两层 ReLU 网络通用近似能力验证

一、理论证明：为什么两层 ReLU 网络可以模拟任意函数？

1. 通用近似定理（Universal Approximation Theorem）的 ReLU 版本：

对于任意定义在紧致集（Compact Set）$K \subset R^n$ 上的连续函数 $f: K \to R$，以及任意精度 $\varepsilon > 0$，存在一个两层神经网络 $\hat{f}(x)$，其隐藏层使用 ReLU 激活函数（$ReLU(x) = \max(0, x)$），使得：

$$\sup_{x \in K} \left| f(x) - \hat{f}(x) \right| < \varepsilon$$

2. 构造性证明：

1）紧致集上的函数分解：

根据 Stone-Weierstrass 定理，连续函数 $f(x)$ 在紧致集上可以用多项式函数一致逼近。进一步，多项式可分解为多个线性段的组合。即对紧致集 K，选择有限个点 $\{x_i\}$ 划分区间，构造分段线性函数 $\hat{f}(x)$，使其在区间 $[x_i, x_{i+1}]$ 上为线性函数。则根据连续性，当划分足够细密（$\lim_{\Delta x \to 0} \left| \hat{f}(x + \Delta x) - \hat{f}(x) \right|$）时，有 $\max \left| f(x) - \hat{f}(x) \right| < \varepsilon$。

2）ReLU 的分段线性表示：

单个 ReLU 神经元 $\mathrm{ReLU}(wx + b)$ 的输出为分段线性函数，其形式为：

$$\mathrm{ReLU}(wx + b) = \begin{cases} wx + b & wx + b > 0 \\ 0 & otherwise \end{cases}$$

通过控制权重 w 和偏置 b，可以控制折点的位置和斜率。

3）多 ReLU 神经元的组合：

设隐藏层有 m 个神经元，每个神经元对应一个“折线段”。通过合理设计权重 $W_1 \in R^{1 \times m}$ 和偏置 $b_1 \in R^m$，使得隐藏层输出为：

$$h(\mathrm{x}) = \mathrm{ReLU}(W_1 x + b_1)$$

由于每个 $h_i(x)$ 贡献一个线性段，叠加后的输出层为：

$$\hat{f}(x) = W_2 h(x) + b_2$$

其中 $W_2 \in R^{m\times 1}$ 控制各段的系数，$b_2 \in R$ 为全局偏置

4）误差控制：

通过增加 m（隐藏层宽度），可以增加折线段数量，减小每个段的长度，从而降低逼近误差 $\left| f(x) - \hat{f}(x) \right|$，从而实现对任意函数的模拟。

本实验通过构造一个隐藏层为 100 个 ReLU 神经元的两层网络，验证其对非线性函数 $f(x) = x^2 + \sin 3x$ 的逼近能力。

二、函数定义

目标函数：

$$f(x) = x^2 + \sin 3x$$

选择理由：

1. 二次项（x^2）：模拟全局非线性增长趋势。
2. 正弦项（$\sin(3x)$）：引入高频周期性波动，增加函数复杂性。
3. 两者的组合形成高度非线性的混合模式，适合验证模型的表达能力。

三、数据采集

数据集	生成方式	样本数量	输入范围
训练集	均匀随机采样	100	$x \in [-5,5]$
测试集	均匀随机采样	200	$x \in [-5,5]$

代码实现：

```python
# 定义目标函数
def f(x):
    return x**2 + np.sin(3 * x)

# 生成训练和测试数据
np.random.seed(42)
x_train = np.random.uniform(-5, 5, size=(100, 1))
y_train = f(x_train)
x_test = np.random.uniform(-5, 5, size=(200, 1))
y_test = f(x_test)
```

四、模型描述

1. 网络结构示意图：

输入层 (1)　　隐藏层 (ReLU, 100)　　输出层 (线性, 1)

$$x \longrightarrow h_1, h_2, \ldots, h_{100} \longrightarrow y_{pred}$$

2. 参数初始化

对于隐藏层参数 W_1, b_1：权重 W_1 使用 He 初始化：$W_1 \sim N(0, \sqrt{2/n_{in}}$，其中 $n_{in} = 1$ 为输入维度。此方法适配 ReLU 的激活特性，缓解梯度消失。同时偏置 b_1 初始化为，保证初始阶段神经元激活均匀。

