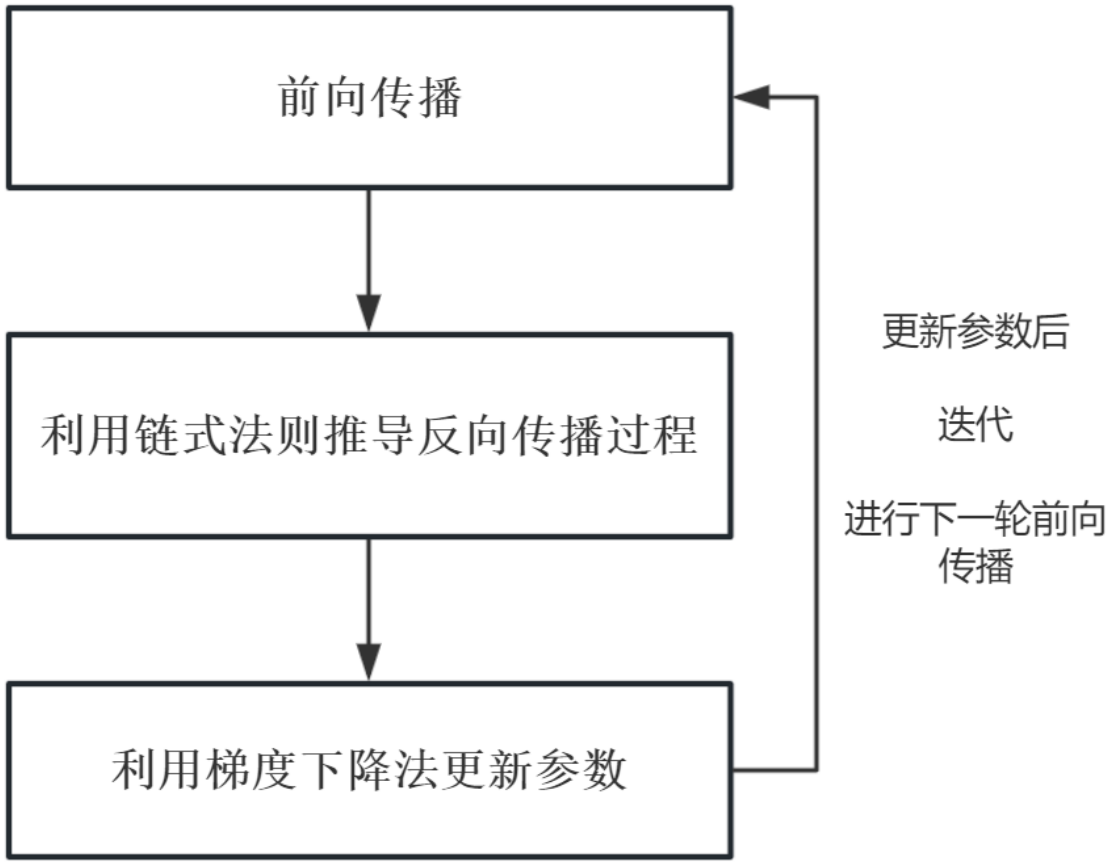
对比一下我们权重的**初始值**：

$$W_0 = \begin{bmatrix} w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.1 & 0.8 \\ 0.4 & 0.6 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} w_{53} & w_{54} \end{bmatrix} = \begin{bmatrix} 0.3 & 0.9 \end{bmatrix}$$

