2.1 函数拟合题目报告

1. 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。

2. 要求

(1) 请自行在函数上采样生成训练集和测试集，使用训练集来训练神经网络，使用测试集来验证拟合效果。

(2) 可以使用深度学习框架来编写模型，如 tensorflow、pytorch、keras 等。

如果不使用上述框架，直接用 NumPy 实现可以最高加 5 分的附加分。

(3)提交时请一并提交代码和报告。

a. 代码建议注释清楚（5 分）

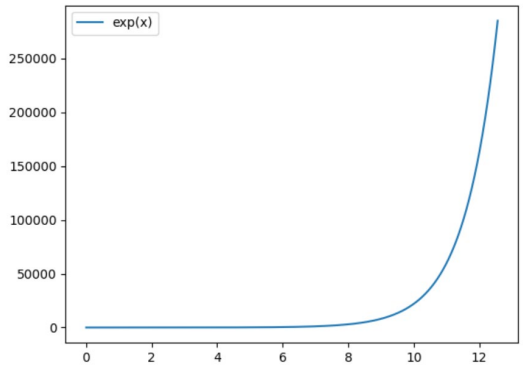
b. 报告至少应包含以下部分：（5 分）

函数定义、数据采集、模型描述、拟合效果。

3. 函数定义

(1) 目标函数类型

本题目的问题为证明一个两层的 ReLU 网络可以模拟任何函数。因此设目标函数为指数函数，范围是$(0, 4\pi)$，步长为 0.01。函数图像如下：



(2) 代码设计

```
def targetFunc(x):
    return np.exp(x)

x = np.arange(0, 4 * np.pi, 0.01)
plt.plot(x, [targetFunc(i) for i in x])
plt.legend(["exp(x)"])
```

4. 数据采集

(1) 采集方式

对于数据集的采集，数据从目标函数中进行等距采集。自变量 x 使用步长为 0.0005，范

围是从 0 到 2π 的等间隔数组，利用 3.中的目标函数设计求得函数值。

对于数据集的划分，将数据划分为训练集、验证集和测试集三部分。测试集占总数据的 20%，训练集占 80%。接下来从训练集中再划分出一部分作为验证集，验证集占训练集的 12.5%。因此，最终训练集占原始数据的 70%，验证集占 10%，测试集占 20%。

另外，random_state=1 用于确保每次划分的结果一致，便于复现实验。

(2) 代码实现

```
x = np.arange(0, 2*np.pi, 0.0005) # 采集数据的步长
y = [targetFunc(i) for i in 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)
```

5. 模型描述

依照深度学习框架，使用 NumPy 实现训练模型。

(1) 模型结构

模型输入层神经元个数为 1，表示输入数据是一个标量。隐藏层的神经元数量 64，使用 ReLU 激活函数。输出层输出神经元个数为 1，表示输出数据也是一个标量。输入层到隐含层，隐含层到输出层，满足模型要求为一个两层的 ReLU 网络。

(2) 参数设计

分别为两层神经网络设计权重矩阵 W1, W2 和偏置向量 b1, b2。其中，权重矩阵使用随机数方法初始化，偏执矩阵则统一初始化为 0。

(3) 前向传播

对于层级之间的传播，使用线性计算

$$z = W \cdot x + b$$

公式计算得出。隐含层使用 ReLU 激活函数进行映射。

(4) 损失函数

使用均方误差（MSE）作为损失函数。

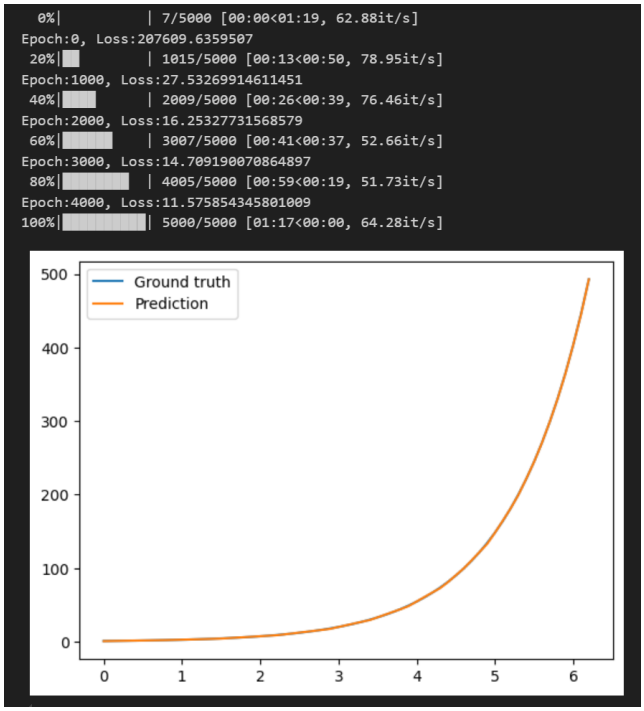
```
def loss_fn(y_pred, target):
    loss = np.sum((y_pred - target) * (y_pred - target), axis=1,
keepdims=True) / len(y_pred)
    cache['loss'] = (y_pred - target) / len(y_pred)
    return loss
```

(5) 反向传播

计算输出层、隐含层的梯度进行反向参数更新。其中，lr 为学习率。

6. 拟合效果

实验结果如图。可以看到函数预测值与真实值基本重合。实验成立。



参考资料

[1] G. Cybenko. 1989. Approximation by superpositions of a sigmoidal function.

[2] K. Hornik, M. Stinchcombe, and H. White. 1989. Multilayer feedforward networks are universal approximators.

[3] Moshe Leshno, et al. 1993. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function

[4] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines.

[5] Xavier Glorot, Antoine Bordes, Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. PMLR 15:315-323.