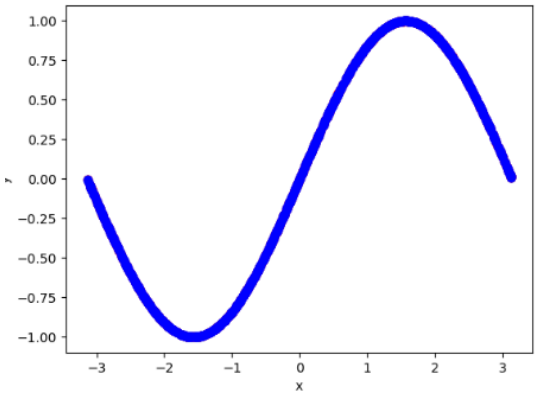
Lab1-Part1

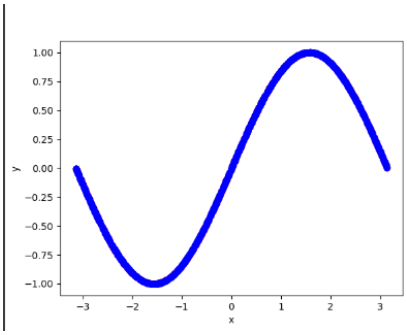
19302010074 张诗涵

(一) Sin 函数拟合

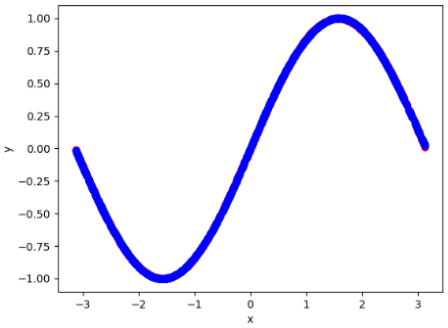
1. 代码基本结构：
   a) Layer：表示每一个线性计算层，通过创建时的参数 self_node_num 与 last_node_num 自动生成大小为 last_node_num * self_node_num 的权重矩阵与大小为 self_node_num 的 bias 矩阵。Conduct()函数计算前传结果，同时保存计算前的传入值作为回传时调参的依据 input；modify()根据传入的 delta_output(可以理解为之前每一层传回来的影响)与自己保存的 input 调整 bias 与 weight，同时将自己的导数乘在 delta_output 上继续回传。
   b) ActivationLayer：表示非线性的激活层，使用了 sigmoid 函数进行激活，conduct()函数对传入值进行 sigmoid 函数处理，同时计算出对应导数保存下来；在 modify()函数，由于激活层本身没有参数需要调整，仅需要将计算出的导数与传入的 delta_output 相乘继续回传。
   c) Model：初始化时根据要求的神经网络层数与每层的神经元数创建线性计算层与激活层，接着随机生成大小为 1000 或以上的数据集。train()函数遍历数据集，并根据每一个数据使用 BP 算法进行计算与调整，modify()函数即为调整函数，
2. 不同系数下的拟合结果（数据集大小为 1200）：
   1) 共 5 层，节点个数分别为[1,10,15,20,1]时训练 2000 次的拟合结果



   2) 共 4 层，节点个数分别为[1,15,20,1]时训练 2000 次的拟合结果：



   3) 共 3 层，每层神经元个数为[1,50,1]时训练 2000 次的拟合结果



(二) 手写汉字分类

1. 代码基本结构:

   大体上结构与 sin 函数拟合所用的框架差不多，复用了其 Layer(线性层)的设计，设计了 TanhActivationLayer 、 ReLuActivationLayer 、 LeakyReLuActivationLayer 继承 ActivationLayer，重写其 conduct 函数，即分别使用 tanh、ReLu、LeakyReLu 作为激活函数。

   由于计算结果最终需要 softmax 并且数据产生、划分方式较为不同，因此重写了 ModelOfClassification 作为分类的模型，由于有了多种继承关系的非线性激活层类，因此可以在初始化时可以更自由地进行激活层组合，并通过比较得出其优劣。

2. 使用不同网络结构的效果:

   1) 激活函数仅 sigmoid 一种：尝试了多种层数、神经元、learning rate 的组合，结果差异不明显，只要 learning rate 较小(0.01 以下，或通过 learning rate decay 在训练后期达到 0.01 以下)且训练 epoch 足够多的情况下，都能达到相似的正确率，在验证集上的正确率大致会逼近 75%同时无法进一步提高。

