深度学习系列（2）：前向传播和后向传播算法

深度学习系列（2）：前向传播和后向传播算法

前言

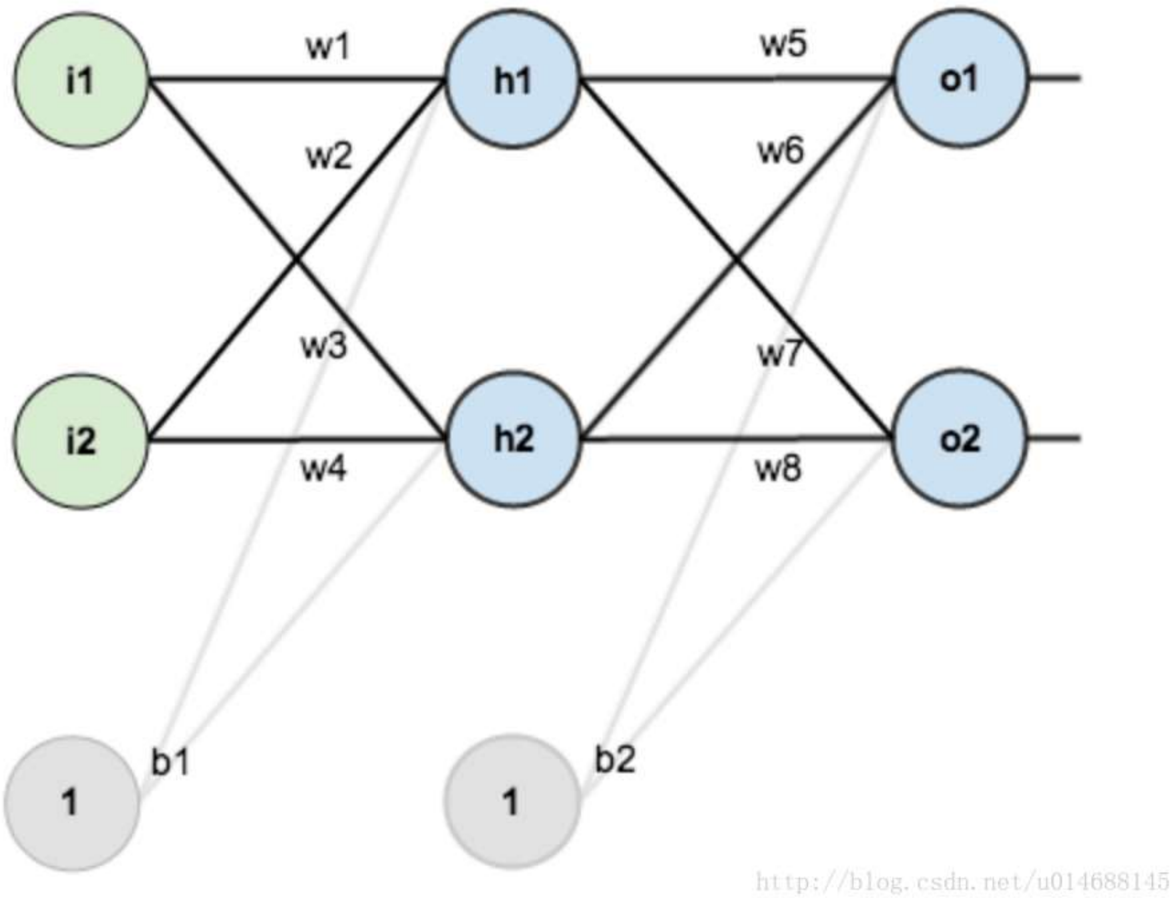
讲真，之前学吴恩达的机器学习课时，还手写实现过后向传播算法，但如今忘得也一干二净。总结两个原因：1. 理解不够透彻。2. 没有从问题的本质播的精髓。今天重温后向传播算法的推导，但重要的是比较前向传播和后向传播的优缺点，以及它们在神经网络 中起到了什么不一般的作用，才让迷。

反向传播的由来

反向传播 由Hinton在1986年发明，该论文发表在nature上，高尚大的杂志啊。

> Rumelhart, David E, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors. ” Nature 323.6088(1986):533-536.

