**Thinking1 什么是反向传播中的链式法则**

在微积分中，链式法则是计算两个或多个函数的组合的导数的公式，也就是说，如果f和g是函数，则链式法则表示它们的组合的导数f∘g（将x映射到f（g（x））的函数。由于在定义过程中求导公式可以表示成一个连乘过程，就像锁链一样一环套一环，故而得名。

以误差（Error）为主导的反向传播（Back Propagation）运动，旨在得到最优的全局参数矩阵，进而将多层神经网络应用到分类或者回归任务中去。前向传递输入信号直至输出产生误差，反向传播误差信息更新权重矩阵。权重得以在信息双向流动中得到优化。

因为梯度下降需要每一层都有明确的误差才能更新参数，比如逻辑回归（Logistic Regression），我们可以把它看做没有隐藏层的网络；但对于多隐藏层的神经网络，输出层可以直接求出误差来更新参数，但其中隐藏层的误差是不存在的，因此这里不可以直接应用梯度下降（Gradient Descent），而是要先将输出层的误差反向传播给隐藏层，再应用梯度下降，其中将误差从末层往前传递的过程需要链式求导法则（Chain Rule）来计算权重以及偏置的偏导，因此反向传播算法可以说是梯度下降在链式法则中的应用。

**Thinking2 请列举几种常见的激活函数，激活函数有什么作用**

* sigmoid激活函数：常用于在二分类问题中，将输出结果映射到[0,1]。均值不为０；经常与二分类交叉熵损失一起使用。
* tanh激活函数：值域：(-1, 1)，均值是０。比于Sigmoid函数更容易训练，具有优越性；通常与L2损失一起使用。
* relu激活函数：对于线性函数而言，ReLU的表达能力更强，尤其体现在深度网络中；而对于非线性函数而言，ReLU由于非负区间的梯度为常数，因此不存在梯度消失问题，使得模型的收敛速度维持在一个稳定状态。

作用：激活函数是用来加入非线性因素的，可以强化网络的学习能力，因为线性模型的表达能力不够。

sigmoid和tanh函数输出值在(0,1)和(-1,1)之间 => 适合处理概率值；会产生梯度消失 => 不适合深层网络训练

relu的有效导数是常数1，解决了深层网络中出现的梯度消失问题 => 更适合深层网络训练

**Thinking3 利用梯度下降法训练神经网络，发现模型loss不变，可能有哪些问题？怎么解决？**

* 训练集loss不下降

训练集的loss在训练过程中迟迟不下降，一般是由这几个方面导致的：

1.模型结构和特征工程存在问题

如果一个模型的结构有问题，那么它就很难训练，通常，自己“自主研发”设计的网络结构可能很难适应实际问题，通过参考别人已经设计好并实现和测试过的结构，以及特征工程方案，进行改进和适应性修改，可以更快更好的完成目标任务。当模型结构不好或者规模太小、特征工程存在问题时，其对于数据的拟合能力不足，是很多人在进行一个新的研究或者工程应用时，遇到的第一个大问题。

2.权重初始化方案有问题

神经网络在训练之前，我们需要给其赋予一个初值，但是如何选择这个初始值，则要参考相关文献资料，选择一个最合适的初始化方案。常用的初始化方案有全零初始化、随机正态分布初始化和随机均匀分布初始化等。合适的初始化方案很重要，用对了，事半功倍，用不对，模型训练状况不忍直视。博主之前训练一个模型，初始化方案不对，训练半天都训练不动，loss值迟迟居高不下，最后改了初始化方案，loss值就如断崖式下降。

3.正则化过度

L1 L2和Dropout是防止过拟合用的，当训练集loss下不来时，就要考虑一下是不是正则化过度，导致模型欠拟合了。一般在刚开始是不需要加正则化的，过拟合后，再根据训练情况进行调整。如果一开始就正则化，那么就难以确定当前的模型结构设计是否正确了，而且调试起来也更加困难。

4.选择合适的激活函数、损失函数

不仅仅是初始化，在神经网络的激活函数、损失函数方面的选取，也是需要根据任务类型，选取最合适的。

比如，卷积神经网络中，卷积层的输出，一般使用ReLu作为激活函数，因为可以有效避免梯度消失，并且线性函数在计算性能上面更加有优势。而循环神经网络中的循环层一般为tanh，或者ReLu，全连接层也多用ReLu，只有在神经网络的输出层，使用全连接层来分类的情况下，才会使用softmax这种激活函数。

而损失函数，对于一些分类任务，通常使用交叉熵损失函数，回归任务使用均方误差，有自动对齐的任务使用CTC loss等。损失函数相当于模型拟合程度的一个评价指标，这个指标的结果越小越好。一个好的损失函数，可以在神经网络优化时，产生更好的模型参数。

