|  |  |
| --- | --- |
| 成绩 |  |

装

订

线

网络空间安全与计算机学院实验报告

实验课程名称**：神经网络与深度学习实验**

实验项目名称： 前馈神经网络（4）-优化问题

学生姓名： 刘新媛

专 业： 人工智能

学 号： 20221205037

实验地点： C1-320

实验时间： 2024.10.30第7-8节

指导教师： 魏勇刚

预习报告部分

1. **实验目的：**发现神经网络模型的优化问题，并思考如何改进。
2. **实验原理：**

1、参数初始化：合适的参数初始化可以避免对称权重现象，提高模型学习效率。

2、梯度消失问题：深层网络中梯度可能因连续乘积而衰减，导致权重更新困难。

3、死亡ReLU: ReLU在负输入时梯度为零，可能导致某些神经元永久失活。

1. **实验内容：**

**[作业内容链接](https://blog.csdn.net/qq_38975453/article/details/126772521)**

**（1）基本概念**

**1、参数初始化**

**2、梯度消失问题**

**3、死亡ReLU问题**

**（2）编程实现**

根据实验教材代码，使用Pytorch完成相应程序编写。

1. **实验所用设备：pycharm 、pytorch框架**

实验报告部分

1. **实验步骤：**

**（1）基本概念**

**1、参数初始化**

实现一个神经网络前，需要先初始化模型参数。如果对每一层的权重和偏置都用0初始化，那么通过第一遍前向计算，所有隐藏层神经元的激活值都相同；在反向传播时，所有权重的更新也都相同，这样会导致模型无法学习，出现**对称权重现象**。

解决方法：为了避免对称权重现象，可以使用**高斯分布或均匀分布**初始化神经网络的参数。

**2 、梯度消失问题**

在神经网络的构建过程中，随着网络层数的增加，理论上网络的拟合能力也应该是越来越好的。但是随着网络变深，参数学习更加困难，容易出现梯度消失问题。

由于Sigmoid型函数的饱和性，饱和区的导数更接近于0，误差经过每一层传递都会不断衰减。当网络层数很深时，梯度就会不停衰减，甚至消失，使得整个网络很难训练，这就是所谓的**梯度消失问题**。

解决方法：

|  |
| --- |
| **调整学习率：**使用公式2-1来计算学习率  （2-1）  其中，为初始学习率，是学习率衰减参数，e是迭代次数 |
| **使用批量规范化（BN）：**[batchnorm原理及代码详解](https://blog.csdn.net/qq_25737169/article/details/79048516) |
| **使用RMSprop：**使用移动平均的梯度的平方来调整学习率。 |
| **使用Relu等激活函数：** f(x) =max(x,0) |

一种简单有效的方式就是使用导数比较大的激活函数，如：ReLU。

**3 、死亡ReLU问题**

ReLU激活函数可以一定程度上改善梯度消失问题，但是在某些情况下容易出现**死亡ReLU问题**，使得网络难以训练。

这是由于当x<0时，ReLU函数的输出恒为0。在训练过程中，如果参数在一次不恰当的更新后，某个ReLU神经元在所有训练数据上都不能被激活（即输出为0），那么这个神经元自身参数的梯度永远都会是0，在以后的训练过程中永远都不能被激活。

一种简单有效的优化方式就是将激活函数更换为Leaky ReLU、ELU等ReLU的变种。

**（2）编程实现**

**1、参数初始化**

模型包含两个全连接层（fc1 和 fc2），将每个线性层的权重和偏置初始化为零，使用 sigmoid 激活函数。

参数全部初始化为0，可以观察到对称权重问题。所以在下面的问题探究中权重不全部初始化为0，采用高斯分布、均匀分布或正态分布的初始化方法之一。

**2、梯度消失问题**

模型包含五个全连接层（fc1 至 fc5），支持三种激活函数：sigmoid、ReLU 和 Leaky ReLU；权重初始化为正态分布，偏置值为1.0。

先使用sigmod函数作激活函数可以看到梯度消失现象，然后再改用relu函数，可以观察到梯度销售现象得到了很大改善。

**3、死亡ReLU问题**

模型包含五个全连接层（fc1 至 fc5）；支持三种激活函数：sigmoid、ReLU 和 Leaky ReLU；权重初始化为正态分布，偏置值为 **-8.0**。

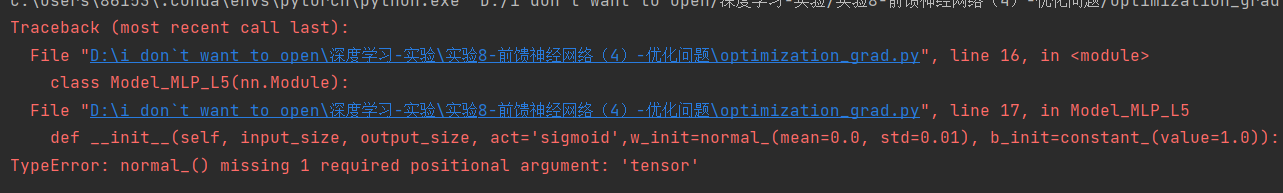
设置偏置为-8.0（一个较大的负数），使用RELU函数激活，可以看到死亡Relu现象，然后换Leak ReLU函数进行优化。

**二、 实验数据及结果分析：**

[点击查看代码以及运行结果分析](https://blog.csdn.net/qq_73704268/article/details/143392523?sharetype=blogdetail&sharerId=143392523&sharerefer=PC&sharesource=qq_73704268&spm=1011.2480.3001.8118)

