同济大学

TONGJI UNIVERSITY

基于两层 ReLU 网络的曲线拟合实验报告

姓　　名	石浩成（2251189）
学院（系）	计算机科学与技术学院
专　　业	计算机科学与技术
指导教师	程大伟
日　　期	2025 年 3 月 15 日

一、实验概述

理论和实验证明，一个两层的 ReLU 网络可以模拟任何函数。本实验的目的是利用两层前馈神经网络，对非线性函数 $y=\sin(2\pi x)+0.3x$ 进行拟合。

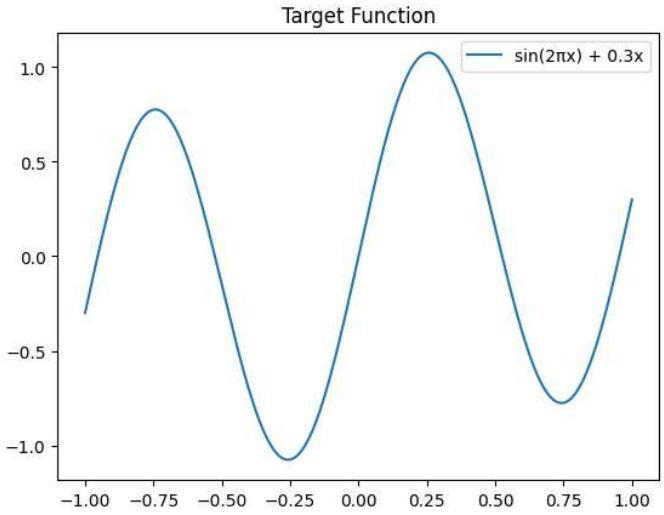
神经网络在函数拟合、模式识别和数据建模方面具有强大的能力。传统的线性回归方法对于复杂的非线性关系难以拟合，而深度学习方法能够通过多层神经元的组合，学习数据中的复杂模式，提高拟合精度。

本实验将构建一个两层神经网络，使用 ReLU 作为激活函数，并采用均方误差（MSE）作为损失函数进行优化。训练完成后，通过可视化手段评估模型的拟合效果，并分析其优缺点。

二、函数定义

目标函数定义为 $y=\sin(2\pi x)+0.3x$，并在区间 $[-1,1]$ 范围内采样数据。

```python
def targetFunc(x):
    """目标函数: y = sin(2πx) + 0.3x"""
    return np.sin(2 * np.pi * x) + 0.3 * x
```



三、数据采集

使用 NumPy 生成数据，并划分为训练集、验证集和测试集。

```python
x = np.linspace(-1, 1, 5000)
y = targetFunc(x)


X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=1)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.125, random_state=1)
```

四、模型描述

本模型是一个两层神经网络，主要用于处理一维输入数据，并进行回归任务。模型的核心架构包括一个隐藏层和一个输出层，并采用均方误差 (MSE) 作为损失函数。

4.1 模型架构

输入层：输入维度为 1，即每个样本包含一个特征。

隐藏层：包含 16 个神经元，采用 ReLU 作为激活函数。

输出层：输出维度为 1，不使用激活函数，直接输出预测值。

4.2 模型参数初始化

（1）权重初始化：

第一层权重矩阵 W1 形状为 (16,1)，采用 np.random.randn(hidden_dim, input_dim) * np.sqrt(1 / input_dim) 进行初始化。

第二层权重矩阵 W2 形状为 (1,16)，采用 np.random.randn(output_dim, hidden_dim) * np.sqrt(1 / hidden_dim) 进行初始化。

（2）偏置初始化：

b1 形状为 (16,1)，初始化为零。b2 形状为 (1,1)，初始化为零。

4.3 模型计算过程

（1）前向传播过程如下：

- 第一层线性变换： Z1 = W1 * X + b1
- 激活函数 ReLU： A1 = ReLU(Z1)
- 第二层线性变换： Z2 = W2 * A1 + b2
- 输出结果： y_pred = Z2

