1.1 关于反向传播中梯度计算的分析；

如前文所示，反向传播的梯度计算有如下形式：

$$\delta^{(n)} = \nabla E \odot \sigma'\left(x^{(n)}\right) \tag{1}$$

$$\delta^{(k)} = \delta^{(k+1)} \cdot (w_{k+1})^T \odot \sigma'\left(x^{(k)}\right) \tag{2}$$

$$\frac{\partial E}{\partial b^{(k)}} = \left(\delta^{(k)}\right)^T \tag{3}$$

$$\frac{\partial E}{\partial w^{(k)}} = \left(y^{(k-1)}\right)^T \cdot \delta^{(k)} \tag{4}$$

其中：∇E 是关于损失函数的偏导，$\sigma'\left(x^{(k)}\right)$ 是关于激活函数的偏导；

分析：

迭代过程将面临两个问题：

1. 当 w_{k+1} 的行和大于 1 时（视 δ 行元素相等的情况下），公式（2）将至少随 w_{k+1} 的维度倍增；而当 w_{k+1} 的行和小于 1 时，公式（2）将随着 k 的迭代逐渐趋向于 0；

2. 每次迭代计算中，梯度（3）（4）将随着样本数量 n 的增长而增长，当 n 过大时，梯度很容易溢出；

为了解决这两个问题，很容易想到的是：

1. 问题 1 的解决方案：
   - 权值参数归一化；
   - 直接对 delta 归一化；

2. 问题 2 的解决方案：
   - 样本按列(样本数量维度)归一化；
   - 设定一个较小的 batch，模型的每次迭代只考虑该 batch 大小的样本。

解决方案可能会产生其他新的问题：

1. 问题 1 优化方案所带来的问题：

1.1 若样本已按列归一化，每个样本的最大元素将不超过 1；同时样本的标签远大于 1（如常见的回归问题）；此情况下样本与权值的乘积较小；

- 若在初始化神经网络时对权值参数归一化，则将导致权值与最优权值偏差较大，增大训练次数；此外，权值很可能将随则迭代次数逐渐增大，这就导致参数归一化的做法是无意义的，甚至是影响模型性能的。
- 为了避免陷入局部最优的情况，模型初始状态时的参数应当尽可能随机；而在初始状态对参数进行归一化的行为可以违法了这一理念；

1.2 需要对可行性做进一步分析；

2. 问题 2 优化方案所带来的问题：

2.1 并非能在真正意义上解决梯度随样本数量增大而增大的问题。

2.2 该方案应当可以有效控制梯度随样本数量变化的问题；但应该注意样本选取的随机性，若要求每一个 epoch 计算考虑所有的样本，最后一个 batch 的大小很快与之前的 batch 不一致，这可能导致 loss function 大小变化、梯度值大小变化等情况。

总结：

1. 上述解决方案 1.1 可能在分类问题中有较好的效果，但其在回归等其他情形下可能造成负优化，具体情况待验证；1.2 具体效果待验证；
2. 方案 2.1 及 2.2 待验证；

1.1.1 模型中参数的统计信息；

该小节由前文所提到的前馈神经网络内容，构造了一个 4 层神经网络；具体网络参数如下：

Layer	Function	Num of Neurons	备注	Act Func	Learn Rate
0	Input	150×16	样本数/维度	None	None
1	Dense	64	None	Relu	1e-10
2	Dense	256	None	Relu	1e-10
3	Dense	32	None	Relu	1e-10
4	Output	1	None	None	1e-10

表 1：网络结构

数据来自 Iris Data Set- R.F，共计 150 条；该小节将统计 10 个 epoches 的迭代计算中 w, b, δ, dw, db 参数的最大值、均值、中位数值、绝对值最小值的 4 个统计信息；

