神经网络大作业一：用多层感知器拟合非线性函数

GitHub repo: https://github.com/dreamerlin/nn_project1

设计方案

该项目设计了一个可以自由设置隐藏层个数和每层神经元数量，并且可以选择激活函数为 ReLU 还是 leaky ReLU，是否使用 Batch Normalization 的网络。

最终选择使用 2 个隐藏层，每个隐藏层的神经元个数都为 10，且每层使用的激活函数都为 ReLU 函数，损失函数为均方误差函数的模型。

输入层的维度为 2 维，输出层的维度为 1，并使用反向传播算法对梯度进行计算传播。

具体实现

1. 网络结构

本程序可以通过传入 bash 参数，自由设计网络隐藏层数量和隐藏层神经元数量，也能选择激活函数为 ReLU 还是 leaky ReLU，是否使用 Batch Normlization 进行归一化，且将多层感知机封装成类。

上述参数可作为参数初始化类对象，可变化多种不同的网络结构，并从中选择最优的模型。最终展示效果的相应网络结构为 2 个隐藏层，每个隐藏层的神经元个数为 10，使用 ReLU 作为激活函数，且不使用 Batch Normalization 进行归一化

2. 初始化

每层神经元都有权重 W 和偏置 b 组成。W 初始化为 (mean, var)=(0, 0.05) 的正态分布矩阵，b 初始化为 0 的向量。从输入层到第一个隐藏层的权重 W 为 2*10 维矩阵，偏置 b 为 10 维向量。从第一个隐藏层到第二个隐藏层的权重 W 为 10*10 维矩阵，偏置 b 为 10 维向量。隐藏层的激活函数均为 ReLU 函数。从第二个隐藏层到输出层使用的权重 W 为 10*1 维，偏置为 1 维，无激活函数，损失函数为均方误差函数。

3. 训练

a) 设置网络的学习率 lr=0.001，设置网络完整迭代次数 epoch=1000。网络训练采取一批数据一起送入网络进行训练，计算这批数据与正确值之间的平均方差。设置批的大小 batch_size=64。

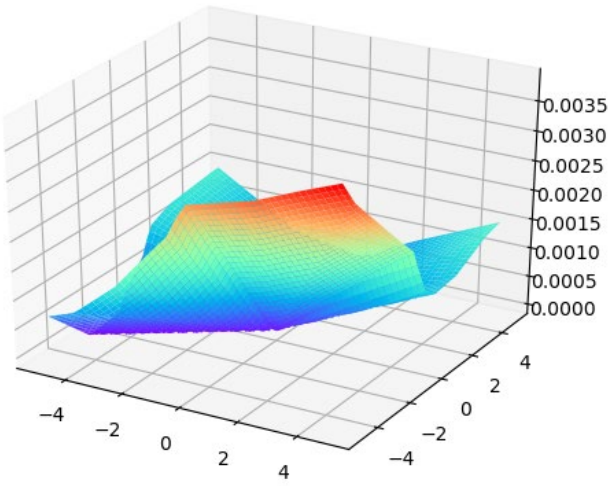
b) 随机生成 N*10000 个数据 X（维度为 10000*2），并且根据目标函数计算相应的 Y 值。 将这些数据分批送入参数初始化后的网络计算得到的值，一个完整的 epoch 会使得数据 X 刚好被完整地训练一次，即包含 N / batch_size 个迭代次数 iter。为了代码实现方便，代码的实现中每次迭代只随机选择 N 个训练数据中的 batch_size 个数据，并没有避免选择与当前 epoch 的之前的 iter 重复的数据。虽然每个 epoch 当中大概只有 2/ 3 的数据被采样到，但是经过多个 epoch 之后实际上每个训练样本被采样到的次数是相近的。

c) 计算一批数据的真实值与神经网络预测得到的值之间的平均方差，使用 BP 算法计算它对于每个网络参数的梯度，用梯度的反方向乘以学习率加到梯度上，从而更新梯度，完成一次迭代训练。

