模式识别第四次实验报告

162050127 颜劭铭

2022 年 12 月 24 日

1 什么是梯度消失和梯度爆炸

梯度消失和梯度爆炸问题，总的来说可以被称为梯度不稳定问题，通常出现在深度神经网络中。

- 梯度消失：梯度呈指数级减少，最终趋近于 0，网络权重无法更新或者更新的幅度很小，导致网络训练过程很慢，无法找到最优点。
- 梯度爆炸：梯度呈指数级增长，最终趋向于一个很大的数，网络权重更新的幅度很大，导致网络训练过程变得很不稳定，无法找到最优点。

如果简单地从二维的角度去定义梯度的话，如图1所示，最优点也就是点 1，起始点可能是在点 A，在梯度消失的情况下，可能经过很多个 Epoch 的情况下，网络只找到了点 B，因此找不到最优点；而在梯度爆炸的情况下，可能第一次更新网络就找到了点 F，第二次更新回到了点 A，也就是说网络的更新过程中一直越过了最优点，因此也找不到最优点。

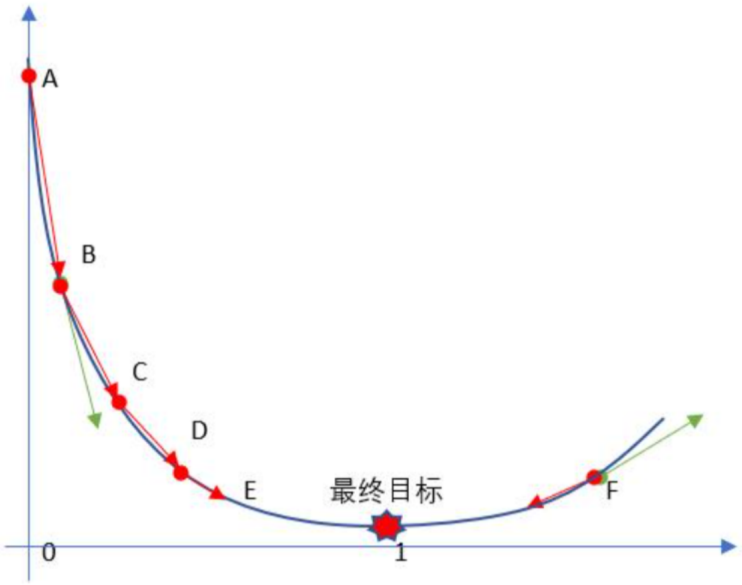


图 1: 梯度图

2 为什么会有梯度消失和梯度爆炸

提到梯度，最熟悉的应该是求解神经网络的模型参数过程中常采用的方法：梯度下降法。

梯度下降法的根源就是深度神经网络与反向传播。深度神经网络实际上就是由许多神经元组成了一个个非线性层的堆叠，以及最后的激活函数和全连接层。每一个非线性层都可以被看做一个非线性函数 $f(x)$，因此整个神经网络可以用如下形式函数表示：

$$F(x) = f_n(\dots f_3(f_2(f_1(x) \times \theta_1 + b_1) \times \theta_2 + b_2) \times \theta_3 + b_3 \dots)$$

而利用深度神经网络解决问题实际上就是希望这个函数可以较为完美地匹配输入到输出的映射。假设我们的任务的 ground truth 是 g(x)，而网络 F(x) 得到的解为 f(x), 那么我们就会设置一个简单的均方误差损失函数：