模型运行结果：

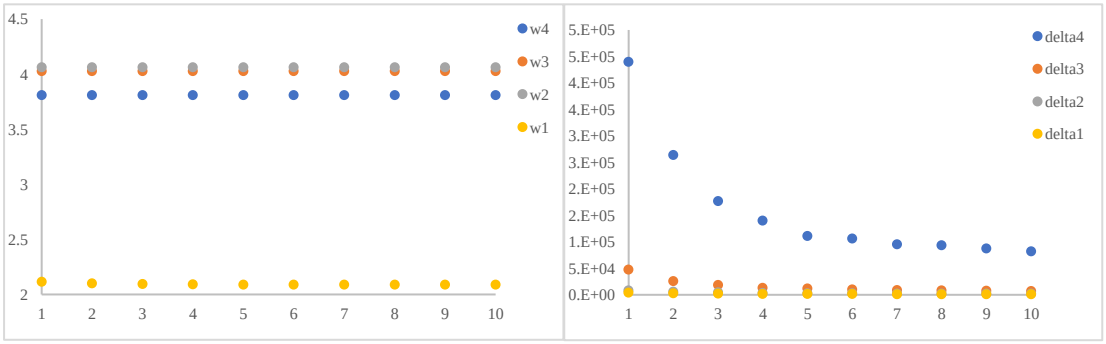

图 1, 2：10 次 epoch 迭代的 w 值（左）与 δ 值（右）（最大绝对值）

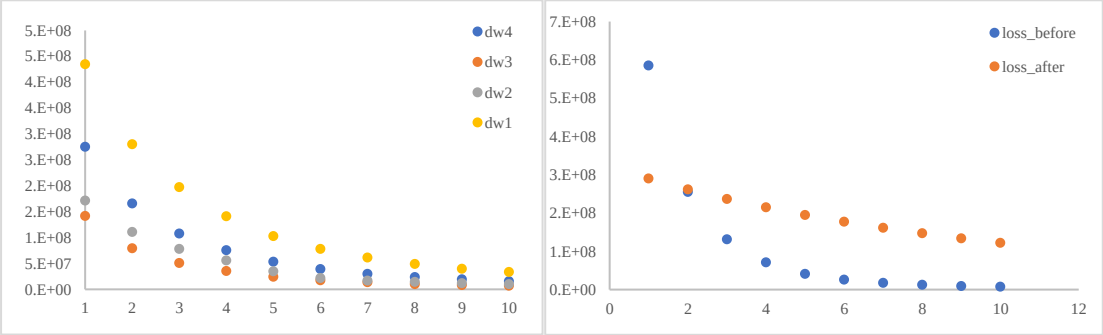

图 3, 4： w 的梯度值（左）（最大绝对值）与数据归一化前后损失函数值（右）

易见：

- w, δ, dw 三者分别在$10^0, 10^5, 10^8$ 量级；
- 参数值随着损失函数值下降时变化不大；
- 梯度值随着损失函数值下降时变化明显。

关于数据归一化的分析：

以下内容基于上述模型，并使用相同的初始化参数；但在训练前按照样本的每个维度的最大绝对值进行归一化：

max_abs_element = max(abs(x), axis=0);

x = x / max_element；

注意，这里不使用 max-min 标准化处理的原因是，该部分并非意图将数据映射到 [0,1] 区间，而只是作等比压缩处理以保留正负信息；模型经训练后损失函数值如图 4 所示；

ep	dw4	dw3	dw2	dw1
1	4.25	3.78	3.45	3.22
2	2.74	2.24	2.37	2.21
3	1.91	1.52	1.76	1.64
4	1.41	1.11	1.33	1.24
5	1.05	0.80	0.87	0.96
6	0.81	0.60	0.58	0.77
7	0.64	0.49	0.47	0.63
8	0.53	0.39	0.41	0.53
9	0.46	0.33	0.36	0.46
10	0.39	0.29	0.32	0.40

表 2：数据归一化前后 dw 比值

综合图 4 与表 1 信息易见，归一化能显著的降低模型初始时刻各梯度值，但同时由于学习率 η 为相同的常量，归一化后较小的梯度值也将导致基于梯度下降的优化缓慢。于是相同 epoch 之后，归一化后的梯度值反而比未归一化的模型的值更大。

但值得注意的是，数据归一化的模型训练时，梯度下降的更加均匀的。实际上，该方法并未在真正意义上解决前文中所提到的问题；

关于参数归一化的分析：

这一部分的内容同样基于前文中所提到的模型与初始化参数，同时不对数据作归一化处理，但对反向传播时的 δ 作上述最大绝对值归一化处理，同时鉴于归一化后梯度值变化过大，将学习率从 1e-10 调整到 1e-6，得到结果：

ep	dw4	dw3	dw2	dw1
1	441673.71	47652.19	8148.31	3881.54
2	295532.33	27925.69	5439.88	2612.83
3	187377.87	18519.08	4250.79	2063.29
4	162933.61	18173.64	4366.00	2141.75
5	242907.05	26442.26	5574.75	3215.70
6	344981.42	37960.99	6195.97	4346.88
7	466126.38	52797.15	8299.13	5819.99
8	499635.76	66435.12	10279.74	6911.91
9	512208.22	68980.64	10686.74	7101.94
10	491963.48	79631.10	11806.40	7827.26