5.选择合适的优化器和学习速率

神经网络的优化器选取一般选取Adam，但是在有些情况下Adam难以训练，这时候需要使用如SGD之类的其他优化器。学习率决定了网络训练的速度，但学习率不是越大越好，当网络趋近于收敛时应该选择较小的学习率来保证找到更好的最优点。所以，我们需要手动调整学习率，首先选择一个合适的初始学习率，当训练不动之后，稍微降低学习率，然后再训练一段时间，这时候基本上就完全收敛了。一般学习率的调整是乘以/除以10的倍数。不过现在也有一些自动调整学习率的方案了，不过，我们也要知道如何手动调整到合适的学习率。

6.训练时间不足

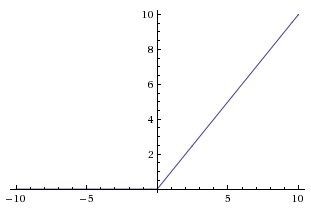
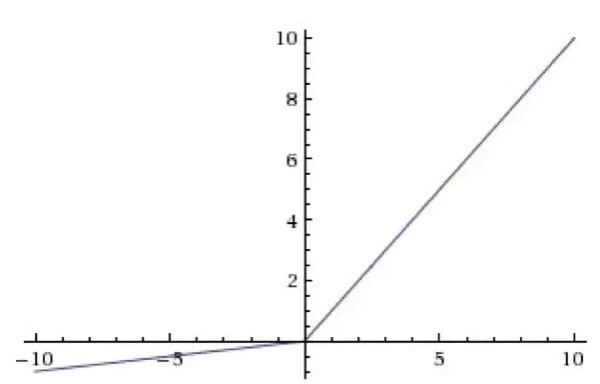
我有时会遇到有人问这样的问题，为什么训练了好几个小时了，怎么loss没降多少，或者怎么还没收敛。心急吃不了热豆腐！各种深度学习的训练都有不同的计算量，当需要的计算量很大时，怎么可能几个小时就训练完，尤其是还在使用自己的个人电脑CPU来训练模型的情况下。一般解决方案就是，使用更快的硬件加速训练，比如GPU，在涉及到计算机视觉方面的任务时，加速效果显著，主要是卷积网络的缘故。当已经没有办法使用硬件来加速的时候，唯一的解决方案就是——等。

7.模型训练遇到瓶颈

这里的瓶颈一般包括：梯度消失、大量神经元失活、梯度爆炸和弥散、学习率过大或过小等。

梯度消失时，模型的loss难以下降，就像走在高原上，几乎任何地方都是高海拔，可以通过梯度的检验来验证模型当前所处的状态。有时梯度的更新和反向传播代码存在bug时，也会有这样的问题。

在使用Relu激活函数的时候，当每一个神经元的输入X为负时，会使得该神经元输出恒为0，导致失活，由于此时梯度为0，无法恢复。有一种解决方案是使用LeakyRelu，这时，Y轴的左边图线会有一个很小的正梯度，使得神经网络在一定时间后可以得到恢复。不过LeakyRelu并不常用，因为部分神经元失活并不影响结果，相反，这种输出为0还有很多积极的作用。因为Relu方程输入为负时，输出值为0，利用此特性可以很好地忽略掉卷积核输出负相关信息，同时保留相关信息。

Relu FunctionLeakyRelu Function

梯度爆炸和梯度弥散产生的根本原因是，根据链式法则，深度学习中的梯度在逐层累积。如1.1的n次方无穷大，0.9的n次方无穷小。网络中某些层过大的输出也会造成梯度爆炸，此时应该为该输出取一个上界，可用最大范数约束。

关于学习率，可按照第5项内容进行调整。

8.batch size过大

batch size过小，会导致模型后期摇摆不定，迟迟难以收敛，而过大时，模型前期由于梯度的平均，导致收敛速度过慢。一般batch size 的大小常常选取为32，或者16，有些任务下比如NLP中，可以选取8作为一批数据的个数。不过，有时候，为了减小通信开销和计算开销的比例，也可以调整到非常大的值，尤其是在并行和分布式中。

9.数据集未打乱

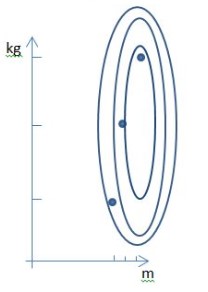
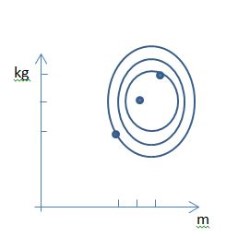
不打乱数据集的话，会导致网络在学习过程中产生一定的偏见问题。比如张三和李四常常出现在同一批数据中，那么结果就是，神经网络看见了张三就会“想起”李四。主要原因是，梯度更新时，总时张三和李四的梯度平均，导致出现固定的梯度走向，使得数据集的丰富度降低了。数据打乱后，张三还会跟王五一起平均，李四也会跟王五一起平均，那么梯度的走向会更丰富，也能更好地在整个数据集上学习到最有用的隐含特征。