（2）损失计算方法：

损失函数采用均方误差 (MSE)，并存储梯度用于反向传播。

（3）反向传播过程如下：

- 计算输出层误差 delta2，并计算 dW2 和 db2。
- 计算隐藏层误差 delta1，仅对正梯度部分进行更新 (ReLU 的梯度特性)。

- 计算 dW1 和 db1。
- 依据学习率 lr 对参数进行更新。

4.4 模型训练过程

本模型采用小批量梯度下降 (Mini-batch Gradient Descent) 进行训练。在每个训练轮次中，将训练数据 X_train 和 y_train 划分成多个小批量，对每个批次执行前向传播计算预测值 y_pred，再计算损失并执行反向传播更新参数。训练过程中，每隔 verbose_ep 轮输出损失信息，方便观察训练进展。

训练超参数如下：

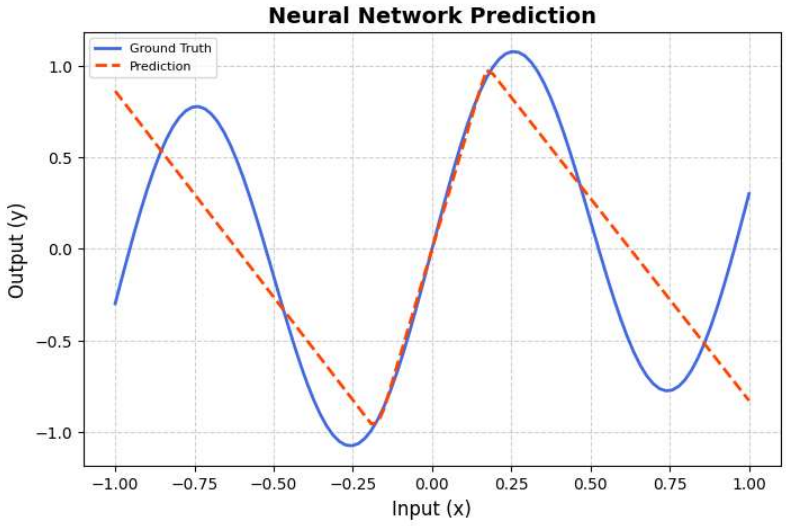
- 训练轮数：5000
- 批量大小：32
- 学习率：0.01

五、拟合效果

训练完成后，使用 predict 方法对模型的拟合效果进行可视化评估。测试过程中，生成一定范围内的输入数据，并利用训练好的神经网络计算对应的预测输出。最终，通过绘制真实函数曲线(Ground Truth)与神经网络预测曲线 (Prediction) 进行对比，以观察模型的拟合能力。

```
def predict(ranger):
    """测试模型 & 可视化结果"""
    x_plot = np.linspace(ranger[0], ranger[1], 100)                          # 生成 100 个点
    y_pred = np.array([forward(np.array([[i]])) for i in x_plot]).squeeze()     # 计算预测值并压缩维度


    plt.figure(figsize=(8, 5), dpi=100)
    plt.plot(x_plot, targetFunc(x_plot), label="Ground Truth", color="royalblue", linewidth=2)
    plt.plot(x_plot, y_pred, label="Prediction", color="orangered", linestyle="dashed", linewidth=2)


    plt.xlabel("Input (x)", fontsize=12)
    plt.ylabel("Output (y)", fontsize=12)
    plt.title("Neural Network Prediction", fontsize=14, fontweight="bold")


    plt.legend(loc="upper left", fontsize=8)
    plt.grid(True, linestyle="--", alpha=0.6)


    plt.show()
```

最终绘制出的可视化结果如下：



从图中可以看出，预测曲线与真实曲线的整体趋势相似，但未能很好地拟合复杂的波动模式，尤其是在函数的局部变化较大的区域（如极值点附近），表现出较大的偏差。这表明两层的 ReLU 网络虽然可以模拟任何函数，但在模拟相对复杂的函数时较为困难。

整体来看，目前的模型可以捕捉数据的全局趋势，但在局部拟合上仍有较大提升空间。改进方案有：增加隐藏层神经元数量（例如从 16 个增加到 32 或 64 个），增强模型的表达能力；增加训练轮数，例如从 5000 轮增加到 10000 轮，以充分训练模型；调整学习率，可能当前学习率过大或过小，影响收敛速度和最终性能。