反向传播算法 —— BackPropagation

前言

- 原文地址：http://www.cnblogs.com/charlotte77/p/5629865.html
- 作者：Charlotte77

首先，谢谢大神写的这么好的文章。

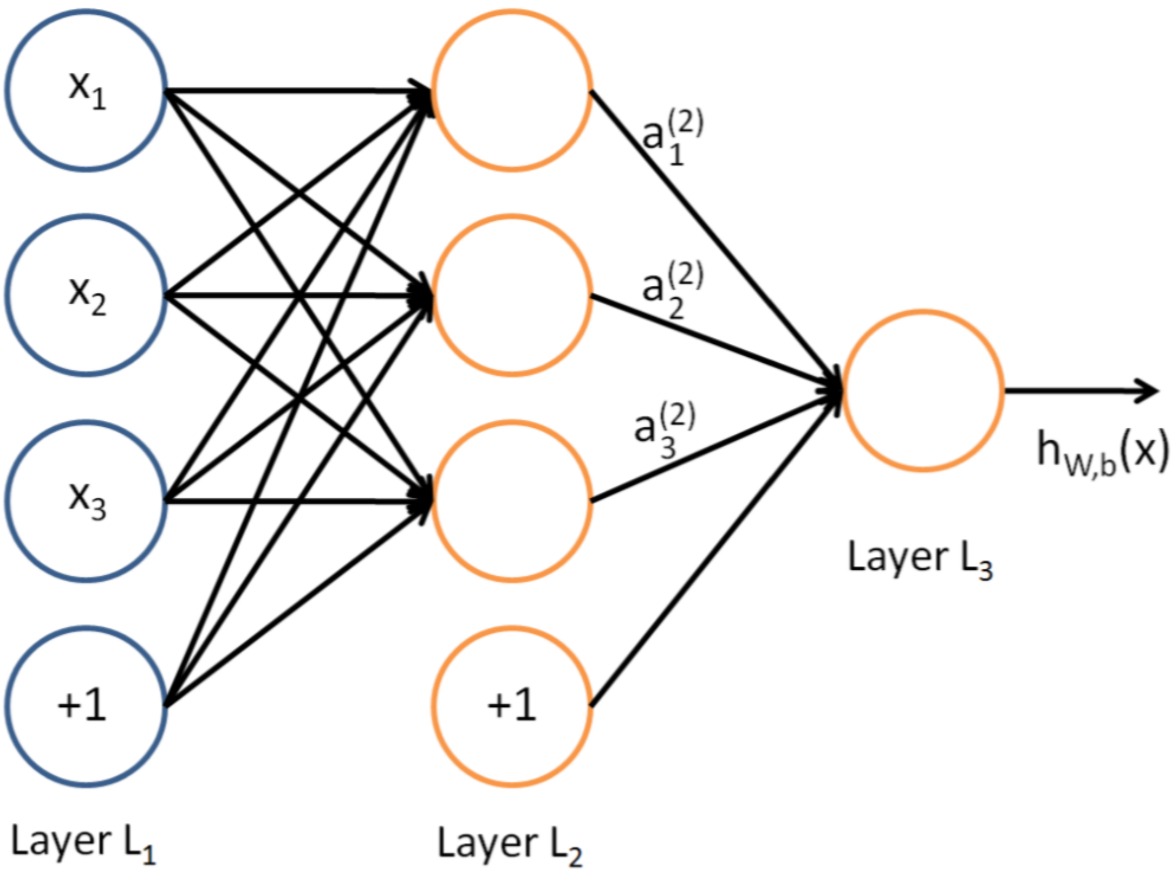
这个文章是我在网上看到的讲解 反向传播算法 最舒服的一个文章，没有之一。也可能是我比较菜吧，只能看懂写的清清楚楚的文章，而不是稍微有一些绕的文章。我一直感觉写那样的文章的作者，虽然知识搞得很熟悉，但是就是像茶壶里装饺子，倒不出来。

相反呢，这篇文章就是和我写文章和 blog 的一个风格，看起来超级爽，把什么都给你讲清楚，一个个把你心目里的问题全部解决完，让你最后都没有问题想问。这才是真正的大神啊，我就赶紧膜拜了一下。哈哈。接下来看文章吧。