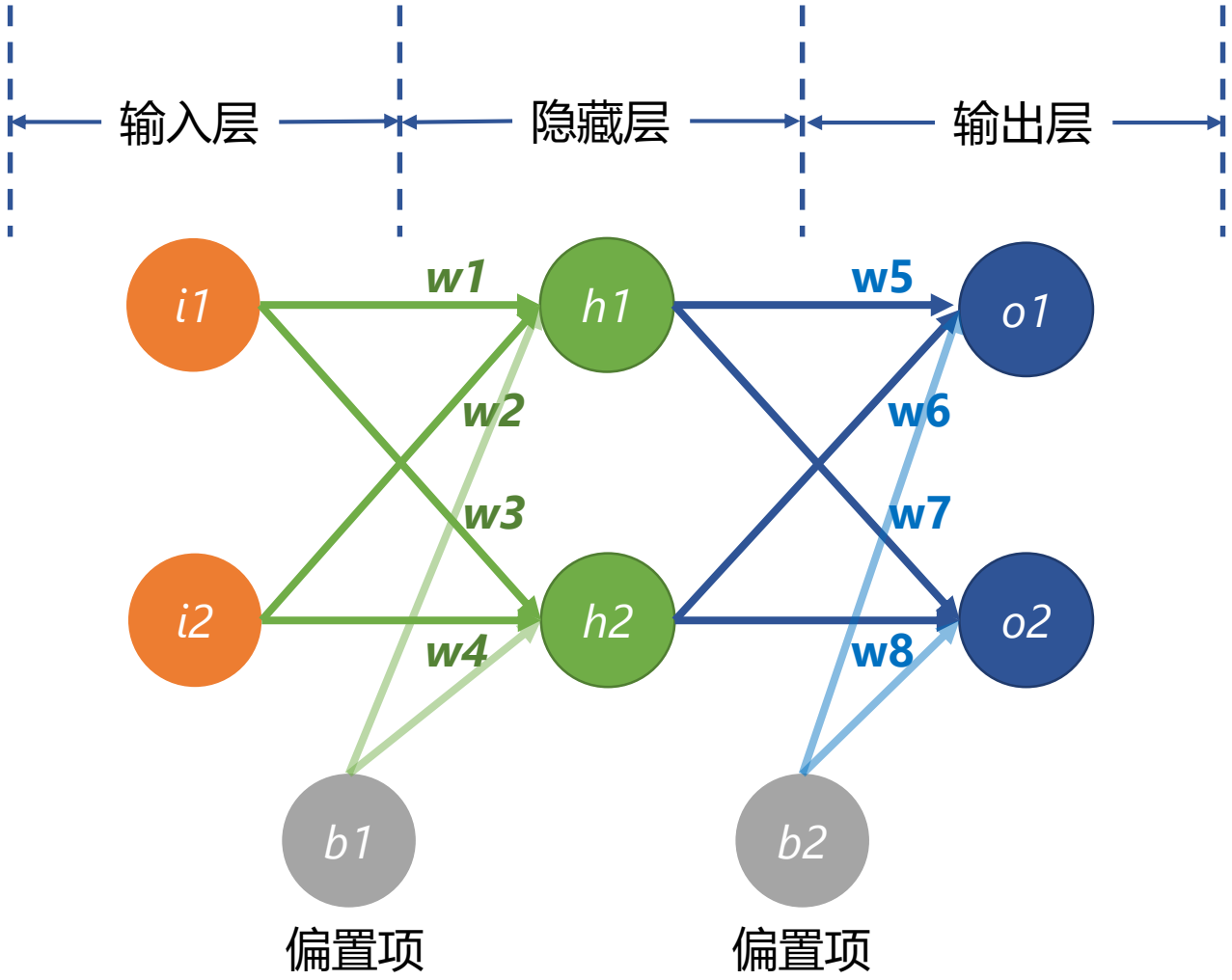
反向传播算法与梯度下降更新

整体流程



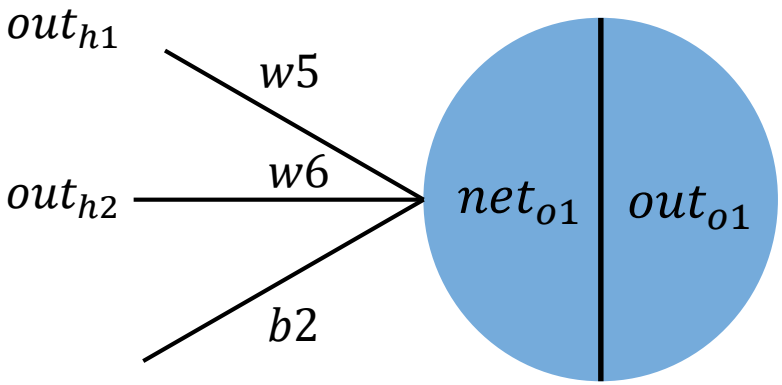
反向传播算法与梯度下降更新

网络结构



反向传播算法与梯度下降更新

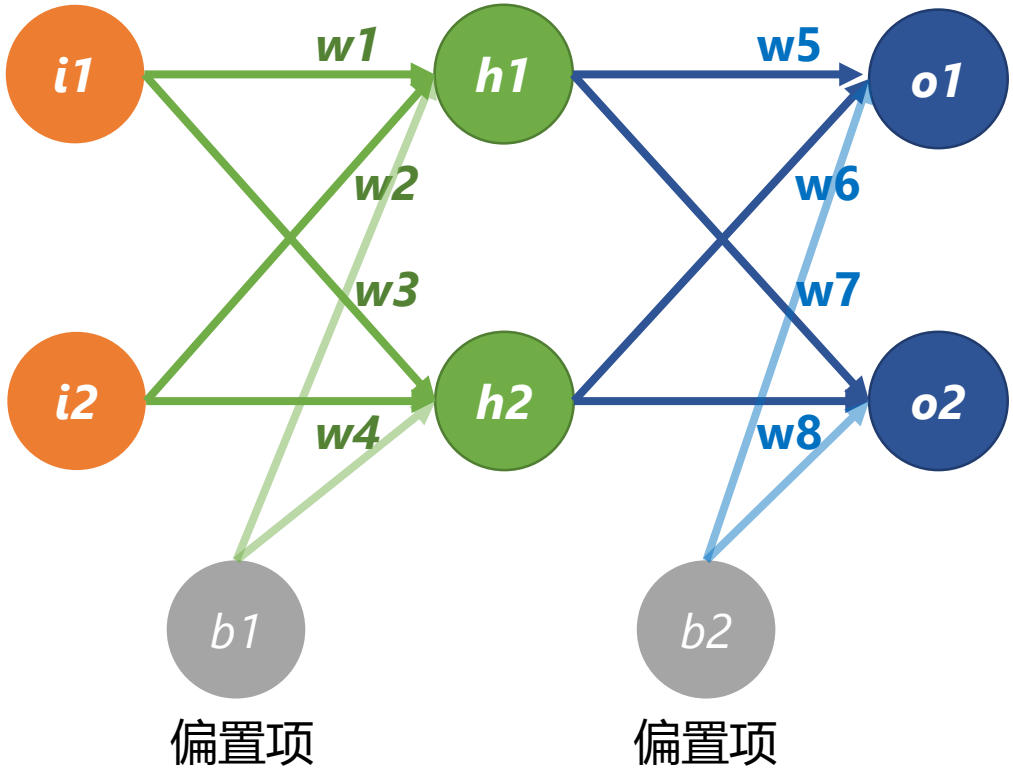
神经元结构



- $net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2$
- $out_{o1} = sigmoid(net_{o1})$

反向传播算法与梯度下降更新

Step0：参数初始化



其中，输入数据 $i1$=0.02，$i2$=0.7;
输出数据 $o1$=0，$o2$=1;