d) 精细调整参数使得后面的 iter 计算出来的损失值不再有明显变化，即说明网络收敛可停止迭代。

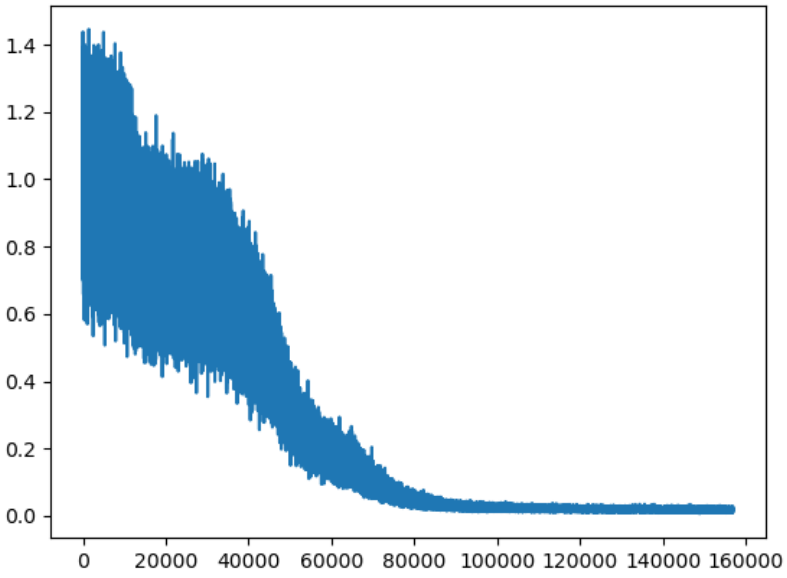
结果

1. 训练前初始化后的预测结果：Loss=0.9994035196911774 得到的特征图



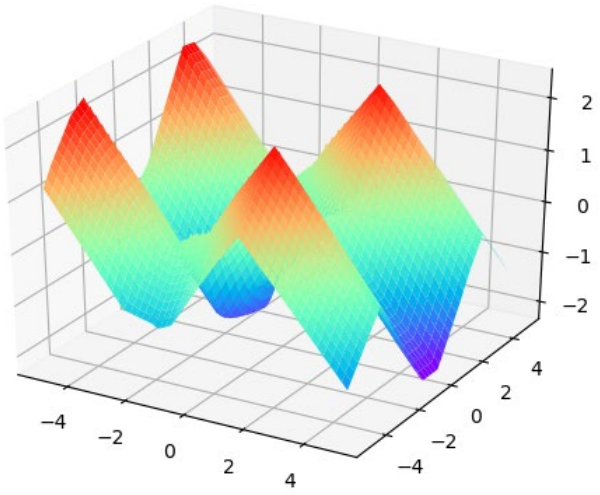
图一：训练前网络预测的目标函数图

2. 训练过程中的损失函数值随迭代次数变化图



图二：2. 训练过程中的损失函数值随迭代次数变化图

3. 训练完成后，神经网络对目标函数的预测图



图三：训练后神经网络对目标函数的预测图

可以看到训练之后的损失值已接近于 0，并且预测的目标函数也与真实的目标函数接近。

分析与讨论

1. 该多层感知机能够拟合 该非线性目标函数，误差反向传播使得网络的输出能够不断靠近真实值，仅有如此是不够的，因为如果激活函数是线性的则神经网络只能拟合与目标函数接近的线性函数，不能无限接近于目标函数，而激活函数非线性则使得神经网络拟合的函数尽可能地拟合目标函数成为可能。
2. 关于为什么选择了前面提到的最终设计方案。这样的超参数设计能够很好地拟合目标函数，在尝试不同的网络结构的过程中，最开始只使用了一层隐藏层，结果 Loss 降到 0.4 附近就不再降低，后来使用了两层隐藏层 Loss 就降到了 0.01 以下，这个结果已经足够好。 没有使用 leaky ReLU 而是使用 ReLU 激活函数，leaky ReLU 的效果并没有想象当中的那么好，反而还要尝试各种不同的 leaky ratio 值。也写了 Batch Norm 的代码，发现 Batch Norm 能够使得 Loss 随着迭代次数的增加下降得更快，但是相应地，每个 iter 的训练时间变得更长了，而且到训练的后期损失变得不稳定。 简单的两层隐层神经元,, ReLU 激活函数就已经使得网络的表现足够好。