简介

反向传播算法其实是神经网络的基础了，但是很多人在学的时候总是会遇到一些问题，或者看到大篇的公式觉得很难就退缩了。说实话，其实并不难，就是一个链式求导法则反复来用。如果不想看公式，我们可以实际地计算一下，体会一下这样一个过程，再来推导公式，这样就会觉得很容易了。

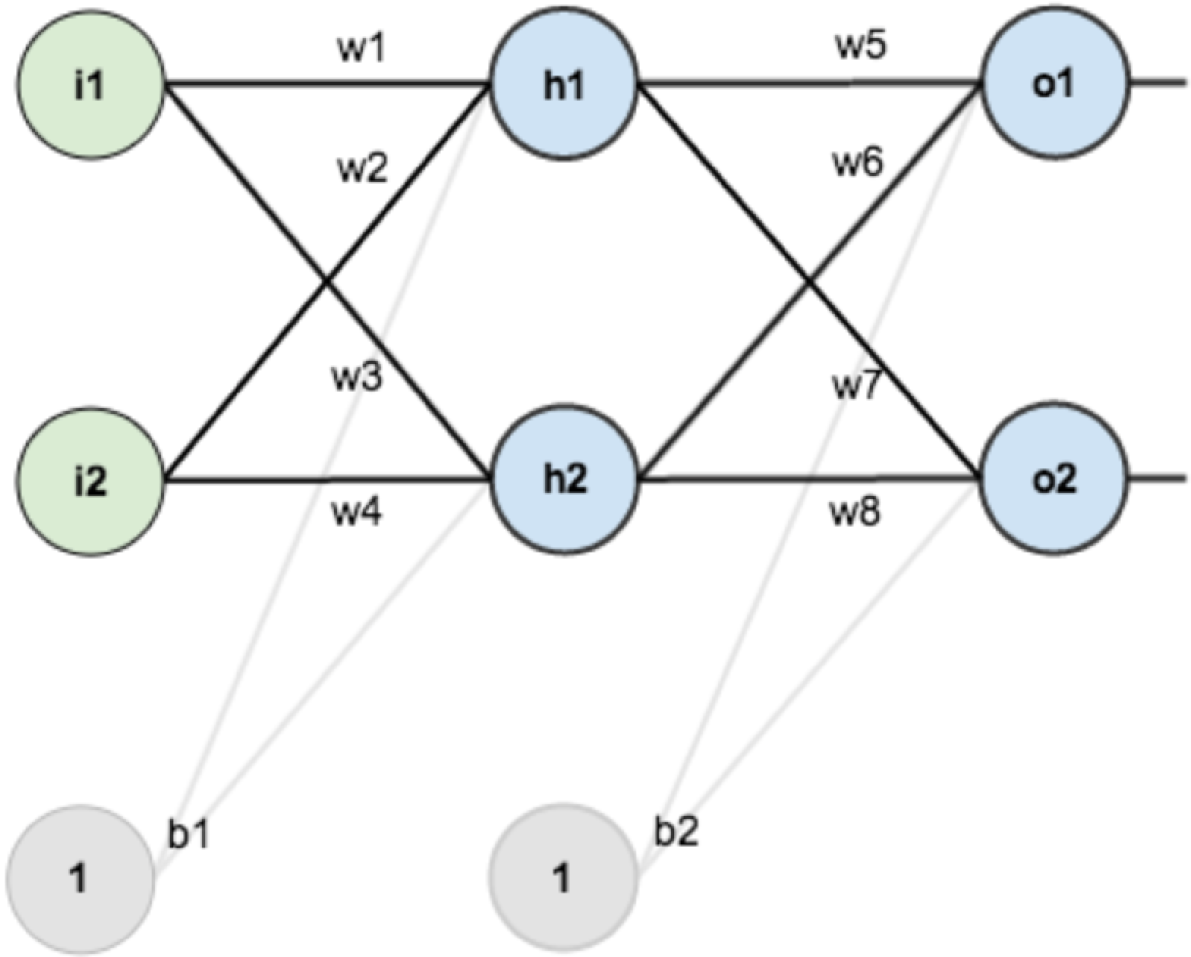
下面这个图片大家应该都不陌生：



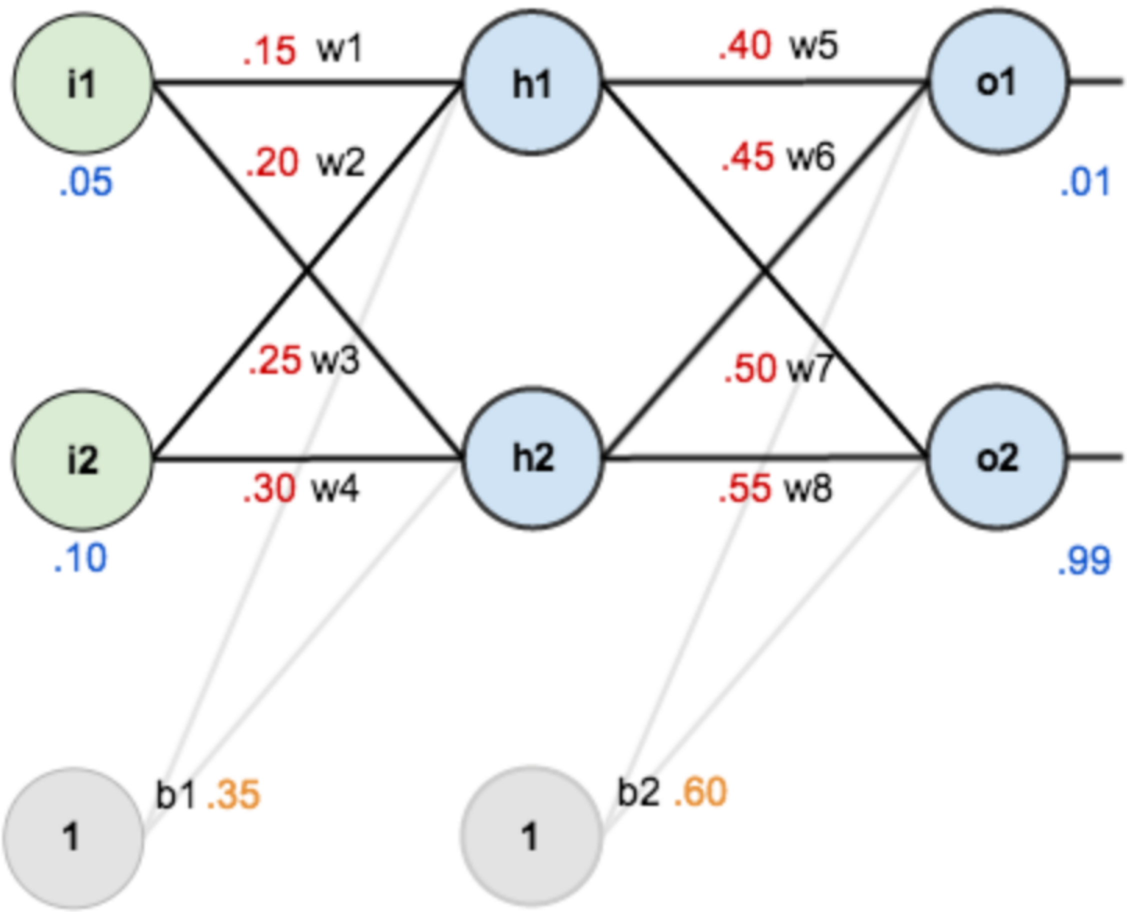
这个是比较典型的三层神经网络的基本构成，Layer L1 是输入层，Layer L2 是隐含层，Layer L3 是输出层。我们现在手里有一堆数据 {x1,x2,x3,...,xn} ,输出也是一堆数据 {y1,y2,y3,...,yn} ,现在要他们在隐含层做某种变换，让你把数据灌进去后得到你期望的输出。如果你希望你的输出和原始输入一样，那么就是最常见的自编码模型（Auto-Encoder）。可能有人会问，为什么要输入输出都一样呢？有什么用啊？其实应用挺广的，在图像识别，文本分类等等都会用到，我会专门再写一篇 Auto-Encoder 的文章来说明，包括一些变种之类的。如果你的输出和原始输入不一样，那么就是很常见的人工神经网络了，相当于让原始数据通过一个映射来得到我们想要的输出数据，也就是我们今天要讲的话题。

