深度前馈网络与Xavier初始化原理

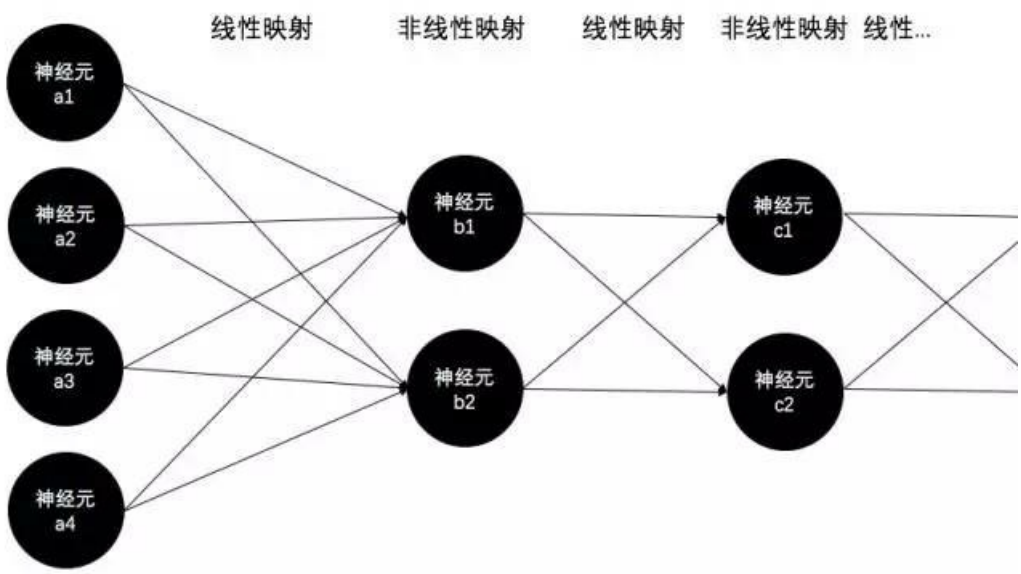
原创 夕小瑶 夕小瑶的卖萌屋 2017-07-17

来自专辑

卖萌屋@深度学习炼丹技巧

基本的神经网络的知识（一般化模型、前向计算、反向传播及其本质、激活函数等）小夕已经介绍完毕，本文先讲一下深度前馈网络的BP过程，再基于此来重点讲解在前馈网络中用来初始化model参数的Xavier方法的原理。

前向过程很简单，每一层与下一层的连接边(即参数)构成了一个矩阵(即线性映射)，每一层的神经元构成一个激活函数阵列（即非线性映射），信号便从输入层开始反复的重复这两个过程直到输出层，也就是已经在《神经网络激活函数》中详细讲过的线性与非线性交替映射的过程：



误差反向传播的过程也很简单，其本质就是基于链式求导+梯度下降，原因已在《BP算法的本质》中讲解，这里就重点讲解一下BP算法的过程。

首先往上翻一翻，记住之前说过的前馈网络无非就是反复的线性与非线性映射。然后：

首先，假设某个神经元的输入为z，经过激活函数f1(·)得到输出a。即函数值a=f1(z)。如果这里的输入z又是另一个函数f2的输出的话（当然啦，这里的f2就是线性映射函数，也就是连接某两层的权重矩阵），即z=f2(x)，那么如果基于a来对z中的变量x求导的时候，由于

$$\frac{\partial a}{\partial x} = \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial x} = f_1'(z) \cdot \frac{\partial z}{\partial x}$$

显然只要乘以激活函数f1的导数，就不用操心激活函数的输出以及更后面的事儿了（这里的“后面”指的是神经网络的

输出端方向），只需要将精力集中在前面的东西，即只需要关注z以及z之前的那些变量和函数就可以了。因此，误差反向传播到某非线性映射层的输出时，只需要**乘上该非线性映射函数在z点的导数**就算跨过这一层啦。

