机器学习：一步步教你理解反向传播方法

2016年09月12日 机器学习 机器学习 字数:5596

> *在阅读反向传播方法的时候，看到了这篇通过示例给出反向传播的博文A Step by Step Backpropagation Example，在这篇博文中，作者通过一个简单的示例，给出了反向传播的过程的过程，非常的清晰，遂结合自己的理解翻译之，希望对反向传播方法有所理解的朋友有些许帮助。*

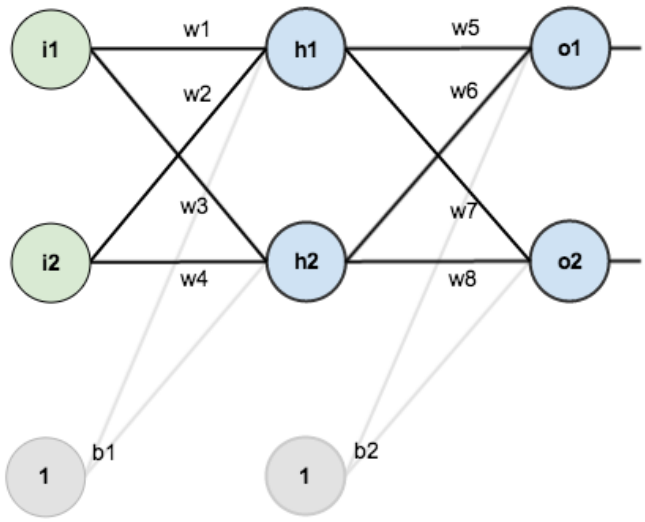
背景

反向传播在神经网络的训练过程中虽然用得如此之多，但是在网上还很少有通过具体的实例来解释反向传播怎么工作的博文。所以在这篇文章中，我会尝试用一个具体的例子来解释反向传播过程，这样有需要的朋友就可以通过自己的计算过程来判断自己对于反向理解的过程是否到位。

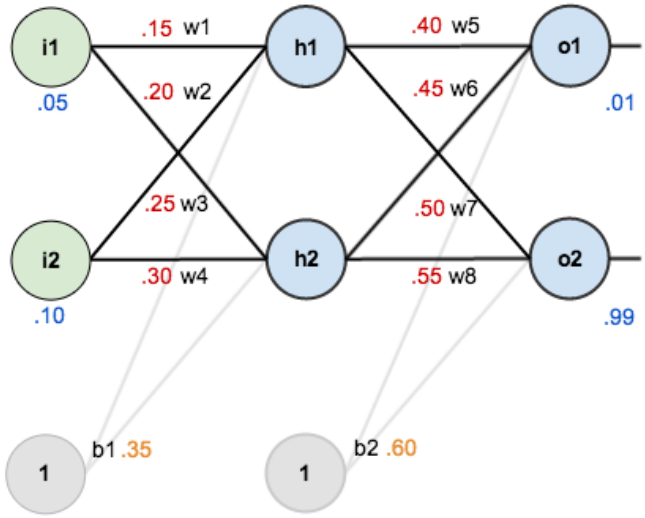
你可以在我的Gihub上找个我写的反向传播的Python实现代码。

概览

在这篇博文中，我们会使用有2个输入单元的神经网络，2个隐层神经元以及2个输出神经元。此外，隐层和输出神经元会包含一个偏置，下面是基本的网络结构：



为了便于后面说明的说明，我们对该网络设置一些初始的权重、偏置以及输入和输出：



反向传播的目标是对权重进行优化，使得神经网络能够学习到从任意的输入到输出的准确映射。

在这篇博文中，我们仅使用一个简单的训练集，即输入为0.05和0.10，我们希望网络的输出为0.01和0.99(即输入的样本是两个: (0.05, 0.99), (0.10, 0.99))。

前向传播

首先来看看对于给定的初始化权重和偏置，网络对于输入0.05和0.10的输出是啥。我们将输入输进网络中。

我们先计算从全部网络的输入到隐层的每一个神经元，激活函数采用logistic函数，对于从隐层到输出层，我们重复这一过程。

> *全部的网络输入也被称为网络的输入Derivation of Backpropagation*

下面是对于h_1全部网络输入的输入计算过程：

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$
$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