下面我们直接举一个例子，代入数值演示反向传播算法的过程，公式的推导等到下次写 Auto-Encoder 的时候再写，其实也很简单，感兴趣的同学可以自己推导下试试（注：本文假设你已经懂得基本的神经网络构成，如果完全不懂，可以参考 Poll写的笔记）。

假设，你有这样一个网络层：



第一层是输入层，包含两个神经元 i1， i2 和截距项 b1 ；第二层是隐含层，包含两个神经元 h1, h2 和截距项 b2 ；第三层是输出 o1, o2 ，每条线上标的 wi 是层与层之间连接的权重，激活函数我们默认为 sigmoid 函数。现在对他们赋上初值，如下图：



其中，输入数据 i1=0.05，i2=0.10;

输出数据 o1=0.01,o2=0.99;

初始权重 w1=0.15,w2=0.20,w3=0.25,w4=0.30;

w5=0.40,w6=0.45,w7=0.50,w8=0.55

目标：给出输入数据 i1， i2（0.05 和 0.10），使输出尽可能与原始输出 o1, o2 （0.01 和 0.99 ）接近。

Step 1 前向传播

1.输入层 -> 隐含层：

计算神经元 h1 的输入加权和：

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

神经元 h1 的输出 o1 ：（此处用到激活函数为 sigmoid 函数）：

$$out_{h1} = \frac{1}{1+e^{-het_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

同理，可以计算得出神经元 h2 的输出 o2 ：

$out_{h2} = 0.596884378$

2.隐含层 ----> 输出层

计算输出层神经元 o1 和 o2 的值：

$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$

$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$

$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$

$out_{o2} = 0.772928465$

这样前向传播的过程就结束了，我们得到输出值为 [0.75136079 , 0.772928465]，与实际值 [0.01 , 0.99] 相差还很远，现在我们对误差进行反向传播，更新权值，重新计算输出。

Step 2 反向传播

1.计算总误差

总误差：（square error）

$E_{total} = \sum \frac{1}{2}(target - output)^2$

但是有两个输出，所以分别计算 o1 和 o2 的误差，总误差为两者之和：

$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$

$E_{o2} = 0.023560026$

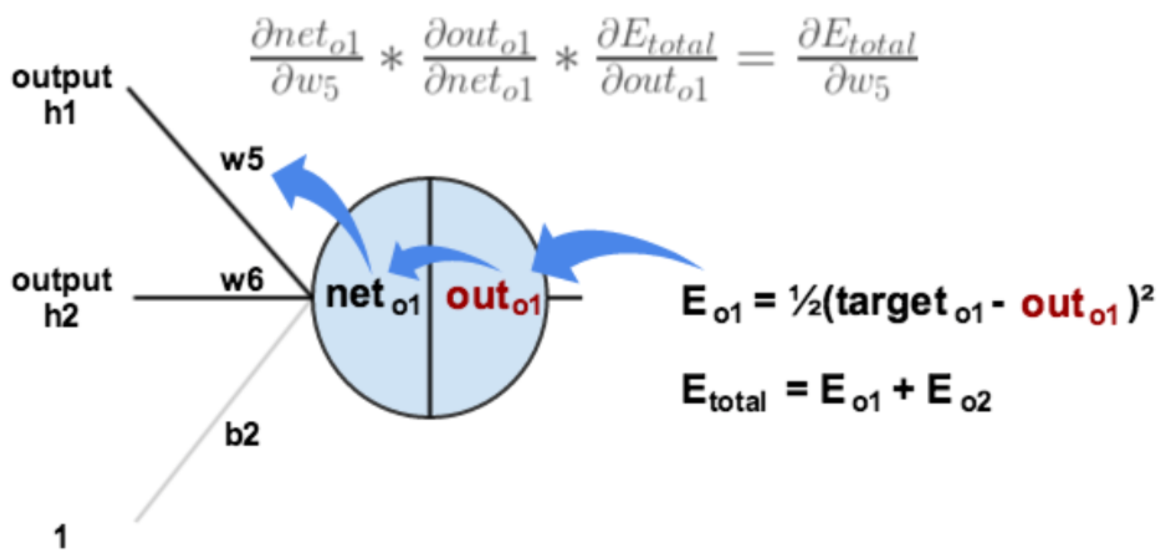
$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$