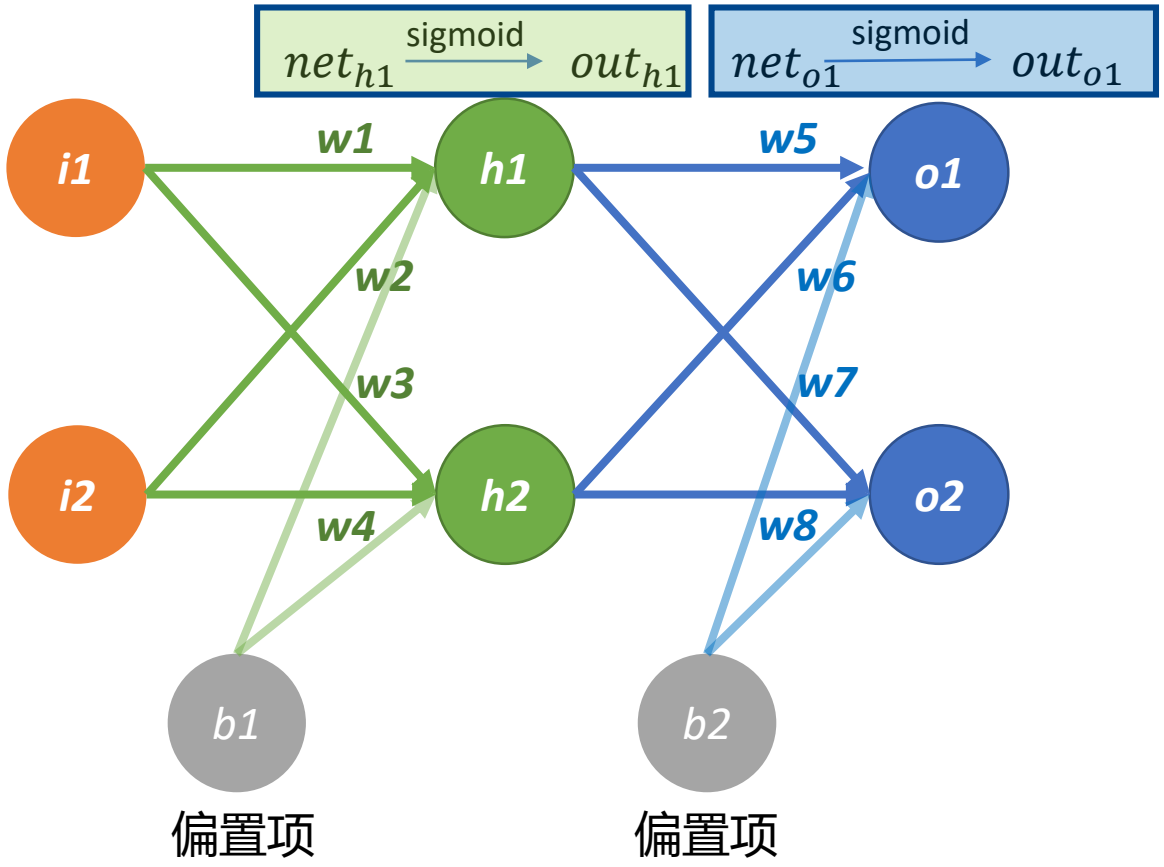
初始化权重和偏置项

$w1$=0.1，$w2$=0.2，$w3$=0.3，$w4$=0.4;
$w5$=0.2，$w6$=0.3，$w7$=0.1，$w8$=0.2
$b1$=0.3，$b2$=0.2

目标：给出输入数据i1,i2(0.02和0.7)，训练模型，更新权重参数，使输出尽可能与原始输出o1,o2(0和1)接近。

反向传播算法与梯度下降更新

Step1：前向传播(激活函数为sigmoid)



1.1 输入层---->隐藏层

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1$$
$$= 0.1 * 0.02 + 0.2 * 0.7 + 0.3 = 0.442$$

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.442}} = 0.608$$

$$out_{h2} = \frac{1}{1+e^{-net_{h2}}} = \frac{1}{1+e^{-0.586}} = 0.642$$

1.2 隐藏层---->输出层

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2$$
$$= 0.2 * 0.608 + 0.3 * 0.642 + 0.2 = 0.514$$

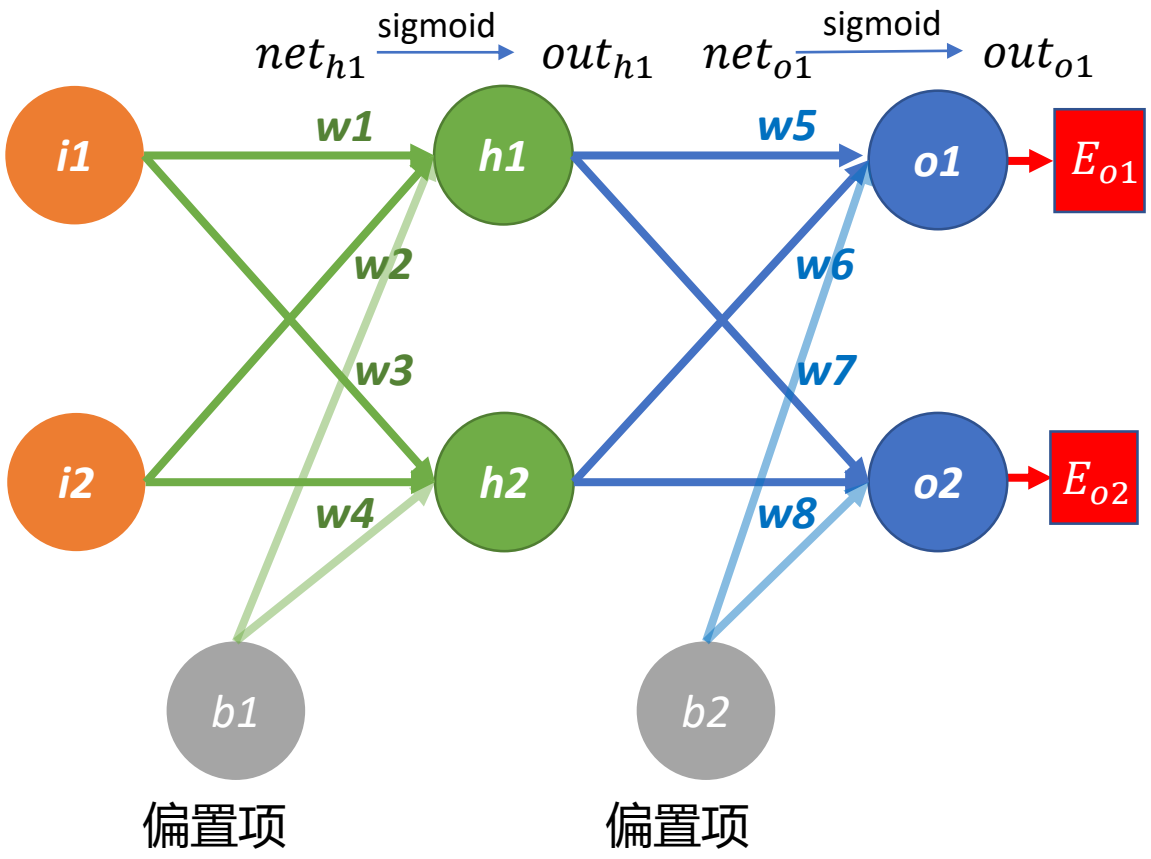
$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-0.514}} = 0.625$$