简单说说吧，反向传播主要解决神经网络在训练模型时的参数更新问题。神经网络如下图：



反向传播算法需要解决每条边对应的权值如何更新，才能使得整个输出的【损失函数】最小。如果对神经网络还不了解，建议先学习了什么是神经网以下内容。

这里推荐几篇关于神经网络的文章，总体来说不错：

1. 计算机的潜意识
2. Machine Learning & Algorithm 神经网络基础

关于反向传播算法有种不太恰当的比方，对于每个输出结点，给定一个输入样例，会得到一个预测值，而这个预测值和真实值之间的差距我们当作误钱），是谁影响了欠债的多少呢？很明显，在神经网络模型 中，只有待求的参数$\{w_1, w_2, ..., w_n\}$了。如何衡量每个参数对误差的影响，我们定

度：当参数w_i在某个很小的范围内变动时，误差变动了多少，用数学表示即：$\frac{\Delta L}{\Delta w_i}$，在考虑极限情况下，即微分：$\frac{\partial L}{\partial w_i}$。

所以我们有了最基础的微分表达式，也是反向传播所有推导公式的源泉，那为什么这个敏感度就能更新权值呢？其实$\frac{\Delta L}{\Delta w_i}$很有意思，因为不管最终$L($是什么样子，$\frac{\Delta L}{\Delta w_i} = \text{定值}$，所以假设$\Delta w_i > 0$，那么该定值为负数的情况下，$w_i$增大的方向上$L(w_i)$将减小，而该定值为正数的情况时，$w_i$增大$L(w_i)$将增大。

所以梯度下降的更新算法有$w := w - \eta \frac{\partial L}{\partial w_i}$，当然你也可以画图形象的理解下，不难。

那么$\frac{\partial L}{\partial w_i}$这玩意怎么计算呢？在简单的感知机模型中很容易计算得到，具体可以参考上一篇博文，这里不再赘述了。

反向传播的计算

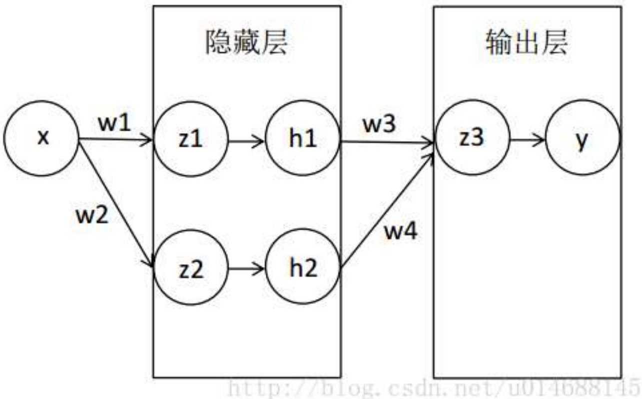
我很讨厌一上来就来了一堆反向传播的公式以及各种推导。这样没错，简单直接，理解了觉得自己还很牛逼，结果过了一段时间怎么又忘了公式的推新推一遍。而理解反向传播的精髓并非这些公式的推导，而是它弥补了前向算法的哪些不足，为啥它就被遗留下来作为神经网络的鼻祖呢？解决了什何优雅的解决了该问题？从哪些角度能让我们构建出反向传播算法才是应该去学习和理解的。

我们先来建个简单的神经网络图吧，注意，这里只是帮助理解反向传播算法的构建过程，与真实的神经网络有一定的差距，但其中的分析过程是大同

此外这三篇文章写的不错，【推导】【本质】【实现】都有了：

1. 【看看就行】机器学习：一步步教你理解反向传播方法
2. 【后续内容基于此文，推荐】Calculus on Computational Graphs: Backpropagation
3. 【python实现ANN，只要42行！】A Neural Network in 11 lines of Python (Part 1)

如图所示：



为了简化推导过程，输入层只使用了一个特征，同样输出层也只有一个结点，隐藏层使用了两个结点。注意在实际神经网络中，大多数文章把z1和h1点来画图的，这里为了方便推导才把两者分开。

所以我们有：
$z_1 = w_1 x$
$z_2 = w_2 x$
$h_1 = \frac{1}{1+e^{-z_1}}$
$h_2 = \frac{1}{1+e^{-z_2}}$
$z_3 = w_3 h_1 + w_4 h_2$
$y = \frac{1}{1+e^{-z_3}}$

假定给了输入x，我们就能根据这一系列公式求得y，接下来我们需要定义损失函数了，使用平方误差函数（只针对一次输入）：

$$L = \frac{1}{2}(y-t)^2$$

t表示真实值，ok，根据第一节的内容，模型训练实际上是更新w_i，既然要更新w_i，就需要求解$\frac{\partial L}{\partial w_i}$，于是对于$w_i$，根据链式法则，可以求得：

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_3}\frac{\partial z_3}{\partial h_1}\frac{\partial h_1}{\partial z_1}\frac{\partial z_1}{\partial w_1}$$