1. **实验结论：**
2. 参数初始化对神经网络的训练至关重要。使用高斯分布、均匀分布等方法初始化参数可以避免对称权重现象，提高模型的学习效率。
3. 梯度消失是深度神经网络训练中的一个常见问题，特别是在使用Sigmoid激活函数时。通过改用ReLU激活函数，可以显著改善梯度消失问题。
4. 死亡ReLU问题可以通过使用Leaky ReLU、ELU等ReLU的变种来解决，这些激活函数允许负输入有一个非零的梯度，从而避免了神经元永久失活的问题。
5. **总结及心得体会：**

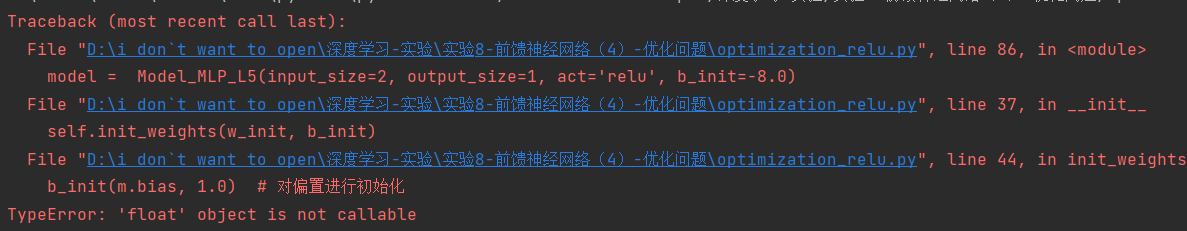
1、这次实验认识到一个新的模块即torch.nn.init模块，通过这个函数可以很方便的对参数进行初始化，但是使用时也遇到了几个问题：



我一开始直接按照paddle中的写的,直接在\_init\_方法中调用函数

但是换到pytorch框架结果却报错。是因为torch.nn.init.normal\_() 和 torch.nn.init.constant\_() 函数的使用方式需要在具体的张量上进行初始化，而不是直接在 \_\_init\_\_ 方法的参数中调用。

torch.nn.init.normal\_() 和 torch.nn.init.constant\_() 等初始化函数的设计是为了对具体的张量进行操作。需要一个目标张量作为输入，并在这个张量上进行初始化。**函数参数不接受张量，所以**不能直接在参数中使用 normal\_() 或 constant\_()。



这里是在探究死亡RElu问题初始化模型时，我将 b\_init 的值直接设为 -8.0，但是这时我直接用的梯度消失问题中的模型结构（在\_\_init\_\_方法中设置了参数b\_init=nn.init.constant\_），还是一个函数参数，我修改为b\_init=-8.0 就不报错了：

def \_\_init\_\_(self, input\_size, output\_size, act='sigmoid', w\_init=nn.init.normal\_, b\_init=-8.0)

2、实验中在可视化方面上花了较长时间，因为可视化每一层梯度范数时，数量级比较小，达到了1E-10以下，但是我在设置坐标轴范围时一直从0开始。即代码中只写plt.ylim(1e-8, 1) 不能改变y轴的刻度，但我改为用 plt.yticks([1, 1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7, 1e-8, 1e-9, 1e-12, 1e-17]就可以了。【又认识到一个好玩的函数 **plt.yticks**，可以自定义刻度值】

3、这次实验切身实践了理论课老师讲过的几个问题与对应的优化方法，之前只是听说，现在亲身经历后对其理解的更加透彻了，实验比较简单，模型构建和模型训练和之前的步骤是一样的，所以代码很快就写好了，总结一下：

权重初始化不能全部初始化为0，应该使用高斯分布、均匀分布等方法进行初始化；当使用sigmod激活函数或其他不恰当的激活函数时，会发生梯度爆炸或消失问题，这时应该改变激活函数（常用RelU）；当偏置为一个大的负数时，会出现死亡ReLu问题，这时可以更改激活函数为leaky\_relu函数。

|  |
| --- |
| **参考链接：** |
| [激活函数（四）：Leaky ReLU函数](https://blog.csdn.net/hy592070616/article/details/120617996?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522B1A3E679-27A8-4A26-AAEA-141744AD3680%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=B1A3E679-27A8-4A26-AAEA-14174) |
| [pytorch 笔记：torch.nn.init](https://blog.csdn.net/qq_40206371/article/details/128741241?ops_request_misc=%257B%2522request%255Fid%2522%253A%25221AA43CF1-24A9-4218-8E25-36F40F71B287%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=1AA43CF1-24A9-4218-8E25-36F40) |
| [梯度爆炸实验](https://blog.csdn.net/m0_62581697/article/details/134972160?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522B2A53952-CF55-4DB2-8C85-ABAEB70D17C1%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=B2A53952-CF55-4DB2-) |
| [深度学习 --- 优化入门三（梯度消失和激活函数ReLU）](https://blog.csdn.net/weixin_42398658/article/details/84553411?ops_request_misc=%257B%2522request%255Fid%2522%253A%252256AD7175-453B-41E9-8B0A-0B6EB38F0EF0%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=56AD7175-453B-41E9-8B0A-0B) |
| [【AI知识点】梯度消失（Vanishing Gradient）和梯度爆炸（Exploding Gradient）](https://blog.csdn.net/weixin_43221845/article/details/142731204?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522545DC685-317C-45B7-A239-4297C973C26F%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=545DC685-317C-45B7-A239-4) |