   2) 激活函数仅 ReLu 一种：一方面由于 ReLu 在大于 0 时直接返回本身而小于 0 时为 0, 容易出现 overflow 与神经元不激活的情况, 特别是后者在多次迭代后明显发生，因此使用纯 ReLu 的效果并不好。

   3) 激活函数间隔使用 sigmoid 与 leakyReLu： 此种情况下达到了最优的情况，正确率可以达到 90%以上（具体需要看初始值情况），同时收敛效果很好，在 100 个 epoch 后便不会出现剧烈的震荡，因此是最终的选择方案。

   4) 激活函数间隔使用 tanh 与 leakyRelu：训练效果优于仅使用 sigmoid 与 leakyRelu，但是与 sigmoid+leakyRelu 相比有明显更加严重的震荡，且最终收敛效果也不如后者，只能达到 87%左右。

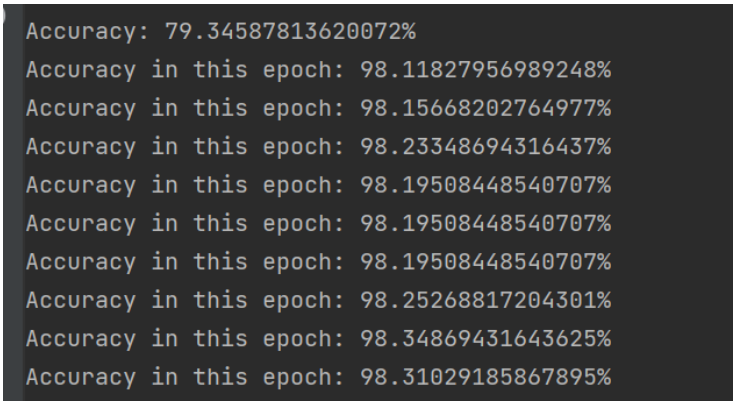
   5) 激活函数仅 LeakyRelu 一种：在仅三层的情况下{784，128，12}表现良好，起初的收敛速度很快, 50 次后即可大概达到 60-65%的正确率, 但之后训练速度明显变慢，可能是因为 learning rate decay 带来的速度降低，但整体仍保持着优化的趋势，跑完 200 次 epoch 后正确率大致稳定在 82%左右。 同时增加层数与神经元个数的影响并不大，无法带来正确率上的突破。

3. 一些思考&尝试使用的辅助手段:

   a) Learning rate decay：即每过一定量的迭代次数后将 learning rate * 某一个略小于 1 的数，理念是刚开始对权重与 bias 的调整可以较大，这样调整的速度更快，并且一开始大概率是离最优解较远的，不容易一下子陷入局部最优，而训练一段时间后应该将 learning rate 调小，较小幅度地进行精细调整，如果此时 learning rate 较大容易发生震荡。

   b) 过拟合：通过实验很明显可以发现存在过拟合的情况，对于使用 3 层+leakyReLu

的模型，经过 250 次迭代，其在训练集上已经可以达到近乎 98%的正确率，但是



在验证集上的正确率却只有 78-79%，并且无法上升。可行的解决方案包括使用 batch 和 weight decay 等，但此处多加考虑后并未使用。

c) 避免 softmax 上下溢出：由于 ReLu 会直接返回 x 本身、比起映射到-1~1 之间的 sigmoid 更容易在进行 exp 运算时发生 overflow，因此在 softmax 计算时首先获取输入的矩阵中的最大值，并将每一项减去这一最大值，再进行计算，这样的计算结果是相等的，但一定程度上避免了溢出的可能。

(三) 实验过程心得：

1) 一开始难以判断正确率没有上升是神经元层数与节点个数不够还是模型本身写错了（以至于没有向梯度下降的方向调整），后来得出的规律是，如果是正确的模型、前几个 epoch 应该都有明显大于 5-10%的正确率上升，如果一直保持在 8%那基本就是在瞎猜。

2) 训练集需要 shuffle：感觉这是还蛮多人的误区，如果不 shuffle 的话显然模型一定会逼近最后训练的那个类别，但此事是疑惑很久不知道为什么测试结果永远是最后一类后才得到了解答。