2.隐含层 ----> 输出层的权值更新

以权重参数 w5 为例，如果我们想知道 w5 对整体误差产生了多少影响，可以用整体误差对 w5 求偏导求出：（链式法则）

$$\frac{\partial E_{taotal}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

下面的图可以更直观的看清楚误差是怎样反向传播的：



现在我们来分别计算每个式子的值：

计算 $\frac{\partial E_{total}}{\partial out_{o1}}$ ：

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

计算 $\frac{\partial out_{o1}}{\partial net_{o1}}$ ：

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

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

这一步实际上就是对 sigmoid 函数求导，比较简单，可以自己推导一下。

计算 $\frac{\partial net_{o1}}{\partial w_5}$ ：

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5{}^{1-1} + 0 + 0 = out_{h1} = 0.593269992$$

最后三者相乘：

$$\frac{\partial E_{taotal}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{taotal}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

这样我们就计算出整体误差 E_{total} 对 w_5 的偏导值了。

回过头来再看看上面的公式，我们发现：

$$\frac{\partial E_{total}}{\partial w_5} = -(target_{o1} - output_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

为了表达方便，用 δ_{o1} 来表示输出层的误差：

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

$$\delta_{o1} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1})$$

因此，整体误差（E_{total}）对 w_5 的偏导公式可以写成：

$$\frac{\partial E_{total}}{\partial w_5} = \delta_{o1} out_{h1}$$

如果输出层误差计为负的话，也可以写成：

$$\frac{\partial E_{total}}{\partial w_5} = -\delta_{o1} out_{h1}$$

最后我们来更新 w_5 的值：

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

其中，η 是学习速率，这里我们取 0.5

同理，可更新 w_6, w_7, w_8 ：

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

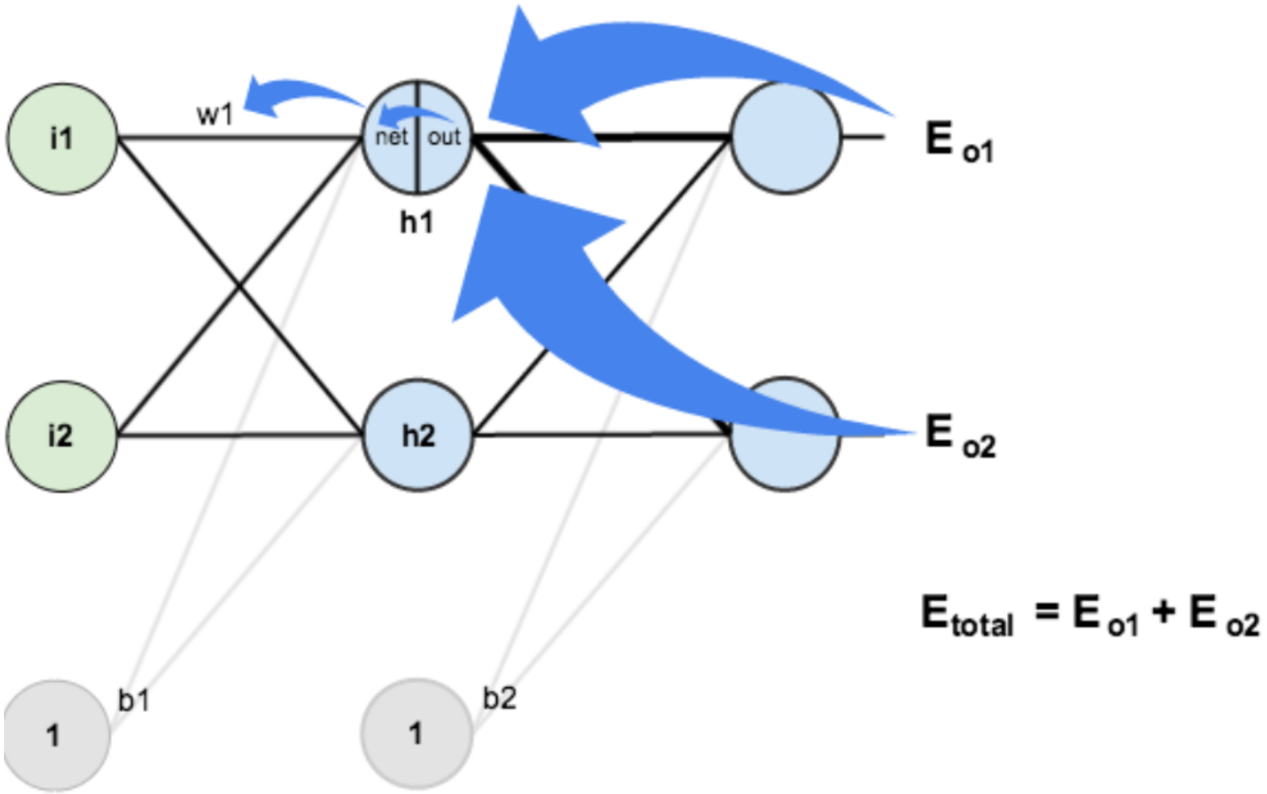
3.隐含层 ---> 隐含层的权值更新

方法其实与上面说的差不多，但是有个地方需要变一下，在上文计算总误差对 w_5 的偏导时，是从 $out_{o1}--->net_{o1}--->w_5$ ，但是在隐含层之间的权值更新时，是 $out_{h1}--->net_{h1}--->w_1$ ，而 out_{h1} 会接受 E_{o1} 和 E_{o2} 这两个地方传来的误差，所以这个地方两个都要计算。

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

$$\downarrow$$

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