对于输出层参数 W_2, b_2：权重 W_2 初始化为小随机数，避免初始输出过大。同时偏置 b_2 初始化为 0，与目标函数的对成性匹配。

3. 前向传播

输入层：$x \in R^{N\times 1}$（N 为样本数）

隐藏层预激活：$a_1 = xW_1 + b_1 \in R^{N\times 100}$

隐藏层激活：$h = \mathrm{ReLU}(a_1) = \max(0, a_1)$

输出层：$y_{pred} = hW_2 + b_2 \in R^{N\times 1}$

4. 反向传播

设损失函数为均方误差 $L = \dfrac{1}{N}\sum_{i=1}^{N}(y^{(i)}{}_{pred} - y^{(i)}{}_{true})^2$，则有：

$$\frac{\partial L}{\partial W_2} = \frac{1}{N} h^T (y_{pred} - y_{true}),$$

$$\frac{\partial L}{\partial b_2} = \frac{1}{N} \sum_{i=1}^{N} (y^{(i)}{}_{pred} - y^{(i)}{}_{true}),$$

$$\frac{\partial L}{\partial W_1} = \frac{1}{N} x^T [(y_{pred} - y_{true}) W_2{}^T \otimes \mathrm{I}(a_1 > 0)],$$

$$\frac{\partial L}{\partial b_1} = \frac{1}{N} \sum_{i=1}^{N} [y^{(i)}{}_{pred} - y^{(i)}{}_{true}) W_2{}^T \otimes \mathrm{I}(a_1{}^{(i)} > 0)].$$

其中$\otimes$表示逐元素乘法，$\mathrm{I}(\cdot)$为指示函数

5. 训练过程

训练初期损失快速下降，后期收敛直至平稳，符合梯度下降的典型行为。梯度下降更新规则如下：

$$W_1 \leftarrow W_1 - \eta \frac{\partial L}{\partial W_1}, b_1 \leftarrow b_1 - \eta \frac{\partial L}{\partial b_1}$$

$$W_2 \leftarrow W_2 - \eta \frac{\partial L}{\partial W_2}, b_2 \leftarrow b_2 - \eta \frac{\partial L}{\partial b_2}$$

其中学习率$\eta = 0.001$，通过小步长更新保证稳定性。

五、拟合效果

1. 损失下降结果

```
PS E:\DeepLearning\DeepLearning-TJU-WCY> & D:/Programs/python/python.exe "e:/DeepLearning/DeepLearning-TJU-WCY/chap4_ simple neural network/test.py"
Epoch 0, Loss: 136.4771
Epoch 1000, Loss: 4.9541
Epoch 2000, Loss: 4.0147
Epoch 3000, Loss: 3.6570
Epoch 4000, Loss: 3.5153
Epoch 5000, Loss: 3.4491
Epoch 6000, Loss: 3.4063
Epoch 7000, Loss: 3.3674
Epoch 8000, Loss: 3.3207
Epoch 9000, Loss: 3.2579
Epoch 10000, Loss: 3.1681
Epoch 11000, Loss: 3.0355
Epoch 12000, Loss: 2.8498
Epoch 13000, Loss: 2.6057
Epoch 14000, Loss: 2.3130
Epoch 15000, Loss: 2.0087
Epoch 16000, Loss: 1.7276
Epoch 17000, Loss: 1.4859
Epoch 18000, Loss: 1.2656
Epoch 19000, Loss: 1.0586
Epoch 20000, Loss: 0.9189
Epoch 21000, Loss: 0.8139
Epoch 22000, Loss: 0.7326
Epoch 23000, Loss: 0.6739
Epoch 24000, Loss: 0.6301
Epoch 25000, Loss: 0.5974
```

```
Epoch 476000, Loss: 0.1037
Epoch 477000, Loss: 0.1037
Epoch 478000, Loss: 0.1037
Epoch 479000, Loss: 0.1037
Epoch 480000, Loss: 0.1037
Epoch 481000, Loss: 0.1037
Epoch 482000, Loss: 0.1037
Epoch 483000, Loss: 0.1037
Epoch 484000, Loss: 0.1037
Epoch 485000, Loss: 0.1037
Epoch 486000, Loss: 0.1037
Epoch 487000, Loss: 0.1037
Epoch 488000, Loss: 0.1037
Epoch 489000, Loss: 0.1037
Epoch 490000, Loss: 0.1037
Epoch 491000, Loss: 0.1037
Epoch 492000, Loss: 0.1037
Epoch 493000, Loss: 0.1037
Epoch 494000, Loss: 0.1037
Epoch 495000, Loss: 0.1037
Epoch 496000, Loss: 0.1037
Epoch 497000, Loss: 0.1037
Epoch 498000, Loss: 0.1037
Epoch 499000, Loss: 0.1037
Test Loss: 0.1034
```

2. 可视化对比