****

**2023——2024学年第二学期**

**使用基于ReLU的神经网络来拟合正弦函数**

姓 名：陆李天

学 号：2151188

所在院系：电子与信息工程学院

学科专业：数据科学与大数据技术

二〇二四年三月

**一、问题描述**

理论和实验证明，一个两层的ReLU网络可以模拟任何函数[1~5]。请自行定义一个函数, 并使用基于ReLU的神经网络来拟合此函数。

**二、函数定义**

这里函数我采用的是正弦函数。

# 定义函数  
def func(x): # 定义一个函数func，输入是x  
 return np.sin(x) # 函数的输出是x的正弦值

**三、数据采集**

我在正弦函数上随机采样生成了训练集，然后在一个区间内均匀采样生成了测试集。

# 生成训练集和测试集  
x\_train = np.random.uniform(-np.pi, np.pi, 1000) # 在[-π, π]区间内随机生成1000个训练数据  
y\_train = func(x\_train) # 使用定义的函数计算训练数据的标签  
x\_test = np.linspace(-np.pi, np.pi, 100) # 在[-π, π]区间内均匀生成100个测试数据  
y\_test = func(x\_test) # 使用定义的函数计算测试数据的标签

**四、模型描述**

在这个实验中，我使用了一个两层的全连接神经网络来拟合函数。这个神经网络模型是通过PyTorch库实现的，模型的结构如下：

输入层：模型的输入是一个实数，对应于要拟合的函数的输入。

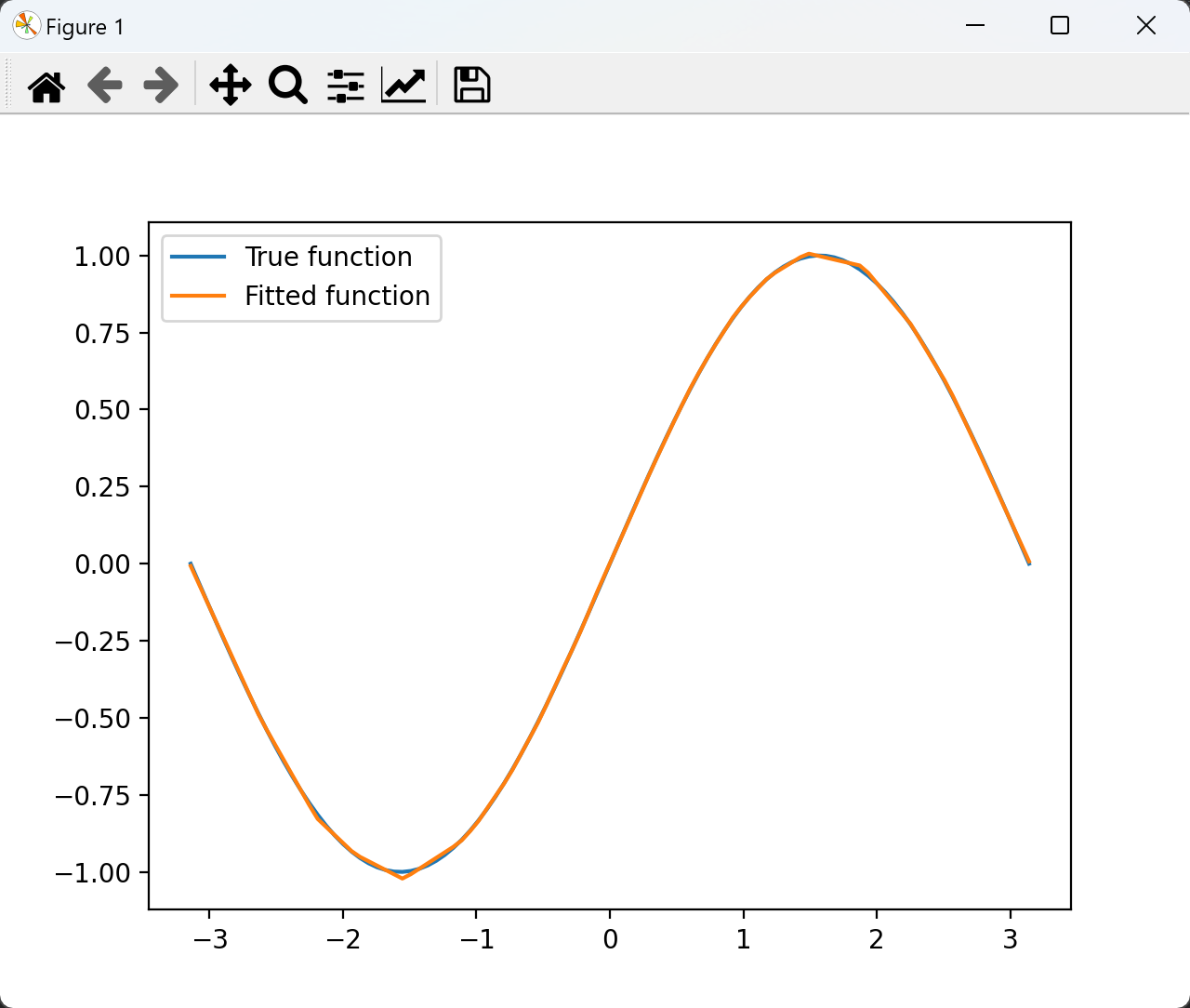
隐藏层：隐藏层包含50个神经元。每个神经元都连接到输入层，并通过一个线性变换（即权重矩阵和偏置向量）和ReLU激活函数来计算其输出。ReLU激活函数是一个非线性函数，它在0以下的值为0，而在0以上的值为输入值本身。这个非线性变换使得神经网络能够拟合复杂的函数。

输出层：输出层是一个线性层，它将隐藏层的输出转换为单个实数，这个实数就是神经网络的输出，对应于要拟合的函数的输出。

在训练过程中，我使用了均方误差损失函数和Adam优化器。均方误差损失函数衡量了神经网络输出和目标输出之间的差异，而Adam优化器则用于根据损失函数的梯度来更新神经网络的权重和偏置，从而改进模型的性能。

# 定义神经网络模型  
class Net(nn.Module): # 定义一个名为Net的类，继承自nn.Module  
 def \_\_init\_\_(self): # 类的初始化函数  
 super(Net, self).\_\_init\_\_() # 调用父类的初始化函数  
 self.fc1 = nn.Linear(1, 50) # 定义第一层全连接层，输入维度为1，输出维度为50  
 self.fc2 = nn.Linear(50, 1) # 定义第二层全连接层，输入维度为50，输出维度为1  
 self.relu = nn.ReLU() # 定义ReLU激活函数  
  
 def forward(self, x): # 定义前向传播函数  
 x = self.relu(self.fc1(x)) # 输入x经过第一层全连接层和ReLU激活函数  
 x = self.fc2(x) # 然后经过第二层全连接层  
 return x # 返回输出  
  
# 训练模型  
model = Net() # 实例化模型  
criterion = nn.MSELoss() # 定义损失函数为均方误差损失  
optimizer = torch.optim.Adam(model.parameters(), lr=0.01) # 定义优化器为Adam，学习率为0.01

**五、拟合效果**



可以看到拟合效果很好，只在极值点处有微小的差异。