10.数据集有问题

当一个数据集噪声过多，或者数据标注有大量错误时，会使得神经网络难以从中学到有用的信息，从而出现摇摆不定的情况。就像有人告诉你1+1=2，还有人告诉你1+1=3时，就会陷入困惑。或者还有可能时读取数据出错导致，这时实际情况跟数据标注错误是一样的。另外，类别的不平衡也会使得较少类别由于信息量不足，难以习得本质特征。

11.未进行归一化

未进行归一化会导致尺度的不平衡，比如1km和1cm的不平衡，因此会导致误差变大，或者在同样的学习率下，模型会以秒速五厘米的步伐，左右两边摇摆不定地，向前走1km。有时，不平衡是由于不同的度量单位导致的，比如kg和m，我们都知道，1kg和1m没有什么可比性，虽然数字都是1。因此，我们完全可以通过放缩，使得特征的数值分布更接近一些。

归一化前归一化后

12.特征工程中对数据特征的选取有问题

数据特征的选取不合理，就像数据标注错误一样，会使得神经网络难以找到数据的本质特征进行学习。而机器学习的本质就是在做特征工程，以及清洗数据(逃)。

* 验证集loss不下降

验证集的loss不下降分为两种。一种是训练集上的loss也不下降，这时问题主要在训练集的loss上，应当先参考上述方法解决。另一种是训练集上的loss可以下降，但验证集上的loss已经不降了，这里我们主要说明这种情况下的问题。

由于验证集是从同一批训练数据中划分出来的，所以一般不存在数据集的问题，所以主要是过拟合。过拟合解决起来，其实不怎么复杂，无非就几种方法，但是对工程师自身的条件要求比较高。

1.适当的正则化和降维

正则化是用来解决模型过拟合问题的一个很重要的手段，博主之前在文章《机器学习：过拟合与欠拟合问题》里面写到过，比如通过增加一个正则项，并且人为给定一个正则系数lambda，进行权重衰减，将一些相关性不大的特征项的参数衰减到几乎为0，相当于去掉了这一项特征，这跟降维类似，相当于减少了特征维度。而去掉基本无关的维度，那么就避免了模型对于这一维度特征的过分拟合。还有在神经网络两个层之间增加Dropout和Normal等，也起到了抑制过拟合的作用。

2.适当降低模型的规模

过拟合很重要的一个原因也是模型的复杂度太高，就像一亩地只种了一棵小麦，那么其他地方不种就会长杂草，于是就会过拟合了一些噪声。所以，除了正则化手段以外，适当减小模型的规模也是很重要的，尽量让神经网络结构的假设空间与预期目标模型需要存储的信息量相匹配。

3.获取更多的数据集

这是终极解决方案，深度学习就是在有大量数据的基础上发展起来的。深度学习的三件套：数据、模型和硬件。模型可以直接拿来用，硬件可以花钱买，但是数据需要一点一点去收集，而且很多问题的解决就依赖于大量的数据，没数据就没有一切。

4.对数据集做扰动和扩增

这个是直接对现有的数据集做扩容，一定程度上可以再次提高验证集上的准确率，比如对图像做旋转，对声音文件进行加噪处理等。最终的效果虽然比不上同等情况下的数据量的增加带来的效果增益，但是在现有条件下，算是扩增数据量的一个有效的方案。

* 测试集loss不下降

测试集一般为模型之前训练时从未见过的新数据，或者目标应用场景下的真实数据。由于训练集和验证集的loss不下降时，应归为前两节的内容，所以这一节中，我们默认训练集和验证集的loss情况是正常的。所以，如果测试集的loss很高，或者正确率很低，那么一般是因为训练数据的分布和场景与测试数据的分布和应用场景不一致。

1.应用场景不一致

比如，一个语音识别模型，输入的数据集都是女性的录音音频，那么对于男性的声音就不能很好的识别出来。这个也是博主之前做语音识别的时候遇到过的一个真实案例，解决方案就是增加含有大量男性录音音频的数据集来训练。

2.噪声问题

噪声问题是实际应用场景下，频繁遇到的问题。直接容易理解的案例就是，在语音识别中，标准语音数据集都是在安静环境下采集的数据，但是在实际应用中，我们录音时多多少少会有噪声，那么我们需要专门去处理噪声，比如进行一个降噪处理，或者在训练数据中添加噪声等。在图像的识别中，那么就需要考虑图片中的遮挡、雾霾、旋转、镜像和大小远近等问题。