$$out_{o2} = \frac{1}{1+e^{-net_{o2}}} = \frac{1}{1+e^{-0.389}} = 0.596$$

输出值为[0.625，0.596]，与实际值[0，1]相差大。

反向传播算法与梯度下降更新

Step2：误差反向传播



2.1 计算总误差E_{total}

采用平方误差的方法

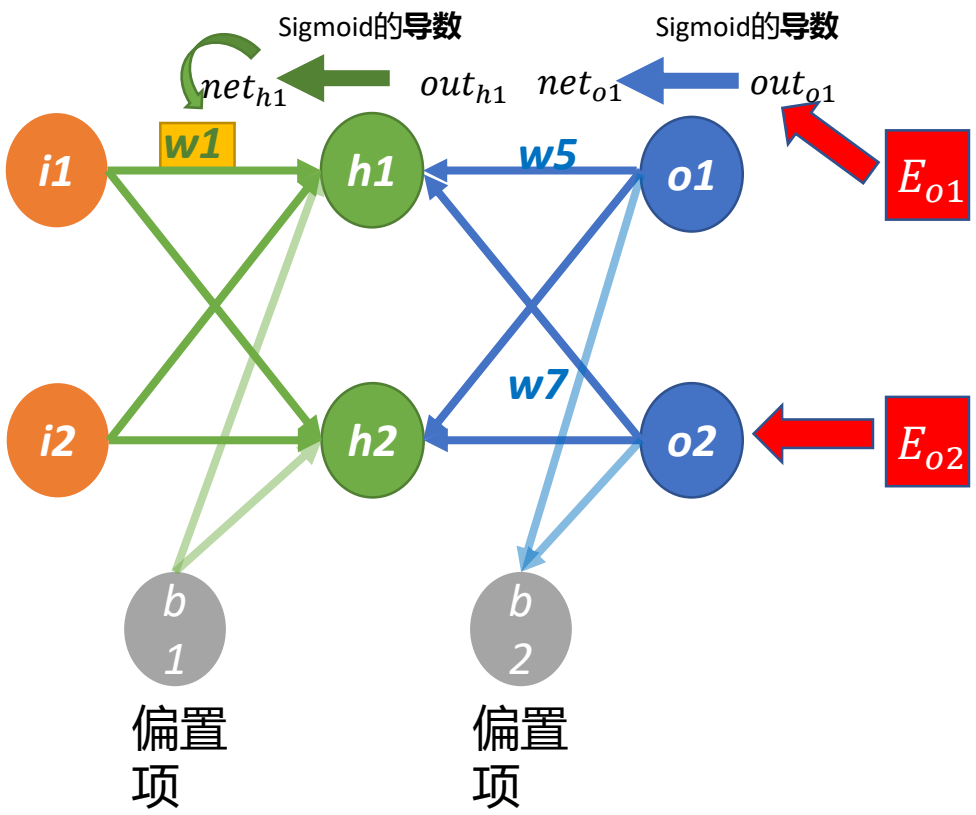
$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2$$
$$= \frac{1}{2}(0 - 0.62586)^2 = 0.195$$

$$E_{o2} = \frac{1}{2}(target_{o2} - out_{o2})^2$$
$$= \frac{1}{2}(1 - 0.59613)^2 = 0.081$$

$$E_{total} = E_{o1} + E_{o2} =$$
0.19585 + 0.08156 = 0.27741

反向传播算法与梯度下降更新

Step2：误差反向传播



2.2 输入层---->隐藏层的权值更新

（以权重参数*w1*为例，需要求出总误差 E_{total}对于*w1*的偏导，即$\frac{\partial E_{total}}{\partial w_1}$）

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}}$$

= 0.625* 0.234*w5

= 0.625* 0.234*0.2 = 0.029

$$\frac{\partial E_{o2}}{\partial out_{h1}} = \frac{\partial E_{o2}}{\partial out_{o2}} * \frac{\partial out_{o2}}{\partial net_{o2}} * \frac{\partial net_{o2}}{\partial out_{h1}} =$$

