深度学习里数学之----方差----美妙而富有韵味

本文是对 Bengio 的论文《Understanding the difficulty of training deep deedforward neural networks》解读的第二部分。

主要的内容:

1、 深度神经网络梯度的学习（梯度消失）输入方差、权重方差、梯度方差的关系

在继上一篇《如何优雅地对深度神经网络进行训练》之后，相信大家对隐含层中激活函数以及不同激活函数带来的影响有了一定的认识。本文将结合激活函数、梯度学习、损失函数、前向传播进一步研究如何优雅地进行深度网络训练，让我们更加明白在训练过程中，梯度消失的问题、如何解决梯度消失的问题，以及到底哪些东西在发生变化，如何变化，这些变化对最终的模型有着怎样的影响。

接《如何优雅地对深度神经网络进行训练》

4.梯度学习及其传播

4.1 损失函数的影响

我们发现逻辑回归或者条件对数似然函数损失函数(-logP(y|x)结合 softmax 输出)相比于平方误差函数会工作(对于分类问题)的更好，平方误差函数过去常常在神经网络的前向传播中使用。这是之前就发现的一个结论，但是我们发现有必要在这里强调一下。我们发现，在以对数似然函数为损失函数的深度神经网络中比较少的出现学习停滞的现象。我们可以从图 5 中看出，将训练准则描绘成一个两层网络（一个隐含层），也就是说这是一个具有两层权重的函数，网络的激活函数是双曲正弦函数，以及随机的输入信号和目标信号。平方误差函数有着更为严重的平稳期（也就是学习停滞的时间持续很长）。

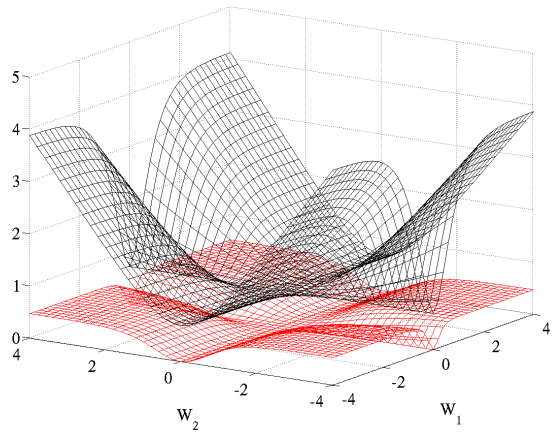

图5:交叉熵（黑色曲面，图上面部分），平方差损失（红色曲面，图下面部分）。神经网络有两层，每一层一个权重。W1、W2分别是第一层、输出层的权重。

（下面的论文内容堪称经典-细细品读，就会有收获）

对于复杂多层人工神经网络，使用在 0 处导数为 1 的对称激活函数(例如 f'(0)=1)。用z^i表示第 i 层的激活向量值，s^i表示激活函数的参数向量，$s^i = z^i w^i + b^i$, $z^{i+1} = \mathrm{f}(s^i)$。

从这些定义中，我们可以得到:

$$\frac{\partial Cost}{\partial s_{l,k}^i} = f'(s_k^i) W_{k,\cdot}^{i+1} \frac{\partial Cost}{\partial s^{i+1}} (1)$$

$$\frac{\partial Cost}{\partial w_{l,k}^i} = z_l^i \frac{\partial Cost}{\partial s^i} (2)$$

方差将由输入、输出、权重随机初始化表示。考虑这样一个假设：权重初始化是线性机制的，也就是说权重是独立随机初始化的，输入特征的方差是相同的。通过这些假设，我们可以说，对于输入是 x 且含有 n^i个神经元 i 层:

$$\mathrm{f}'(s_k^i) \approx 1 (3)$$

$$\mathrm{Var}[z^i] = \mathrm{Var}[\mathrm{x}] \prod_{i'=0}^{i-1} n_{i'} Var[w^{i'}] (4)$$

