链滴

梯度消失和梯度爆炸

作者：lai-bluejay
原文链接：https://ld246.com/article/1555827028483
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

1 是什么

首先，BP神经网络基于梯度下降策略，以目标的负梯度方向进行权重更新，$\omega \leftarrow \omega + \Delta\omega$, 给定学习率$\alpha$, $\Delta \omega = -\alpha \times \frac {\partial{Loss}}{\partial \omega}$。假设每层全连接网络激活函数为$f(\dot)$, 则i+1层的输入$f_{i+1}= f(f_i * w_{i+1}+ b_{i 1})$, 则$\frac{\partial{f_{i+1}}}{\partial{w_{i+1}}}= f_i$

根据链式求导法则，当需要更新第二层隐层梯度信息时: $\Delta w_{1}=\frac{\partial \text {Loss}}{\partial w_{2}}=\frac{\partial {Los}s}{\partial f_{4}} \frac{\partial f_{4}}{\partial f_{3}} \frac{\partial f_{3}{\partial f_{2}} \frac{\partial f_{2}}{\partial w_{2}}$，又 $\frac{\partial f_{2}}{\partial w_{2}}=f_1$。现每一层的更新都需要求激活函数在上层输出值下的导数值。如果激活函数>1或<1，在层数加深时导数就会呈指数型变化，就可能产生梯度消失或梯度爆炸。

接下来，我们来看一段代码。

```
class TorchNet():
    dtype = torch.float
    device = torch.device('cpu')

    # 定义batch size, 输入特征数， 隐层特征，输出数
    N, D_in, H, D_out = 64, 1000, 100, 10

    # 随机初始化
    x = torch.randn(N, D_in, device=device, dtype=dtype)
    y = torch.randn(N, D_out, device=device, dtype=dtype)

    w1 = torch.randn(D_in, H, device=device, dtype=dtype)
    w2 = torch.randn(H, D_out, device=device, dtype=dtype)
    # 定义学习率，可以修改一下，看看结果
    learning_rate = 1e-6
    for t in range(500):
        # forward
        h = x.mm(w1)
        h_relu = h.clamp(min=0)
        y_pred = h_relu.mm(w2)

        # compute loss
        loss = (y_pred - y).pow(2).sum().item()
        print(t, loss)

        # backprop to compute gradients
        grad_y_pred = 2 * (y_pred - y)
        grad_w2 = h_relu.t().mm(grad_y_pred)
        grad_h_relu = grad_y_pred.mm(w2.t())
        grad_h = grad_h_relu.clone()
        grad_h[h<0] = 0
        grad_w1 = x.t().mm(grad_h)

        # update gradientts
        w1 -= learning_rate * grad_w1
        w2 -= learning_rate * grad_w2
```

（可以通过链式求导法则理解）我们看到，权值w的更新和梯度息息相关，每次反向传播 , 如果激活

数的导数趋近于0，那么权值在多重传播之后可能不再更新，则是梯度消失；如果梯度取值大于1，经多层传播之后，权值的调整就会变得很大，导致网络不稳定。

这些问题都是“反向传播训练法则”所具有的先天问题。

2. 如何判断

1. 看loss的变化/权值/参数的变化是否稳定，或者无法更新
2. loss是否变成了NaN
3. 权重是否变成NaN

3. 如何解决出现的问题；如何避免

我们看到，梯度爆炸或者消失的根本原因来自于激活函数，同时，激活函数的导数值又影响实际的梯更新，因此我们考虑从激活函数和函数的导数值来解决激活函数的问题。

3.1 重新设计网络

1. 层数更少的简单网络能够降低梯度消失和梯度爆炸的影响
2. 更小的训练批次也能在实验中起效果
3. 截断传播的层数
4. 同样，长短期记忆网络（LSTM）和相关的门单元也能减少梯度爆炸。
5. 对网络权重使用正则，防止过拟合。 Loss =(y-W^Tx)^{2}+\alpha\|W\|^2

3.2 修改激活函数

1. 修改为ReLU，leaky ReLU，PReLU，ELU，SELU，都可以。也可以使用maxout

3.3 使用梯度截断（Gradient clipping）

对大型深层网络，还是要检查和限制梯度大小，进行梯度截断。WGAN中，限制梯度更新是为了保证lpchitz条件。

3.4 Batch Normalization

Batchnorm 具有加速网络收敛速度，提升训练稳定性的效果，通过规范化操作将输出信号x规范化到值为0、方差为1，保证网络的稳定性。batchnorm在反向传播的过程中，由于对输入信号做了scale和hift，通过观察激活函数及其函数的图像(请点击链接中的超链)，这样可以是激活函数的值落在对非线函数比较敏感的区域。这样也会使损失函数产生较大的变化，让梯度整体变大，避免梯度消失；同时意味着收敛速度更快，学习速度更快。

3.5 ResNet，残差网络结构

终结者，ResNet。