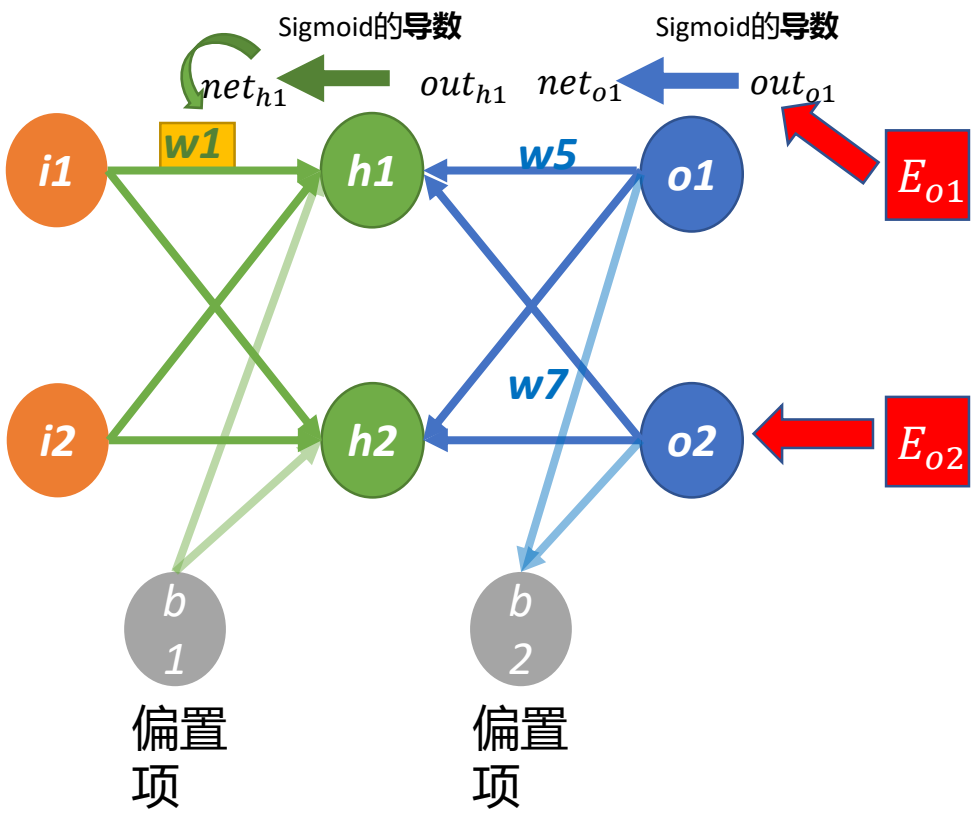
-0.403* 0.240*w7

= -0.403* 0.240 *0.1 = -0.009

故而 $\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$ = 0.02931 + (- 0.00972) = 0.019

反向传播算法与梯度下降更新

Step2：误差反向传播



2.2 输入层---->隐藏层的权值更新

（以权重参数*w1*为例，需要求出总误差E_{total}对于*w1*的偏导，即$\frac{\partial E\ total}{\partial w_1}$）

$$\frac{\partial E\ total}{\partial w_1} = \frac{\partial E\ total}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial\ net_{h1}}{\partial w_1}$$

