实验主题：证明一个两层的 ReLU 网络可以模拟任何有界闭集函数

一、理论证明

根据万能近似定理，一个具有至少一个使用 “挤压” 性质激活函数（如 Sigmoid）隐藏层的前馈神经网络，在给定足够多隐藏单元的情况下，能够以任意精度逼近任何从一个有限维空间到另一个有限维空间的 Borel 可测函数。虽然 ReLU 不是“挤压”函数，但它分段线性的特性，使得两层 ReLU 神经网络通过调整权重和偏置，可以将输入空间映射到新的特征空间，然后在新空间上进行线性组合，从而构造出复杂的决策边界来逼近任意连续函数。

二、实验证明

1、函数定义：

本次实验采用的函数为正弦函数，即 sin(x)，x 取值范围为（-2pi, 2pi）。

2、数据采集

本次实验中训练集将从-2pi 到 2pi 的区间内生成等间距数值序列作为横坐标，训练数据共 500 个，测试集采样方法相同，测试数据共 200 个。

```
# 生成训练数据
x_train = np.linspace(-2 * np.pi, 2 * np.pi, 500).reshape(-1, 1)
y_train = np.sin(x_train)
```

```
# 生成测试数据
x_test = np.linspace(-2 * np.pi, 2 * np.pi, 200).reshape(-1, 1)
```

3、模型描述

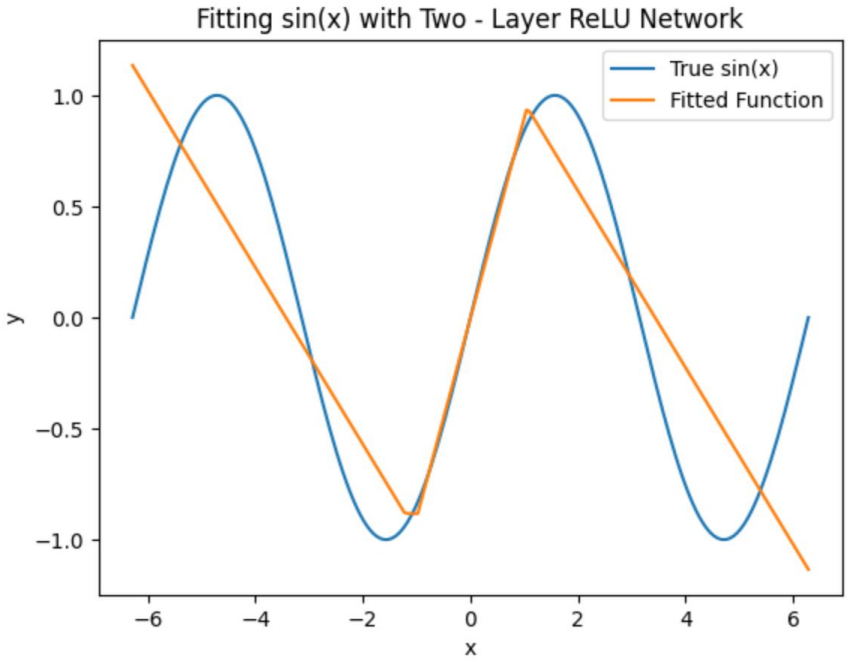
本次实验采用的是具有一个隐藏层一个输出层的神经网络模型，激活函数为 Relu 函数，即 f(x)=max(0,x)。隐藏层和输出层的初始权重值为随机值，初始偏置为 0。隐藏层大小为 100，训练轮数为 40000，学习率设置为 0.01。

```
class mymodel:
    def __init__(self, input_size, hidden_size, output_size):
        self.w1 = np.random.randn(input_size, hidden_size) * 0.01
        self.b1 = np.zeros((1, hidden_size))
        self.w2 = np.random.randn(hidden_size, output_size) * 0.01
        self.b2 = np.zeros((1, output_size))
```

```
# 初始化神经网络
input_size = 1
hidden_size = 100
output_size = 1
network = mymodel(input_size, hidden_size, output_size)
# 训练神经网络
epochs = 40000
learning_rate = 0.01
```

4、模型效果

```
Epoch 39800: Loss = 0.1403189273508129
Epoch 39900: Loss = 0.1403189220660335
```



本次实验拟合效果一般，可能的原因是两层的 Relu 网络可能还是相对简单，难以学习到 sin 函数的曲线特征。