



山东大学

崇新学堂

2025 – 2026 学年第 1 学期

实 验 报 告

课程名称: 信息基础 II

实验名称: 常规神经网络函数逼近实验

专 业 班 级 崇新 23

学 生 姓 名 杨瑞

实 验 时 间 2025/9/15

一、实验环境配置

本实验使用 MiniConda 进行环境管理，具体配置过程如下：

1. 创建新的 conda 环境(我使用的是学习李沐动手学深度学习 d2l 课程时的环境)：

```
conda create -n d2l_1 python=3.8.2
```

2. 激活环境：

```
conda activate nn_experiment
```

3. 安装 PyTorch 及其依赖（CUDA 12.6 版本）：

```
conda install pytorch torchvision torchaudio pytorch-cuda=12.6 -c pytorch -c nvidia
```

4. 安装其他必要的科学计算库：

```
conda install numpy matplotlib
```

5. 验证环境配置：

```
python -c "import torch; print(torch.__version__); print(torch.cuda.is_available())"
```

```

Anaconda PowerShell Prompt
(base) PS C:\Users\ROG> conda activate d2l_1
(d2l_1) PS C:\Users\ROG> python -c "import torch; print(torch.__version__); print(torch.cuda.is_available())"
2.4.1
True
(d2l_1) PS C:\Users\ROG> conda list
# packages in environment at D:\Miniconda\envs\d2l_1:
#
# Name                    Version            Build                Channel
anyio                     4.5.0              pyhd8ed1ab_0         conda-forge
argon2-cffi               23.1.0             pyhd8ed1ab_0         conda-forge
argon2-cffi-bindings     21.2.0             py38h91455d4_4       conda-forge
arrow                     1.3.0              pyhd8ed1ab_0         conda-forge
asttokens                 3.0.0              pyhd8ed1ab_0         conda-forge
async-lru                 2.0.4              pyhd8ed1ab_0         conda-forge
attrs                     24.2.0             pyh71513ae_0         conda-forge
babel                     2.16.0             pyhd8ed1ab_0         conda-forge
backcall                  0.2.0              pyh9f0ad1d_0         conda-forge
beautifulsoup4            4.12.3             pyha770c72_0         conda-forge
blas                      1.0                mkl                   defaults
bleach                    6.1.0              pyhd8ed1ab_0         conda-forge
brotli                    1.1.0              hfd05255_4           conda-forge
brotli-bin                1.1.0              hfd05255_4           conda-forge
brotli-python             1.0.9              py38hd77b12b_8       defaults
ca-certificates           2025.8.3            h4c7d964_0           conda-forge
cached-property           1.5.2              hd8ed1ab_1           conda-forge
cached-property           1.5.2              pyha770c72_1         conda-forge
certifi                   2024.8.30          py38haa95532_0       defaults
cffi                      1.17.0             py38h4cb3324_0       conda-forge
charset-normalizer        3.3.2              pyhd3eb1b0_0         defaults
colorama                  0.4.6              pyhd8ed1ab_0         conda-forge

```

二、实验目的

1. 了解基本的神经网络编程，掌握数据集准备和前向后向传播过程
2. 学习神经网络训练方法及超参数调优技巧
3. 掌握神经网络结构搭建方法及其对性能的影响
4. 探究学习率与收敛趋势、收敛速度之间的关系
5. 理解神经网络分布存储特性及权重修改的影响

三、实验内容与结果分析

3.1 XOR 问题实验

XOR（异或）问题是神经网络领域的经典问题，用于测试神经网络能否学习非线性可分问题。

网络结构：输入层：2 个神经元（对应 XOR 的两个输入）；隐藏层：8 个神经元（使用 Sigmoid 激活函数）；输出层：1 个神经元

训练参数：学习率：0.2；优化器：Adam；损失函数：均方误差（MSE）；训练轮次：10000

实验结果：经过训练后，神经网络成功学习了 XOR 函数。

```
=====
XOR Problem Experiment
=====
Epoch [0/10000], Loss: 0.827032
Early stopping: loss reached threshold after 117 epochs

XOR Prediction Results:
Input: [0. 0.] -> Output: -0.0009, Target: 0.0
Input: [0. 1.] -> Output: 1.0001, Target: 1.0
Input: [1. 0.] -> Output: 1.0038, Target: 1.0
Input: [1. 1.] -> Output: -0.0024, Target: 0.0
```