$$out_{h1} = \frac{1}{1 + e^{-net_{h1}}}$$

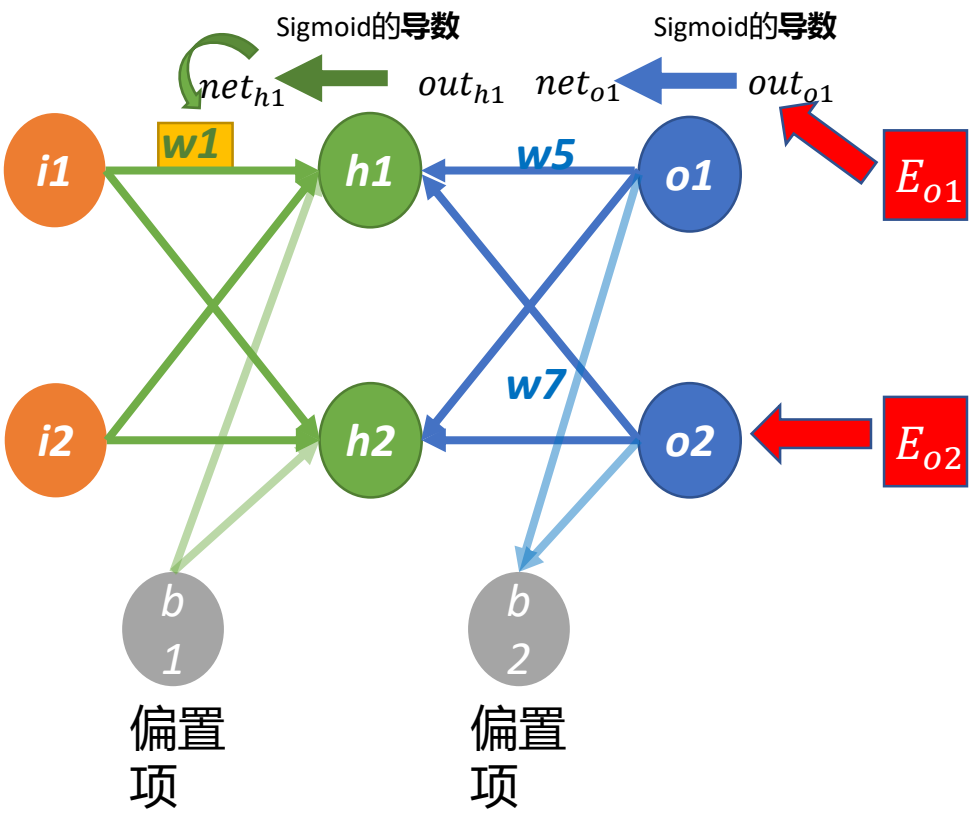
$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1})$$
$$= 0.608 * (1 - 0.608)$$
$$= 0.238$$

$$\frac{\partial\ net_{h1}}{\partial w_1} = i_1 = 0.020$$

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1$$

反向传播算法与梯度下降更新

Step2：误差反向传播



2.3 输入层---->隐藏层的权值更新

（以权重参数*w1*为例，需要求出总误差E_{total}对于*w1*的偏导，即$\frac{\partial E\ total}{\partial w_1}$）

将计算好的三者进行相乘：

$$\frac{\partial E\ total}{\partial w_1} = \frac{\partial E\ total}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial\ net_{h1}}{\partial w_1}$$
$$= 0.019 * 0.238 * 0.020$$
$$= 0.00009$$