表 3：δ 归一化前后 dw 比值

该操作对本节内容的影响是显著的；同时有下图 5 可知，该操作对梯度下降的均匀程度无影响；

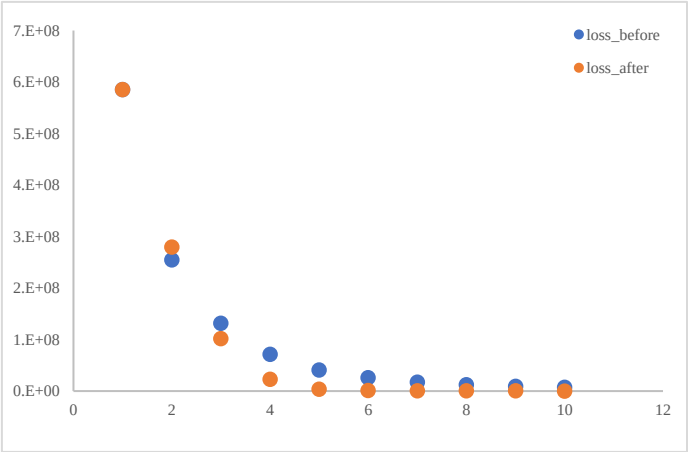

图 5：δ 归一化前后损失函数值（右）

关于 batch 训练的分析：

前文所提到的两种方案各有优点；但实际上如果样本数量过大（例如数亿条），公式（2）中输入层到第一隐藏层的计算结果以及公式（4）的计算结果可能会直接因为矩阵运算的结果超过模型所使用的编程语言所定义的浮点数类型最大值，即计算结果溢出；这种情况下，两种优化方法是无意义的。为了解决该问题，最简单的办法仍然是方案 2.2；关于该方案所产生的问题的初步简略分析在前文已经提到；

该方案在 class 中实现；

1.1 动态学习率

考虑前文中神经网络反向传播的过程，若第 k 层权值及偏置的梯度已确定:

$$\frac{\partial E}{\partial b^{(k)}} = \left(\delta^{(k)}\right)^T$$

$$\frac{\partial E}{\partial w^{(k)}} = \left(y^{(k-1)}\right)^T \cdot \delta^{(k)}$$

此时需要利用上述梯度更新该层权值与偏置；常规前馈神经网络中，通常会设定一个固定的学习率 η，以参数梯度与数 η 的乘积更新相应参数；

$$b^{(k)} = b^{(k)} - \eta \cdot \frac{\partial E}{\partial b^{(k)}}$$

$$w^{(k)} = w^{(k)} - \eta \cdot \frac{\partial E}{\partial w^{(k)}}$$

此处略过许多种学习率选定方法的介绍

描述：

每次更新时，考虑参数与梯度数量级大小，以两者数量级确定 η，以保证每次更新时，$\eta \cdot \frac{\partial E}{\partial b^{(k)}}$都是$b^{(k)}$的固定数量级；

1. 保证 η 梯度比参数小一定比例，防止参数大小溢出；
2. 保证 η 梯度不会比参数小太多，防止更新太慢；

描述详细版：

定义以下数量级函数：

$$\phi(x) = 10^{int(log_{10}x)}$$

例如：

$$\phi(81) = 10^1$$

$$\phi(101) = 10^2$$

定义学习率比率： ε；于是：

$$cb_j^{(k)} = \frac{\phi\left(b_i^{(k)}\right)}{\phi\left(\frac{\partial E}{\partial b_i^{(k)}}\right)}$$

$$cw_{i,j}^{(k)} = \frac{\phi\left(\max\left(w_{i,j}^{(k)}\right)\right)}{\phi\left(\max\left(\frac{\partial E}{\partial w_{i,j}^{(k)}}\right)\right)}$$

进一步令：

$$\eta b_j^{(k)} = \varepsilon \cdot cb_j^{(k)}$$

$$\eta w_{i,j}^{(k)} = \varepsilon \cdot cw_{i,j}^{(k)}$$

更新方式：

$$b_j^{(k)} = b_j^{(k)} - \eta b_j^{(k)} \cdot \frac{\partial E}{\partial b_j^{(k)}}$$

$$w_{i,j}^{(k)} = w_{i,j}^{(k)} - \eta \cdot \frac{\partial E}{\partial w_{i,j}^{(k)}}$$

此外，为了使得学习率能自适应地在训练前期较大以减少计算量，在训练后期较小以防止参数在最优值附近震荡，定义常量 d_{rate}，计算每一次迭代后损失函数值较上次损失函数值的变化，在 loss 函数值不降反增时将学习率比率自除d_{rate}，即：

$$if\ \Delta(e) < 0:$$

$$\varepsilon \coloneqq \varepsilon \div d_{rate}$$