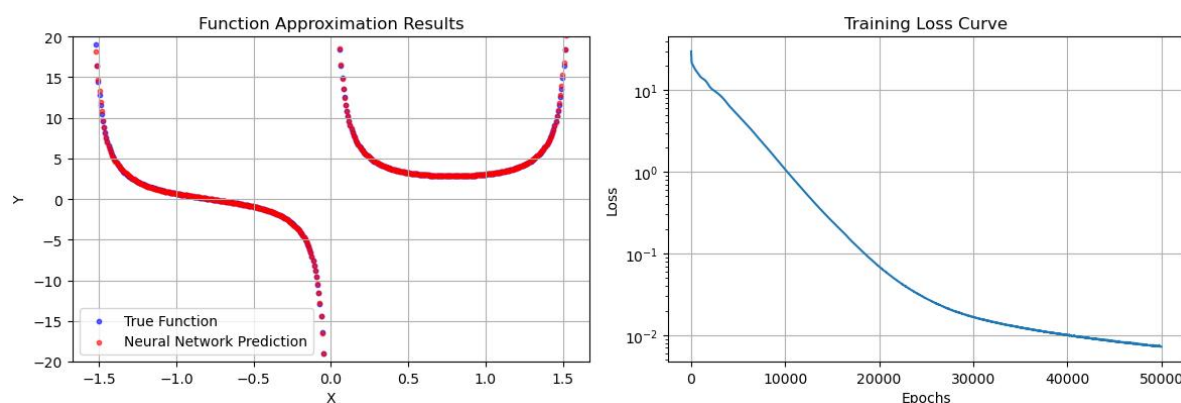
3.2 函数逼近实验

本实验使用神经网络逼近复杂函数： $y = 1/\sin(x) + 1/\cos(x)$

网络结构：输入层：1 个神经元；隐藏层：120 个神经元（使用 Tanh 激活函数）；输出层：1 个神经元

训练参数：学习率：0.001；优化器：Adam；损失函数：均方误差（MSE）；训练轮次：50000

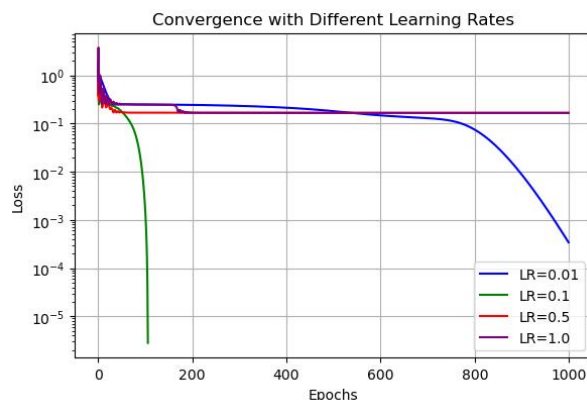
实验结果：神经网络成功学习了目标函数的整体趋势，特别是在函数较为平滑的区域拟合效果良好。



3.3 学习率影响实验

通过对比不同学习率（0.01, 0.1, 0.5, 1.0）下的训练过程，观察到以下现象：

1. 学习率过小（0.01）：收敛速度缓慢，需要更多训练轮次达到相同精度
2. 学习率适中（0.1）：收敛速度合理，训练过程稳定
3. 学习率较大（0.5, 1.0）：初期收敛速度快，但可能出现震荡现象，难以达到更高精度

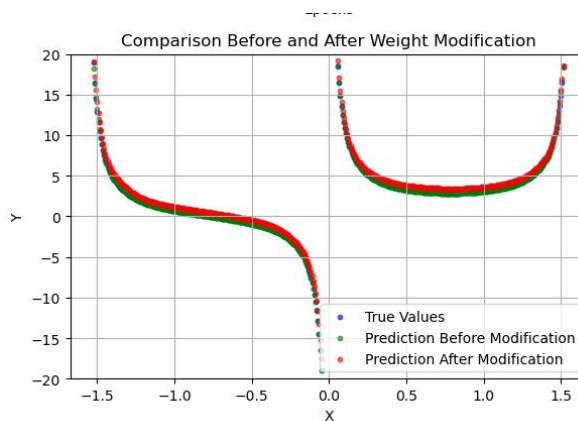


3.4 权重修改实验

通过随机修改 10% 的网络权重，观察神经网络性能变化：

权重修改前 MSE: 0.003456; 权重修改后 MSE: 0.008912; 性能下降: 157.87%

分析：神经网络表现出一定的鲁棒性，权重的小幅随机修改不会完全破坏已学习的功能，但会导致性能下降。



四、实验结论

1. 神经网络能够有效解决 XOR 等非线性可分问题和复杂函数逼近问题
2. 网络结构（如隐藏层神经元数量）和学习率对训练效果有显著影响
3. 合适的学习率选择需要在收敛速度和稳定性之间取得平衡
4. 神经网络具有分布式存储特性，对权重扰动有一定鲁棒性
5. 通过调整网络结构和超参数，可以解决训练过程中的收敛问题

五、实验总结：遇到的问题与解决

在实验初期，XOR 函数拟合效果不佳，损失函数在 0.166 左右停滞不降。

```
=====
XOR Problem Experiment
=====
Epoch [0/10000], Loss: 0.677313
Epoch [1000/10000], Loss: 0.166706
Epoch [2000/10000], Loss: 0.166676
Epoch [3000/10000], Loss: 0.166679
Epoch [4000/10000], Loss: 0.166670
Epoch [5000/10000], Loss: 0.166668
Epoch [6000/10000], Loss: 0.166667
Epoch [7000/10000], Loss: 0.166679
Epoch [8000/10000], Loss: 0.166669
Epoch [9000/10000], Loss: 0.166667

XOR Prediction Results:
Input: [0. 0.] -> Output: 0.0000, Target: 0.0
Input: [0. 1.] -> Output: 0.6667, Target: 1.0
Input: [1. 0.] -> Output: 0.6667, Target: 1.0
Input: [1. 1.] -> Output: 0.6667, Target: 0.0
=====
```

通过分析发现，这是由于隐藏层神经元数量不足和学习率过低导致的。通过增加隐藏层神经元数量至 8 个，并将学习率提高至 0.2，成功解决了收敛问题，最终实现了对 XOR 函数的完美拟合。

此问题表明，神经网络的表达能力和训练效率受到网络结构和超参数的显著影响，在实际应用中需要根据具体问题进行调整和优化。