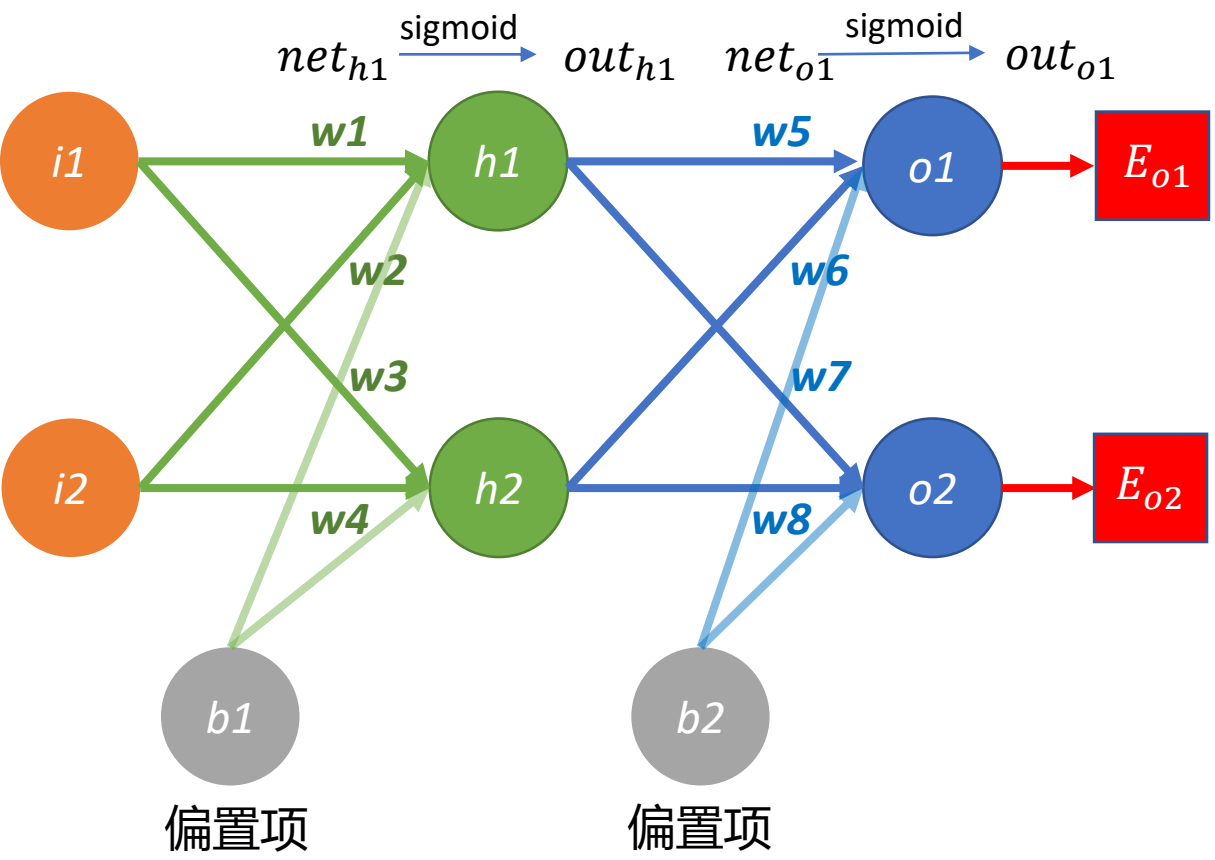
最后，更新*w1*权值：

$$w_1^+ = w_1 - \eta * \frac{\partial E\ total}{\partial w_1}$$
$$= 0.1 - 0.5 * 0.00009$$
$$= 0.099$$

同理，可更新*w2,w3,w4*

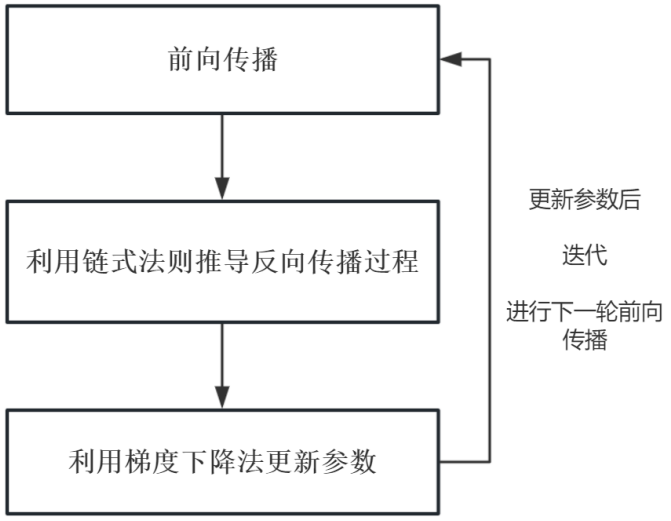
反向传播算法与梯度下降更新

Step3：迭代计算



- 第一次误差反向传播完成后，总误差*E(total)*由0.277下降至0.265；
- 把更新的权值重新计算，不停地迭代；
- 迭代10000次后，总误差为 0.0000636，输出为[0.008,0.991 (原真实标签为[0,1])，已经很接近了。

回顾：整体流程



ANN总结

- 神经网络模拟生物系统，其中学习涉及对神经元之间突触连接的调整。
- 输入灵活（可以处理相关性）。
- 输出可以是离散的，也可以是连续的。

(预测和分类)

- 训练时间长，容易陷入局部最优。
- 很难解释结果 (black box).

Batch

直观理解：

Batch Size定义：一次训练所选取的样本数。其影响模型的优化程度和速度。同时其直接影响到GPU内存的使用情况

- Why?

在小样本数的数据库中，不使用Batch Size是可行的，而且效果也很好。但是一旦是大型的数据库，一次性把所有数据输进网络，肯定会引起内存的爆炸。而且，在这情况下，计算得到不同梯度值差别巨大，难以使用一个全局的学习率，所以就提出Batch Size的概念。

Batch Size

- Batch Size 的影响

  - 没有Batch Size，梯度准确，只适用于小样本数据库
  - Batch Size=1，梯度变来变去，非常不准确，网络很难收敛。
  - Batch Size增大，梯度变准确，
  - Batch Size增大，梯度已经非常准确，再增加Batch Size也没有用
  - 注意：Batch Size增大了，要到达相同的准确度，必须要增大epoch。

Epoch, Iteration, Batch

- Epoch（轮/轮次）：
  - 一个Epoch就是将所有训练样本训练一次的过程。
  - 所有训练样本在神经网络中都进行了一次正向传播和一次反向传播
- Batch（批 / 一批样本）：
  - 将整个训练样本分成若干个Batch。
- Iteration（一次迭代）：
  - 训练一个Batch就是一次Iteration

- 为什么需要多个epoch?
  - 欠拟合 or 过拟合

Slide credit: L. Lazebnik

Reference

1. http://neuralnetworksanddeeplearning.com
2. http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/
3. https://zhuanlan.zhihu.com/p/390341772