$$loss = \|g(x) - f(x)\|_2^2$$

此时的损失函数在二维的情况下就如图1所示，此时我们就可以使用梯度下降法寻找该损失函数的最优点，当找到最右点的时候，此时神经网络的参数就是最优参数。

接下来介绍反向传播，以一个简单的三层感知机为例子，如图2所示。

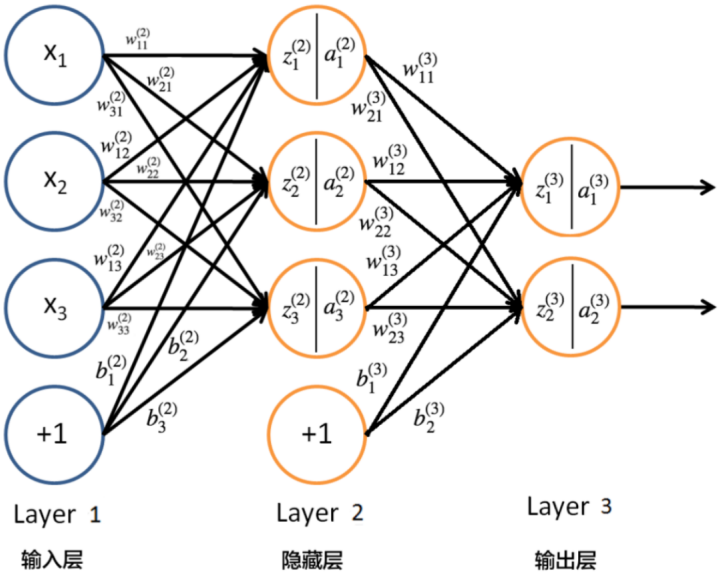


图 2: 三层感知机图

输入数据 $x = (x_1, x_2, x_3)^T$ 为三维的，以对输出层的权重参数求偏导为例，如下所示：（该部分只是简单的说明，解释得不是很全面，主要为了说明权重更新过程和梯度下降法的关系，来推导出为什么会导致梯度爆炸和梯度消失）

$$\begin{aligned}
\frac{\partial E}{\partial w_{11}^{(3)}} &= \frac{1}{2} \cdot 2 \left(y_1 - a_1^{(3)}\right) \left(-\frac{\partial a_1^{(3)}}{\partial w_{11}^{(3)}}\right) \\
&= -\left(y_1 - a_1^{(3)}\right) f'\left(z_1^{(3)}\right) \frac{\partial z_1^{(3)}}{\partial w_{11}^{(3)}} \\
&= -\left(y_1 - a_1^{(3)}\right) f'\left(z_1^{(3)}\right) a_1^{(2)}
\end{aligned}$$

如果我们把 $\frac{\partial E}{\partial z_i^{(l)}}$ 记为 $\delta_i^{(l)}$, 即做下面的定义:

$$\delta_i^{(l)} \equiv \frac{\partial E}{\partial z_i^{(l)}}$$

则 $\frac{\partial E}{\partial w_{11}^{(3)}}$ 显然可以写为:

$$\begin{aligned}
\frac{\partial E}{\partial w_{11}^{(3)}} &= \frac{\partial E}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial w_{11}^{(3)}} \\
&= \delta_1^{(3)} a_1^{(2)}
\end{aligned}$$

其中: $\delta_1^{(3)} = \frac{\partial E}{\partial z_1^{(3)}} = \frac{\partial E}{\partial a_1^{(3)}} \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} = -\left(y_1 - a_1^{(3)}\right) f'\left(z_1^{(3)}\right)$

同样的对于其它层的权重参数更新的过程也可以如对于输出层的权重参数求偏导所示。总的来说，反向传播可以总结为如下四个公式：

$$
\begin{aligned}
\delta_i^{(L)} &= -\left(y_i - a_i^{(L)}\right) f'\left(z_i^{(L)}\right) \\
\delta_i^{(l)} &= \left(\sum_{j=1}^{n_{l+1}} \delta_j^{(l+1)} w_{ji}^{(l+1)}\right) f'\left(z_i^{(l)}\right) \\
\frac{\partial E}{\partial w_{ij}^{(l)}} &= \delta_i^{(l)} a_j^{(l-1)} \\
\frac{\partial E}{\partial b_i^{(l)}} &= \delta_i^{(l)}
\end{aligned}
\tag{1}
$$