计算 $\frac{\partial E_{total}}{\partial out_{h1}}$ ：

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

计算 $\frac{\partial E_{o1}}{\partial out_{h1}}$ ：

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

$$\frac{\partial E_{o1}}{\partial out_{h1}}=\frac{\partial E_{o1}}{\partial net_{o1}}*\frac{\partial net_{o1}}{\partial out_{h1}}=0.74136507*0.186815602=0.138498562$$

$$net_{o1}=w_5*out_{h1}+w_6*out_{h2}+b_2*1$$

$$\frac{\partial net_{o1}}{\partial out_{h1}}=w_5=0.40$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = 0.138498562 * 0.40 = 0.055399425$$

同理，计算得出：

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049119$$

两者相加得到总值：

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + -0.019049119 = 0.036350306$$

再计算 $\frac{\partial out_{h1}}{\partial net_{h1}}$ ：

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

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709$$

再计算 $\frac{\partial net_{h1}}{\partial w_1}$ ：

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

最后，三者相乘：

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

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

为了简化公式，用 δ_{h1} 表示隐含层单元 $h1$ 的误差：

$$\frac{\partial E_{total}}{\partial w_1} = (\sum_o \frac{\partial E_{total}}{\partial out_o} * \frac{\partial out_o}{\partial net_o} * \frac{\partial net_o}{\partial out_{h1}}) * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = (\sum_o \delta_o * w_{ho}) * out_{h1}(1 - out_{h1}) * i_1$$

$$\frac{\partial E_{total}}{\partial w_1} = \delta_{h1} i_1$$

最后，更新 w_1 的权值：

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

同理，可更新 w_2, w_3, w_4 的权值：

$$w_2^+ = 0.19956143$$

$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$

这样，误差的反向传播就完成了，最后我们再把更新的权值重新计算，不停地迭代，在这个例子中第一次迭代后，总误差 E_{total} 由 0.298371109 下降至 0.291027924 。迭代 10000 次后，总误差为 0.0000035085 ，输出为[0.015912196,0.984065734] (原输入为[0.01,0.99]),证明效果还是不错的。