到这里你能看出什么？不着急，我们再求一个w_3：

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_3}\frac{\partial z_3}{\partial w_3}$$

从中，我们可以看到一些模式（规律），实际上w_1的更新，在它相关的路径上，每条边的后继和前继结点对应的就是偏导的分子和分母。w_3同样如 关边有三条（最后y指向L的关系边没有画出来），而对应的链式法则也恰好有三个偏导。

> **结论：每条关系边对应于一个偏导！！！什么是关系边？Okay，就是中间变量如z_1, h_1都与w_1有关系，连接这些结点的边。**

咱们继续细化上述公式，目前来看，这跟反向传播八竿子打不着。的确就这些性质不足以引出反向传播，不着急，继续往下看。

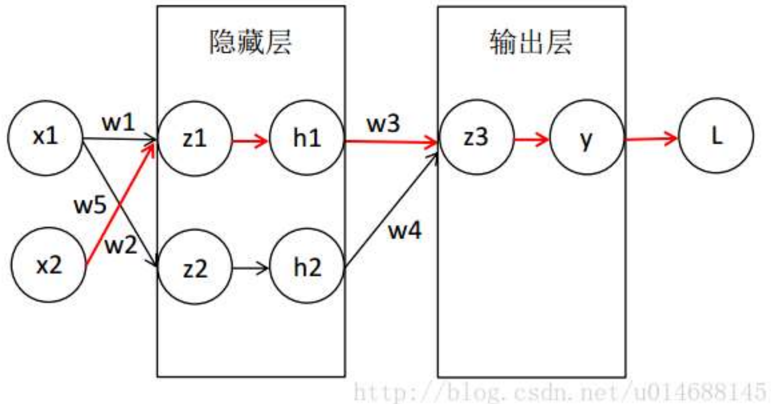
因为偏导数中的每个函数映射都是确定的，所以我们可以求出所有偏导数，于是有：

$$\frac{\partial L}{\partial w_1} = (y-t)\cdot y\cdot(1-y)\cdot w_3\cdot h_1\cdot(1-h_1)\cdot x$$

很有意思，式中x, t是由样本给定，而那些y, h_1, w_3都在计算y时，能够得到，这就意味着所有变量都是已知的，可以直接求出$\frac{\partial L}{\partial w_1}$，那怎么就有了前 【传播】之说呢？

宏观上，其实可以考虑一个非常大型的神经网络，它的参数w_i可能有成千上万个，难道对于每一个参数我们都要列出一个偏导公式么，显然不现实。 还需要进一步挖掘它们共通的模式。

继续看图：



假设我们加入第二个特征x_2，那么对应的w_5的更新，我们有如下公式：

$$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_3}\frac{\partial z_3}{\partial h_1}\frac{\partial h_1}{\partial z_1}\frac{\partial z_1}{\partial w_5}$$

对比一波w_1：

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_3}\frac{\partial z_3}{\partial h_1}\frac{\partial h_1}{\partial z_1}\frac{\partial z_1}{\partial w_1}$$

有什么不同么，眼神不好的还以为没有区别，实际上就最后一个偏导的分母发生了变化，而我们刚才也总结出了一个重要结论，每个偏导代表一条边 w_5的更新，前面四个偏导值都需要重新在计算一遍，也就是红线指出的部分，为了算w_5，需要重新再走过w_1的部分路径。

所以即使我们用输入(x_1, x_2)求出了每个结点，如$z_1, h_1, z_2, h_2, z_3, y$的值，为了求出每个$w_i$的偏导，需要多次代入这些变量，产生了大量的冗余， 上面也已经指出，每个w_i都需要手工求偏导么？庞大的神经网络太复杂了，之所以叫前向传播算法，是因为从输入(x_1, x_2)出发，能够求出对应的所 值，这个过程是正向的。

学过动态规划的可能一下子就能理解反向传播的精髓了，如果我们有个中间变量$\delta_j = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_3}\frac{\partial z_3}{\partial h_1}\frac{\partial h_1}{\partial z_1}$来存储，那么计算$w_1$和$w_5$时，只要对应的$\delta_j \cdot$ $\delta_j \cdot \frac{\partial z_1}{\partial w_5}$即可。那么中间的子状态只需要计算一次即可，而不是指数型增长。

这和递归记忆化搜索（自顶向下）以及动态规划（自底向上）的两种对偶形式很像，为了解决重复子问题，我们可以反向传播，如果能够定义出合适 且得出递推式那么这件事就做成了。

