问题

ReLU函数在0处不可导，为什么在深度学习网络中还这么常用?

问题背景

这是在阿里的机器学习岗一面的时候问的一个问题，最开始的问题是“为什么机器学习中解决回归问题的时候一般使用平方损失（即均方误差）？”。

当时我的回答是损失函数是是模型预测值与真实值之间的一种距离度量，我们可以计算出每个样本的预测值与真实值之间的距离，全部加起来就得到了所谓的损失函数。而距离的度量可以采用预测值与真实值之间差的绝对值，或者两者之差的平方，当然更高次的也行，只要你喜欢。正如问题所述，为什么我们一般使用的是两者之差的平方而不是两者只差的绝对值呢？其实这与模型的求解相关，举最简单的线性回归为例，如果采用的距离是两者之差的绝对值，那么求解的目标函数如下：

$$(\omega^*, b) = argmin_{(\omega,b)} \sum_{i=1}^{m} |f(x_i) - y_i| \tag{1}$$

如果采用的距离是两者之差的平方，那么求解的目标函数如下：

$$(\omega^*, b) = argmin_{(\omega,b)} \sum_{i=1}^{m} (f(x_i) - y_i)^2 \tag{2}$$

其中：$f(x_i) = \omega x_i + b$ 即预测值，y_i 为真实值，m 为样本总数，ω 和 b 为要求解的参数

要求得使以上损失函数最小化对应的那个 ω 和 b ，可将损失函数对 ω 和 b 求导，并令导数为0。但是当采取的距离是两者之差的绝对值时，函数在0处不可导，且还增加的一个工作量是需要判断 $f(x_i) - y_i$ 正负号。而采用的距离是两者之差的平方时就没有这方面的问题，所以解决回归问题的时候一般使用平方损失。但理论上两者都可以使用，只是如果用两者之差的绝对值的话，那么需要判断和处理的东西多点，例如人为设定0处的导数为0等等。

（以上是我当时给面试官的答案，目前还没验证对错，权当参考）

回答完以上问题之后，面试官就自然而然地引出了本节要讨论的问题“ReLU函数在0处不可导，为什么在深度学习网络中还这么常用?“

问题解答

ReLU函数在0处确实不可导，但是在实际中却被大量使用也是事实，为什么？因为真的好用，10亿AI调参侠都在用，用了都说好。但它在0处不可导，怎么办？**其实我们可以人为提供一个伪梯度，例如给它定义在0处的导数为0，其实tensorflow在实现ReLU的时候也是定义ReLU在0处的导数为0的。**

另外还有一个方法是使用 $ln(1 + e^x)$ 来近似，这个函数是连续的，它在0点的导数是0.5。也就是相当于relu在0点的导数取为0.5，也正好是0和1的均值。

这里参考的资料有：
一、relu不可微为什么可用于深度学习
二、激活函数RELU在0点的导数是多少？

拓展

ReLU的好用体现在哪呢？下面阐述下使用ReLU函数的优势

1. ReLU函数的形式非常简洁，`ReLU = max(0, x)`，是由两段线性函数（大于0部分是线性的，小于0部分也是线性的，但是组合起来后却是非线性的）组成的非线性函数，函数形式看似简单，但ReLU的组合却可以逼近任何函数。
2. 其实ReLU提出的最大作用是解决sigmoid函数导致的**梯度消失**问题的（这里对于梯度消失就不拓展，留作下个问题再具体探讨），所以ReLU的优势大部分是与它的死对头sigmoid函数对比体现出来的，对比这两个函数的图形可以看出：ReLU有单侧抑制，即Relu会使一部分神经元的输出为0，这样就造成了网络的稀疏性，并且减少了参数的相互依存关系，缓解了过拟合问题的发生。另外这也更符合生物神经元的特征。
3. ReLU的运算速度快，这个很明显了，max肯定比幂指数快的多。

当然ReLU函数也是有缺点的：可能会导致神经元死亡，权重无法更新的情况。这种神经元的死亡是不可逆转的死亡

解释：训练神经网络的时候，一旦学习率没有设置好，第一次更新权重的时候，输入是负值，那么这个含有ReLU的神经节点就会死亡，再也不会被激活。因为：ReLU的导数在x>0的时候是1，在x<=0的时候是0。如果x<=0，那么ReLU的输出是0，那么反向传播中梯度也是0，权重就不会被更新，导致神经元不再学习。
也就是说，这个ReLU激活函数在训练中将不可逆转的死亡，导致了训练数据多样化的丢失。在实际训练中，如果学习率设置的太高，可能会发现网络中40%的神经元都会死掉，且在整个训练集中这些神经元都不会被激活。所以，设置一个合适的较小的学习率，会降低这种情况的发生。为了解决神经元节点死亡的情况，有人提出了Leaky ReLU、P-ReLu、R-ReLU、ELU等激活函数。

这里参考的资料有：
[一、激活函数及其作用以及梯度消失、爆炸、神经元节点死亡的解释]

By Yee

2020.05.07