我们用$Var[w^{i'}]$表示i'层所有权重的方差，那么对于一个 d 层网络，有

$$Var\left[\left[\frac{\partial Cost}{\partial s^i}\right] = Var\left[\frac{\partial Cost}{\partial s^d}\right] \prod_{i'=i}^{d} n_{i'+1} Var[w^{i'}] (5)$$

$$\mathrm{Var}[\frac{\partial Cost}{\partial w^i}] = \prod_{i'=0}^{i-1} n_{i'} Var\left[w^{i'}\right] \prod_{i'=i}^{d-1} n_{i'+1} Var\left[w^{i'}\right] \times \mathrm{Var}[\mathrm{x}] Var\left[\frac{\partial Cost}{\partial s^d}\right] (6)$$

从前向传播的角度来看，想要信息（梯度信息）能够流动，我们需要如下要求：

$$\forall(\mathrm{i}, i'), \mathrm{Var}[z^i] = \mathrm{Var}[z^{i'}] (7)$$

从反向传播的角度来看，我们需要如下要求：

$$\forall(\mathrm{i}, i'), Var\left[\frac{\partial Cost}{\partial s^i}\right] = Var\left[\frac{\partial Cost}{\partial s^{i'}}\right] (8)$$

综上两个条件转化为如下：

$$\forall(\mathrm{i}, i'), \mathrm{n}_i \mathrm{Var}[W^i] = 1 (9)$$

$$\forall(\mathrm{i}, i'), \mathrm{n}_{i+1} \mathrm{Var}[W^i] = 1 (10)$$

作为这两个条件的折中选择，我们可以得到如下：

$$\forall(\mathrm{i}), \mathrm{Var}\left[W^i\right] = \frac{2}{n_i + n_{i+1}} (11)$$

注意到当所有层含有相同的宽度时（每一层的神经元数量相同）这两个条件是如何满足的。同时，如果我们对于权重使用相同的初始化方法，我们将会得到如下有趣的性质：

$$\forall \mathrm{i}, Var\left[\frac{\partial Cost}{\partial s^i}\right] = nVar[W]^{d-i} - Var[x] (12)$$

$$\forall \mathrm{i}, Var\left[\frac{\partial Cost}{\partial w^i}\right] = \left[nVar[W]\right]^d Var[x] Var[\frac{\partial Cost}{\partial s^d}] (13)$$

我们从上面的公式可以看出所有层的权重梯度的方差是相同的，但对于深度神经网络，反向传播的梯度，可能会出现梯度消失或者激增。

按照$W_{ij} \sim U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$初始化的方法得到的方差有如下性质：

$$nVar[W] = \frac{1}{3} (14)$$

这里 n 是层大小（假设所有层都有相同的大小）。这将会引起每层反向传播的梯度方差不是独立的（逐渐下降）。

因此，鉴于层层之间的乘积形式，当深度神经网络初始化时，规范化因素显得十分重要，我们建议按照如下的初始化方式能够满足在梯度传播时保持激活函数值方差不变的要求。我们称之为规范化初始化：

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}\right] (15)$$

让我们一起来细细评味大神世界的梯度流动，相信看完后，你会觉得----“这世界好美妙”。

公示(1、2、3)很好理解，重点说一下公示(4):

首先，我们知道$s = w * x + b$，这里 x 是输入，w、b 是权重与偏置。如果 x,w 独立同分布$U[-\sigma,\sigma]$,那么根据统计学我们知道$s \sim U[-\sigma_w\sigma_x, \sigma_w\sigma_x]$,

$s = \sum_{i=0}^{l} w_i * x_i + b_i$，我们可以得到:

$s^1 = x$

s 的均值是 0;

s 的方差是: $var[s^2] = (\sigma_w^1 * \sigma_x^1) + (\sigma_w^1 * \sigma_x^1) + (\sigma_w^1 * \sigma_x^1) \ldots + (\sigma_w^1 * \sigma_x^1)$

$$var[s^2] = n_1 * \sigma_w^1 * \sigma_x^1 (16)$$

这里 n 是某一层神经元的数量（某层的 size）。

对于$s^{i+1} = \sum_{j=0}^{l} w_j * s_j^i + b_j$，有:

$$var[s^{i+1}] = n_i * \sigma_w^i * \sigma_s^i (17)$$

根据数学归纳法，进一步推导出:

$var[s^{i+1}] = n_i * \sigma_w^i * n_{i-1} * \sigma_w^{i-1} * n_{i-2} * \sigma_w^{i-2} * n_{i-3} * \sigma_w^{i-3} * \ldots n_1 * \sigma_w^1 * \sigma_x^1$

$= (n_i * \sigma_w^i) * (n_{i-1} * \sigma_w^{i-1}) * (n_{i-2} * \sigma_w^{i-2}) * (n_{i-3} * \sigma_w^{i-3}) * \ldots (n_1 * \sigma_w^1) * \sigma_x^1 (18)$

我们得到论文的公式:

$$\mathrm{Var}[x] = \sigma_x^1 (19)$$

$$\mathrm{Var}[s^i] = \mathrm{Var}[x] \prod_{i'=0}^{i-1} n_{i'} Var[w^{i'}] (20)$$

同理得到公式(6、7)。

此时，为了保证梯度能够正常的流动，且根据级数定理知识我们假设公式(9)(10)成立，因为无论$\prod_{i'=0}^{i-1} n_{i'} Var[w^{i'}]$还是$\prod_{i'=0}^{i-1} n_{i'} Var[\frac{\partial Cost}{\partial s^{i'}}]$，如果其值大于 1 时，就会发散，造成梯度激增问题；而如果小于 1，就会造成梯度消失问题。这里需要解释一下(9)(10)里的n_i、n_{i+1}(这里 n_i是第 i 层的维

度，n_{i+1}是第 i+1 层的维度)。在前向传播时，对于某层的一个神经元，它的输入维度是n_i（前层网络的神经元的个数)，而在反向传播时，该层的这个神经元的输入是下一层神经元的个数n_{i+1}，(这里考虑全连接层)所以便有:

对于前向：$\mathrm{var}[W^i]=\frac{1}{n_i}$

对于反向：$\mathrm{var}[W^i]=\frac{1}{n_{i+1}}$

为了不引起误解这里将n_{i+1}用m_{i+1}代替，作为一个折中的选择，大神 Yoshua bengio 找到一个折中的选择:

$$\mathrm{var}[W^i]=\frac{2}{n_i+m_{i+1}}(21)$$

对于公式(14、15)，对于均匀分布来说U~$[-t,t]$，其方差是$\frac{t^2}{3}$，结合公式(21)，便有:

$$\mathrm{W}\sim\mathrm{U}\left[-\frac{\sqrt{6}}{\sqrt{n_j+m_{j+1}}},\frac{\sqrt{6}}{\sqrt{n_j+m_{j+1}}}\right]$$

到这里，便得出了初始化权重的分布，“环环相扣，步步惊心”。赞一下大神的思想巧夺天工，很美妙。可能还没有看出这样的权重初始化方法有什么益处，相信在看到“后面的内容”，你会赞叹“这世界好美丽！”

在看完上面的推导之后，我们会发现以前学的数学知识是多么的精妙，在这里向我们展示了数学的魅力，同时也展现了神经网络（深度学习），它不是一个黑盒子，而是一辈又一辈“潜心钻研”的伟大的人精心设计的美好事物。同时也告诫我们，想要得到好的模型，需要的是扎实的数学基础、缜密的逻辑推理能力、持之以恒的激情，真正的投入才会有所收获。。。作为解读文章，我在这里写这些可能有些不合时宜，只是对作者的敬仰之情让我有些情不自禁的想要赞美 **bengio** 等大神一番。以后的文章里，我会继续解读“后面的内容”。