(译者注：类比到CNN网络里，这个过程就是卷积过程，得到特征响应图)

然后我们将其输入到激活函数中，得到输出h_1：

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

(译者注：类比到CNN网络里，这个过程特征响应图经过激活函数运算的过程)

对于h_2通过上面相同的过程，我们可以得到：

$$out_{h2} = 0.596884378$$

对于输入层神经元，将隐层的输出作为输入(译者注：在CNN中，还需要经过池化后才能作为下一层的输入，至于为啥需要池化，这里译者不就解释了)，重复上面相同的过程，我们可以得到：

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$
$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$
$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}} = \frac{1}{1 + e^{-1.105905967}} = 0.75136507$$

同样的，重复上面相同的过程，可以得到O_2:

$$out_{O2} = 0.772928465$$

计算总误差

现在对于输出的每一个神经元，使用平方误差函数求和来计算总的误差:

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

> *output就是我们的预测label，而target就是groundtruth。$\frac{1}{2}$使得我们在求骗到的时候可以消去2，不影响模型参数的结果求解。*

对于第一个神经元的输出$O1$真实值是0.01，而网络的输出是0.75136507， 因而第一个神经元的输出误差为:

$$E_{O1} = \frac{1}{2}(target - output)^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

重复上面过程，可以得到第二个神经元的输出$O2$为:

$$E_{O2} = 0.023560026$$

所以整个神经网络的误差求和为:

$$E_{total} = E_{O1} + E_{O2} = 0.274811083 + 0.023560026 = 0.298371109$$

反向传播

反向传播的目标是：通过更新网络中的每一个权重，使得最终的输出接近于groundtruth，这样就得到整个网络的误差作为一个整体进行了最小化。

输出层

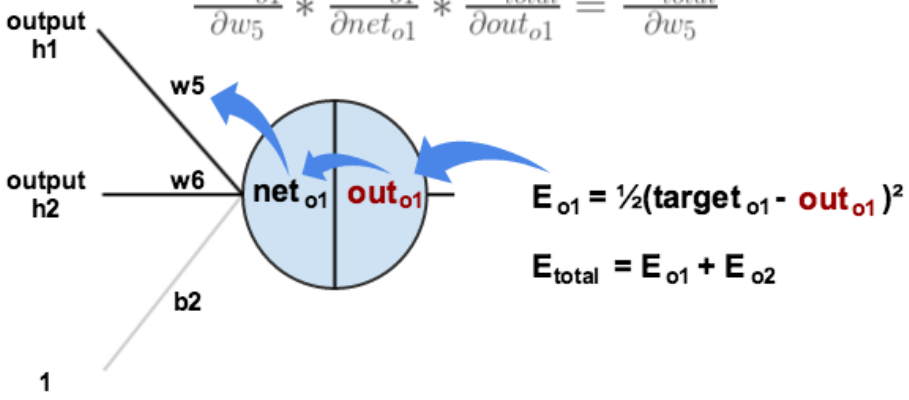
先来考察w_5，我们想知道对于w_5的改变可以多大程度上影响总的误差，也就是$\frac{\partial E_{total}}{\partial w_5}$。

通过使用链式法则，可以得到:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{O1}}{\partial net_{O1}} * \frac{\partial net_{O1}}{\partial w_5}$$

为了更直观的表述上面链式法则的过程，对其进行可视化:

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



我们对上面使用链式法则得到的每一项分别进行计算。首先，整体误差关于各个神经元的输出改变了？

$$E_{total} = \sum \frac{1}{2}(target - output)^2 = \frac{1}{2}(target_{O1} - output_{O1})^2 + \frac{1}{2}(target_{O2} - output_{O2})^2$$

$$\frac{\partial E_{total}}{\partial out_{O1}} = 2 * \frac{1}{2}(target_{O1} - output_{O1})^{2-1} * -1 + 0 = -(target_{O1} - output_{O1}) = -(0.01 - 0.75136507) = 0.74136507$$

logistic函数的偏导数为输出乘以1减去输出，即：

$$out_{O1} = \frac{1}{1 + e^{-net_{O1}}}$$

$$\frac{\partial out_{O1}}{\partial net_{O1}} = out_{O1}(1 - out_{O1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