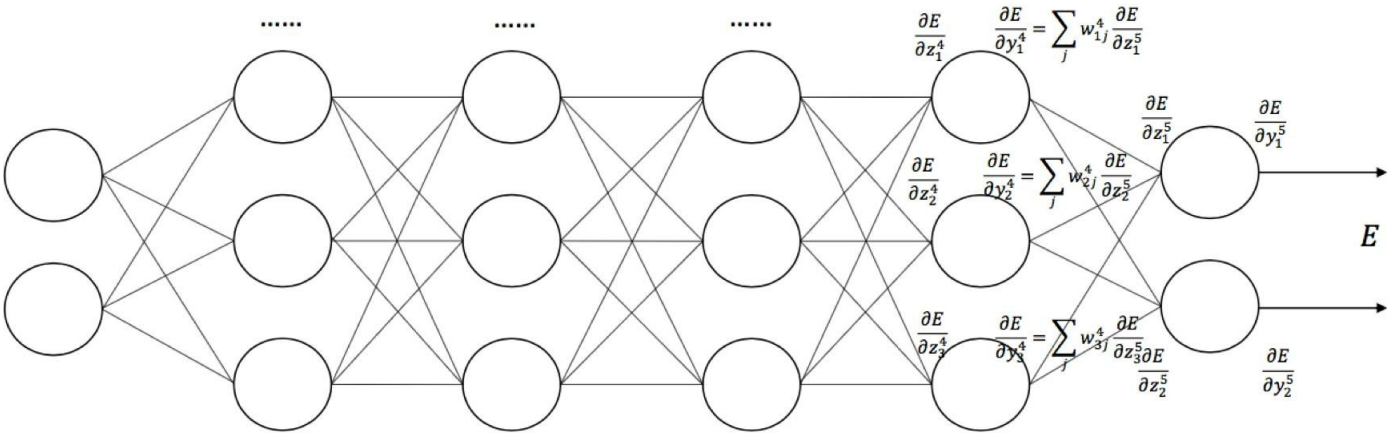
而由于f2(·)是个线性映射函数，即f2(x)=w\cdot x，因此

$$\frac{\partial z}{\partial x} = f_2'(x) = [wx]' = w$$

因此，当误差反向传播到线性映射层的输出时，若想跨过该层，只需要**乘以线性映射函数的参数**就可以啦~即乘上w。

而这里的x，又是更后面的非线性映射层的输出，因此**误差在深度前馈网络中反向传播时，无非就是反复的跨过非线性层和线性层，也就是反复的乘以非线性函数的导数(即激活函数的导数)和线性函数的导数（即神经网络的参数/权重/连接边）**。

也就是下面这张图啦（从右往左看）：



(图片来自中科院自动化所赵军老师的课件)

前向与反向都讲解完了，就到了至关重要的工程trick环节了。限于篇幅，本文只讲一下xavier初始化方法来初始化模型参数。其余tricks交给其他文章啦。

如果无脑的将model的参数初始化成全0，那训练到天荒地老也是个废猫倒了。

而为什么初始化成全0就不行了呢？这个不用讲啦，全0的时候每个神经元的输入和输出没有任何的差异，换句话说，根据前面BP算法的讲解，这样会导致误差根本无法从后一层往前传（**乘以全0的w后误差就没了**），这样的model当然没有任何意义。

那么我们不把参数初始化成全0，那我们该初始化成什么呢？换句话说，如何既保证输入输出的差异性，又能让model稳定而快速的收敛呢？

要描述“差异性”，首先就能想到概率统计中的方差这个基本统计量。对于每个神经元的输入z这个随机变量，根据前面讲BP时的公式，它是由线性映射函数得到的，也就是$z=\sum_{i=1}^{n} w_i x_i$，其中n是上一层神经元的数量。因此，根据概率统计里的两个随机变量乘积的方差展开式：

$$Var(w_i x_i) = E[w_i]^2 Var(x_i) + E[x_i]^2 Var(w_i) + Var(w_i)\ Var(x_i)$$

可以得到，如果E(xi)=E(wi)=0（可以通过批量归一化Batch Normalization来满足，其他大部分情况也不会差太多），那么就有：

$$Var(z) = \sum_{i=1}^{n} Var(xi) Var(wi)$$

如果随机变量xi和wi再满足独立同分布的话：

$$Var(z) = \sum_{i=1}^{n} Var(xi) Var(wi) = nVar(w) Var(x)$$

好了，这时重点来了。

试想一下，根据文章《激活函数》，整个大型前馈神经网络无非就是一个超级大映射，将原始样本稳定的映射成它的类别。也就是将样本空间稳定的映射到类别空间。试想，如果样本空间与类别空间的分布差异很大，比如说类别空间特别稠密，样本空间特别稀疏辽阔，那么在类别空间得到的用于反向传播的误差丢给样本空间后简直变得微不足道，也就是会导致模型的训练非常缓慢。同样，如果类别空间特别稀疏，样本空间特别稠密，那么在类别空间算出来的误差丢给样本空间后简直是爆炸般的存在，即导致模型发散震荡，无法收敛。因此，我们要让样本空间与类别空间的分布差异（密度差别）不要太大，**也就是要让它们的方差尽可能相等**。

因此为了得到Var(z)=Var(x)，只能让n*Var(w)=1，也就是**Var(w)=1/n**。

同样的道理，正向传播时是从前往后计算的，$Var(w) = 1/(n_{in})$，反向传播时是从后往前计算的，$Var(w) = 1/(n_{out})$。然而n_{in}和n_{out}往往不相等啊，怎么办呢？所以就取他们的均值就好啦~即：

令 $Var(w) = \frac{2}{n_{in} + n_{out}}$

假设w为均匀分布的话，由w在区间[a,b]内均匀分布时的方差为

$$Var = \frac{(b-a)^2}{12}$$

代码前面的Var(w)公式解得参数a和b，即得：

$$w \sim U[-\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}]$$

（让w在这个区间里均匀采样就好啦）

得到的这个结论就是**Xavier初始化方法**。这就是为什么使用Xavier初始化这个trick经常可以让model的训练速度和分类性能取得大幅提高啦~所以在使用前馈网络时，除非你的网络设计的明显不满足xavier的假设，否则使用xavier往往不会出错。当然，另一方面来说，也很少有场合可以完全迎合xavier假设，因此时间充裕的话，改改分子，甚至去掉n_{out}都有可能带来意想不到的效果。

参考文献：Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks[J]. Journal of Machine Learning Research, 2010, 9:249-256.