Okay，再来对比下w_1和w_3的偏导，继续找找规律吧：

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_3}\frac{\partial z_3}{\partial h_1}\frac{\partial h_1}{\partial z_1}\frac{\partial z_1}{\partial w_1}$$
$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_3}\frac{\partial z_3}{\partial w_3}$$

两个式子，找找相同的，只有前两部分是一样的，所以可以令$\delta^1 = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_3}$，这样的好处在于：

求w_3时，可以有：

$$\frac{\partial L}{\partial w_3} = \delta^1 \frac{\partial z_3}{\partial w_3}$$

求w_5时，可以有：

$$\frac{\partial L}{\partial w_1} = \delta^1 \frac{\partial z_3}{\partial h_1}\frac{\partial h_1}{\partial z_1}\frac{\partial z_1}{\partial w_1}$$

从图上来理解的话，δ^1 表示【聚集】在z_3的误差，为啥到z_3呢，因为在这里刚好可以求出w_3的偏导，从公式上理解的话就是那公共部分（重复子问

既然这么定义了，我们可以同样定义第二层的误差δ_1^2 表示【聚集】在z_1的误差。δ_2^2表示【聚集】在z_2的误差。所以有：

$$\begin{aligned}\delta_1^2 &= \delta^1 \frac{\partial z_3}{\partial h_1}\frac{\partial h_1}{\partial z_1} \\ &= \delta^1 \cdot w_3 \cdot \frac{\partial h_1}{\partial z_1}\end{aligned}$$

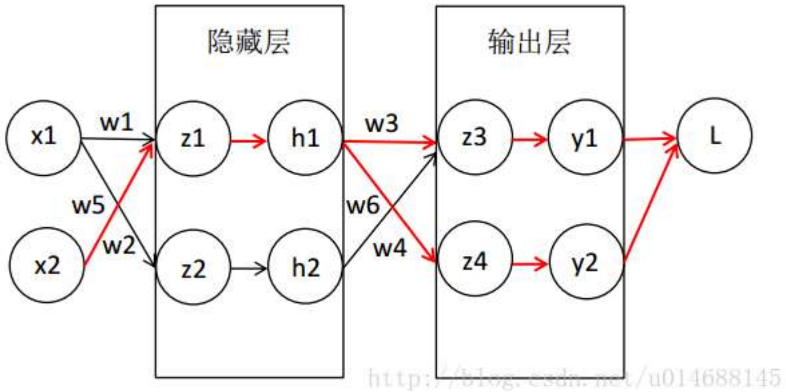
对应地w_1的偏导公式就可以有$\frac{\partial L}{\partial w_1} = \delta_1^2 \frac{\partial z_1}{\partial w_1}$

哈哈哈，对比一波w_1, w_5, w_3，可以得到：

$$\frac{\partial L}{\partial w_1} = \delta_1^2 \frac{\partial z_1}{\partial w_1}$$
$$\frac{\partial L}{\partial w_5} = \delta_1^2 \frac{\partial z_1}{\partial w_5}$$
$$\frac{\partial L}{\partial w_3} = \delta^1 \frac{\partial z_3}{\partial w_3}$$

别迷糊了，它们都属于同一种形式，写算法就好些很多，而δ^2是由δ^1加上对应的w_i求得，形象了吧，所以我们首要的目标是求出最后一层的δ^1，接着
前一层的权值w_i求出前一层每个结点的δ^2，更新公式都一样，δ^2乘以上一层的输出值而已，谁叫 $y = h_1 w_1 + h_2 w_2$是线性的呢，求偏导h_1得到w_1
w_1得h_1，这实在太巧妙了。

这就对了吗？不，离真正的反向传播推导出的公式还差那么一点点，继续看图：



我们按照关系边的概念，可以知道w_5的关系边应该由红色的边组成。所以δ_1^2的更新不仅仅只跟z_3有关系了，还和z_4有关。为什么？此时损失函数由
成，对应一个输入样例(x1, x2)，有：

$$L = \frac{1}{2}(y_1 - t_1)^2 + \frac{1}{2}(y_2 - t_2)^2$$

所以对L求偏导，由加法法则，可以得到$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial y_1} + \frac{\partial L}{\partial y_2}$，没错，多个结点指向同一个结点时，把它们的偏导值加起来即可（损失函数就这么定义
$\delta_j^2 = \frac{\partial h_j}{\partial z_j}\sum w_{ji} \cdot \delta_i^1$.

此时再看看完整的反向传播公式推导吧，或许就明白其中缘由了。参考链接：http://blog.csdn.net/u014313009/article/details/51039334