最后，整个网络的输入$O1$关于w_5改变了多少呢？

$$net_{O1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{O1}}{\partial net_{O1}} * \frac{\partial net_{O1}}{\partial w_5}$$

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

> *你也会看到用delta规则表示的形式：*
>
> $$\frac{\partial E_{total}}{\partial w_5} = -(target_{O1} - out_{O1}) * out_{O1}(1 - out_{o1}) * out_{h1}$$
>
> *我们可以将$\frac{\partial E_{total}}{\partial out_{O1}}$和$\frac{\partial out_{O1}}{\partial net_{O1}}$写为$\frac{\partial E_{total}}{\partial net_{O1}}$，并用$\delta_{O1}$表示它，从而可以将上面的式子表示为：*
>
> $$\delta_{O1} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{O1}}{\partial net_{O1}}$$
>
> $$\delta_{O1} = -(target_{O1} - out_{O1}) * out_{O1}(1 - out_{o1})$$
>
> *因此有：*
>
> $$\frac{\partial E_{total}}{\partial w_5} = \delta_{O1} out_{h1}$$
>
> *有一些论文中通过将负号从δ中提出来将其也可以写为下面这种形式：*
>
> $$\frac{\partial E_{total}}{\partial w_5} = -\delta_{O1} out_{h1}$$

为了减小误差，我们将w_5原来的值减去目前的权重(通常会乘上一个学习率η，这里我们将其设置为0.5)：

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5}$$

> *学习率在不同的文章中可以记法不一样，有用α的，有用η的，有用ϵ的。*

重复上面的过程，我们可以得到更新后的w_6、w_7和w_8：

$$w_6^+ = 0.408666186 w_7^+ = 0.511301270 w_8^+ = 0.561370121$$

注意，在我们继续向前推进反向传播的时候，在要使用到w_5、w_6、w_7和w_8的地方，我们仍然使用的是原来的权重，而不是更新后的权重。

隐层

我们继续推进反向传播来计算w_1、w_2、w_3和w_4更新的权重：

同样使用链式法则，我们可以得到：

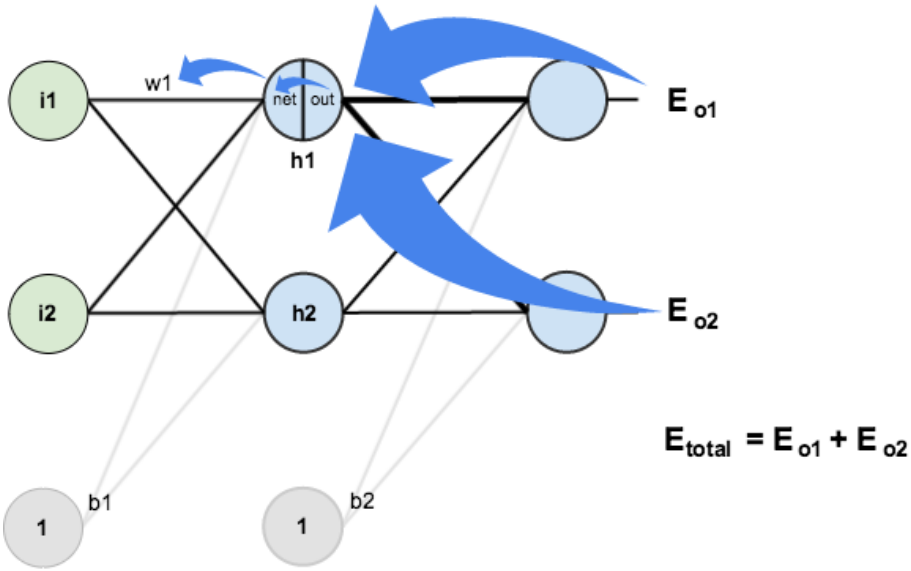
$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

