## 函数拟合报告

#### 一、函数定义

在本实验中,我定义了一个sin(x)三角函数,该三角函数基于numpy库中的sin函数,定义该函数用来生成训练集和测试集,并将该函数进行简单的封装,该函数用来作为神经网络的学习以及拟合的目标。

**二、数据采集**

为了训练神经网络,需要采集一定数量的输入和输出数据对.在本次实验我使用了线性空间np.linspace函数生成了一组在[-π, π]区间内均匀分布的输入值x\_train,并通过自定义函数sin(x)函数计算对应的y\_train。这些数据对构成了训练数据集,用于训练神经网络。

**三、模型描述**

在本次实验中,我构建了一个基于ReLU激活函数的两层神经网络模型来拟合自定义的sin(x)函数,该模型包括一个输入层、一个隐藏层和一个输出层.输入层接收一维的输入数据,隐藏层包含一定数量的神经元,并使用ReLU函数作为激活函数,而输出层则输出拟合的结果。

在模型的构建过程中,我是用了PyTorch深度学习框架。首先定义了一个TwoLayerReLUNet类,该类继承了nn.Module基类,并实现了前向传播函数forward.在前向传播函数中,输入数据首先通过第一个全连接层,然后经过ReLU激活函数进行非线性处理,接着处理后的数据通过第二个全连接层进行进一步的线性变换,追后输出拟合结果。

为了训练模型,我选择了均方差(MSEloss)作为损失函数,并使用Adam优化器进行权值更新.通过迭代训练数据集,模型逐渐学习到输入和输出之间的映射关系,并优化其权值以最小化损失函数。

**四、拟合效果(可视化)**

经过一定轮数(我设计的是1000轮),可以评估神经网络对自定义的sin(x)函数的拟合效果,首先,我使用了训练好的模型对测试数据集进行预测,并且计算预测值和真实值之间的误差,通过比较预测值和真实值的曲线图,可以直观的看到模型的拟合效果,实验结果发现,模型拟合效果较为可观.

根据以上实验结果,可以发现,基于ReLU激活函数的两层神经网络能够较好地模拟自定义函数sin(x),在训练过程中,随着轮数地增加,损失函数地值逐渐减小,说明模型地拟合能力在不断提升.最终,预测值和真实值之间地误差会变得较小,且曲线会基本重合。

但是,可以注意到,在某些局部区域,预测值和真实值之间存在一定偏差.这可能是因为我的神经网络模型地复杂度不够,或者训练数据集不够充分。

**五、实验总结**

本次实验通过定义自定义的sin(x)函数,并基于ReLU激活函数的两层神经网络进行拟合,展示了神经网络在拟合函数方面的能力。为了解决"在某些局部区域,预测值和真实值之间存在一定偏差"的问题,将来可以尝试增加隐藏层神经元的数量、调整学习率或增加训练轮数等策略。