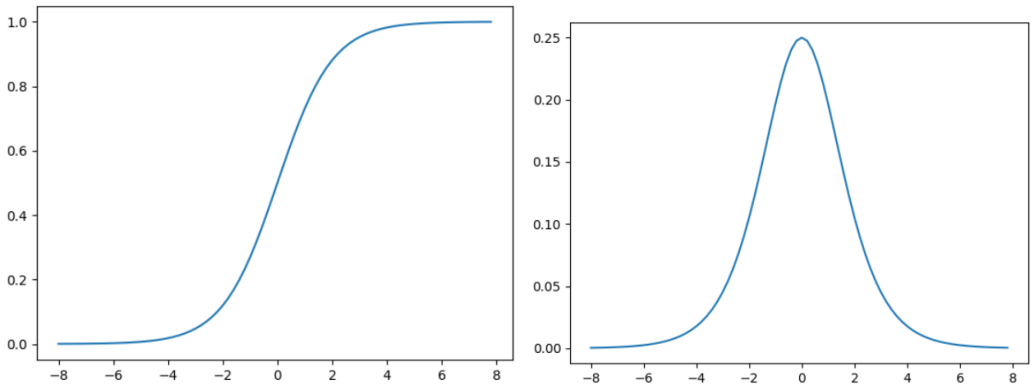
而更新权重的公式如下所示：

$$
\begin{aligned}
w &\leftarrow w + \Delta w \\
\Delta w &= -\alpha \frac{\partial E}{\partial w_{ij}^{(l)}}
\end{aligned}
$$

而深度神经网络更新权重时的反向传播简单来说也就是多个公式 $\frac{\partial E}{\partial w_{ij}^{(l)}}$ 的连乘，因此当 $\frac{\partial E}{\partial w_{ij}^{(l)}}$ 部分大于 1 时，随着层数的不断增多，梯度更新信息将会以指数形式增长，发生梯度爆炸，而当 $\frac{\partial E}{\partial w_{ij}^{(l)}}$ 部分小于 1 时，随着层数的增多，梯度更新信息将会以指数形式减小，发生梯度消失。

还可以从激活函数的角度进行理解，例如使用 sigmoid 函数作为激活函数，函数图像和导数图像如图3.(a) 和图3.(b) 所示：

$$
\begin{aligned}
S(x) &= \frac{1}{1 + e^{-x}} \\
S'(x) &= \frac{e^{-x}}{\left(1 + e^{-x}\right)^2} = S(x)(1 - S(x))
\end{aligned}
$$



(a) Sigmoid　　(b) Sigmoid 导数

图 3: Sigmoid 函数及导数图像

可以看出，Sigmoid 作为激活函数的时候，他的导数值，也就是梯度值是不可能超过 0.25 的，而在神经网络层数加深的时候，对于某个权重进行更新需要进行多层链式求导，就很容易出现梯度消失的问题。

3 怎么缓解梯度消失和梯度爆炸

3.1 梯度剪切、权重正则化

- 梯度剪切：主要针对梯度爆炸，设置一个梯度剪切阈值，在更新梯度的时候，若梯度超过阈值，就将其缓解在阈值范围内，缓解梯度爆炸。
- 权重正则化：常见的有 L1 正则化，L2 正则化。

正则化损失函数形式如下所示：

$$Loss = (y - W^T x)^2 + \alpha \|W\|^2$$

其中 α 是正则化项系数，通过正则化项可以在一定程度上缓解梯度爆炸的发生。

3.2 使用其他激活函数

例如 RELU 函数，表达式如下所示：

$$Relu(x) = \max(0, x)$$

Relu 函数的导数在 x>0 的情况下是恒等于 1 的，因此在深度神经网络权重更新的过程中使用 Relu 激活函数就不会导致梯度消失和梯度爆炸的问题，但是由于负数部分导数恒为 0，也会导致部分神经元无法激活。

3.3 BatchNorm 批归一化

通过将每层网络的神经元输入分布归一化到正态分布，使得激活函数的输入值落在对于输入变换较为敏感的区域，输入的略微变化也会导致损失函数也会有较大的变化，也就是梯度较大，缓解了梯度消失的问题，同时也可以加快学习速度。

3.4 残差结构

残差结构如图4所示。

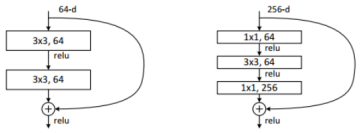


图 4: 残差结构

残差结构中的跨层连接结构，即在前向传播的过程中，输出端的值加上输入 X 再传到下一层网络。由于 X 是直接传递的，因此他没有权重参数，也就是说跨层连接的分支 X 的梯度始终为 1，因此即使主干网络 F(x) 前向传播的时候梯度值很小，加上分支的梯度都可以很好地缓解梯度消失的问题。