可视化上面的链式法则：

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

$$\downarrow$$

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



对于这一层(隐层)的更新我们采用上面输出层相似的处理方式，不过会稍有不同，这种不同主要是因为每一个隐层神经元的输出对于最终的输出都是有贡献的。我们知道out_{h1}既影响out_{O1}也影响out_{O2}，因此$\frac{\partial E_{total}}{\partial out_{h1}}$需要同时考虑到这两个输出神经元影响：

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{O1}}{\partial out_{h1}} + \frac{\partial E_{O2}}{\partial out_{h1}}$$

又由于：

$$\frac{\partial E_{O1}}{\partial out_{h1}} = \frac{\partial E_{O1}}{\partial net_{O1}} * \frac{\partial net_{O1}}{\partial out_{h1}}$$

我们可以用前面计算的值来计算$\frac{\partial E_{O1}}{\partial net_{O1}}$:

$$\frac{\partial E_{O1}}{\partial net_{O1}} = \frac{\partial E_{O1}}{\partial out_{O1}} * \frac{\partial out_{O1}}{\partial net_{O1}} = 0.74136507 * 0.186815602 = 0.138498562$$

又因为$\frac{\partial net_{O1}}{\partial out_{h1}}$等于$w_5$:

$$net_{O1} = w_5 * out_{h1} + w_6 * out_{h2} + b2 * 1$$
$$\frac{\partial net_{O1}}{\partial out_{h1}} = w_5 = 0.40$$

将上面每步分开算的结果合起来得:

$$\frac{\partial E_{O1}}{\partial out_{h1}} = \frac{\partial E_{O1}}{\partial net_{O1}} * \frac{\partial net_{O1}}{\partial out_{h1}} = 0.138498562 * 0.40 = 0.055399425$$

与上面的步骤一样，我们可以得到:

$$\frac{\partial E_{O2}}{\partial out_{h1}} = w_5 = -0.019049119$$

因此:

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{O1}}{\partial out_{h_1}} + \frac{\partial E_{O2}}{\partial out_{h1}} = 0.055399425 + (-0.019049119) = 0.036350306$$

现在我们已经有了$\frac{\partial E_{total}}{\partial out_{h1}}$，我们还需要为每一个需要更新的权重计算$\frac{\partial out_{h1}}{\partial net_{h1}}$和$\frac{\partial net_{h1}}{\partial w}$:

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

如我们前面对于输出神经元所做的一样，我们计算$h1$的全部输入关于w_1求偏导:

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$
$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

将上面计算的各个部分合起来:

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$
$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

> *你可以可能会看到下面的这种写法:*
>
> $$\frac{\partial E_{total}}{\partial w_1} = (\sum_O \frac{\partial E_{total}}{\partial out_O} * \frac{\partial out_O}{\partial net_O} * \frac{\partial net_O}{\partial out_{h1}}) * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$
> $$\frac{\partial E_{total}}{\partial w_1} = (\sum_O \delta_O * w_{hO}) * out_{h1}(1 - out_{h1}) * i$$
> $$\frac{\partial E_{total}}{\partial w_1} = \delta_{h1} i_1$$

现在我们可以更新w_1：

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

根据上面相同的计算过程，我们可以得到w_2、w_3和w_4：

$$w_2^+ = 0.19956143 w_3^+ = 0.24975114 w_4^+ = 0.29950229$$

现在，我们已经更新了所有的权重，在最初，在我们的输入为0.05和0.1的时候，网络的误差为0.298371109， 经过第一次方向传播后，网络的误差降低到了0.291027924。虽然看起来下降得不是很多，但是在重复这个过程10000次以后，网络的误差就下降到了0.000035085。这个时候，当我们把0.05和0.1再输入进去，两个神经元的输出为0.015912196(vs 0.01)和0.984065734(vs 0.99)。

如果你在读上面的博文的时候发现有任何错误，不要犹豫告诉我。如果你有更清楚的想读者讲解的方式，不要犹豫给我留言。谢谢！

请我喝杯咖啡

←机器学习：随机梯度下降法

机器视觉：特征序列化→