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

网络结构中隐藏层数量和每个隐藏层神经元数量可以作为参数来构建网络，即隐藏层数量和每层隐藏层神经元个数是可以自由设置的，因此本项目可以实现多种不同的网络结构进行比较，从中选择最好的网络模型。除此之外，还能选择激活函数的类型，有三种类型的激活函数可供选择：relu，leaky_relu，sigmoid。Leaky_relu 的参数可以自由调整。Batch normalization https://arxiv.org/pdf/1502.03167.pdf 也能用于网络结构中增加网络的收敛速度。

实验结果发现，仅使用两层隐藏层，每层隐藏层的神经元个数在[10-20]个左右，训练轮数足够多时即能取得好的效果。

具体实现

1. **网络结构**

本程序可以通过传入 bash 参数来设置不同的网络结构。最终展示的网络结构包含 2 个隐藏层，第一个隐藏层的神经元数量为 20 个，第二个隐藏层的神经元数量为 15 个，使用 ReLU 作为第一层和第二层的激活函数，输出层不使用激活函数，不使用 Batch Normalization 进行归一化，损失函数为均方误差函数，训练算法为梯度反向传播算法。

2. **初始化**

每层神经元都由权重 w 和偏置 b 组成。权重 w 的维度为(上一层输出数据维度)*(本层输入数据维度)，用均值为 0，方差为 0.05 的正态分布初始化。偏置 b 的维度为本层输入数据维度，初始化为全 0。因此最终展示的网络结构中第一个隐藏层的权重 w 的维度为(2*20)，b 的维度为(10,)，第二个隐藏层的权重 w 的维度为(20*15)，b 的维度为 15，输出层的权重 w 的维度为(15*1)，b 的维度为 1。

3. **训练**

a) 网络训练采取随机梯度下降的方式，一批数据同时送入网络进行训练，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。设置网络的学习率 lr=0.001，设置网络完整迭代次数 epoch=600。一个 epoch 即训练集整个地被训练一次。

b) 生成 N=10000 个[-5, 5]之间均匀分布的数据 X（维度为 10000*2），并且根据目标函数$y = \sin(x_1) - \cos(x_2)$计算每组数据的正确函数值。

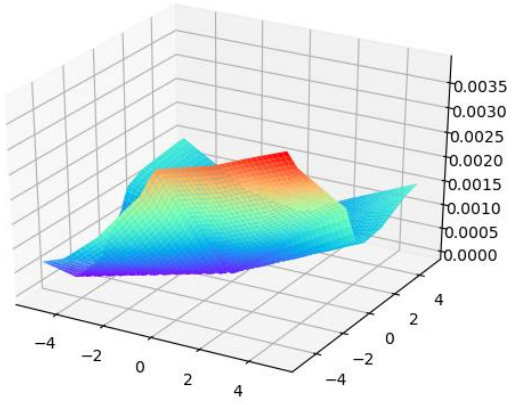
c) 将训练数据分批送入网络，用一批数据的真实值与经过神经网络之后的值之间的平均方差作为 loss 值，并使用梯度反向传播算法计算 loss 对于网络中的每一个参数的梯度值。

d) 对于每一个网络参数，将计算得到的梯度的反方向的值乘以学习率加到该网络参数上来更新梯度。网络参数的每次更新都完成了一次迭代。一个 epoch 是训练集中所有的训练样本都被训练一次，因此一个 epoch 里包含了多个迭代。

e) 重复 epoch 个上述 c)-d)的过程，完成网络训练。如果发现网络最后还没有收敛(即 loss 值还比较大)，则需要增加 epoch 数量，如果网络早已收敛，则可以适当减少 epoch 数量。

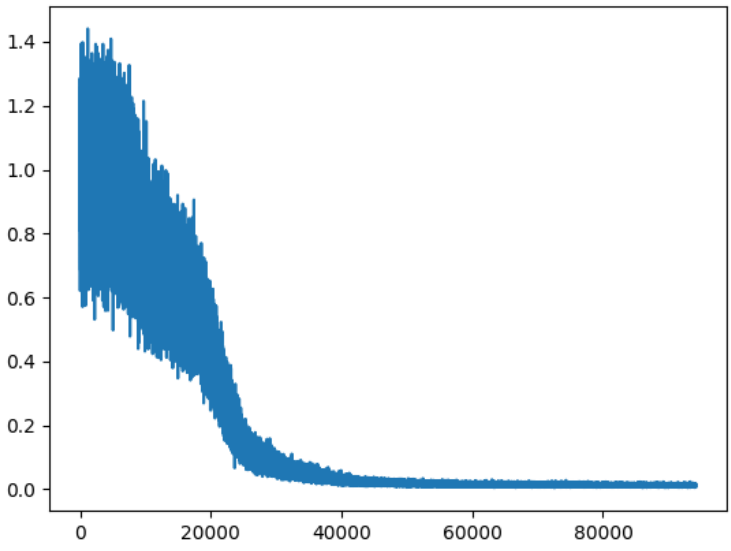
结果

1. 训练前：
   计算一批数据得到的 loss=0.9994
   网络拟合的函数图：



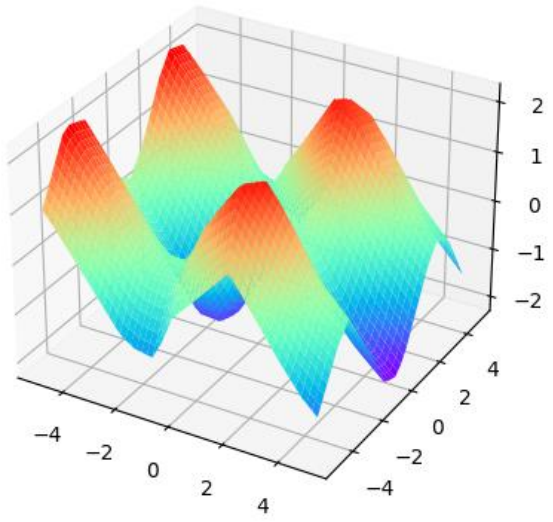
图一：训练前网络对目标函数拟合的图像

2. 训练过程中：
   损失函数随着迭代次数增加的变化图：



图二：2. 训练过程中 loss 值随迭代次数的变化

3. 训练后：
一批数据计算出的 loss=0.004659
网络对目标函数的拟合图：



图三：训练后网络对目标函数的拟合图像

训练之后损失值在 1e-3 的数量级，已与 0 很接近，并且拟合的函数图像与真实的函数图像接近。

分析与讨论

理论上来说，分段线性函数可以拟合任意连续函数，因为可以用无限小的线性函数片段去拟合连续函数的任意小的一段，因此神经网络具备拟合该目标函数的理论支撑，只要有足够深的层数和足够多的神经元个数。但是在实现中很难找到一个精确的带权重值的网络结构刚好拟合任意一个非线性函数。在本实验中，该非线性函数较为简单，同时也是一个连续函数，训练数据也足够多(10,000 个样本)，并且使用非线性激活函数能直接为网络引入非线性性，从而拟合非线性函数更快更容易。本报告中展示的效果图已经足够接近目标函数，尝试更多不同的网络结构已没有必要，因此本实验的最终网络结构如前所述。