ReLU 神经网络拟合自定义函数的实验报告

2252699 王鉴戈

1. 引言

本实验旨在验证 **两层 ReLU 神经网络可以拟合任意函数** 的理论。我们使用 PyTorch 构建一个简单的 **两层 ReLU 网络** 来拟合一个 **目标函数**，并评估其拟合效果。

2. 函数定义

本实验自定义了如下目标函数：
$f(x) = \log(1 + e^x)$
该函数是一个 **平滑的非线性函数**，具有指数增长特性。该函数的可视化如下：

```
import torch
import matplotlib.pyplot as plt

def target_function(x):
    return torch.log(1 + torch.exp(x))

x = torch.linspace(-10, 10, 1000)
y = target_function(x)
plt.plot(x.numpy(), y.numpy())
plt.xlabel("x")
plt.ylabel("f(x)")
plt.title("Target Function: log(1 + exp(x))")
plt.show()
```

3. 数据采集

我们从区间 **[-10, 10]** 生成 **1000 个样本点** 作为数据集，并将数据集划分为：

- 训练集（64%）
- 验证集（16%）
- 测试集（20%）

数据的划分使用 `train_test_split`，具体代码如下：

```
from sklearn.model_selection import train_test_split
import torch.utils.data as DataLoader

X = torch.linspace(-10, 10, 1000).reshape(-1, 1)
y = target_function(X)

# 80% 训练 + 20% 测试
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)
# 训练集再划分 80% 训练 + 20% 验证
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,
test_size=0.2)

# 数据加载
train_dataset = DataLoader.TensorDataset(X_train, y_train)
val_dataset = DataLoader.TensorDataset(X_val, y_val)
test_dataset = DataLoader.TensorDataset(X_test, y_test)
```

---

4. 神经网络模型

模型架构

- 输入层: 1 维 (`x`)
- 隐藏层: 10 个神经元，使用 **ReLU** 作为激活函数
- 输出层: 1 维 (`f(x)`)

模型代码如下：

```
import torch.nn as nn
import torch.nn.functional as F

class MyModel(nn.Module):
    def __init__(self):
        super(MyModel, self).__init__()
        self.fc1 = nn.Linear(1, 10)  # 1 -> 10
        self.fc2 = nn.Linear(10, 1)  # 10 -> 1

    def forward(self, x):
        x = F.relu(self.fc1(x))  # ReLU 激活
        x = self.fc2(x)  # 线性输出
        return x
```

---

5. 训练过程

损失函数

我们使用 均方误差（**MSE Loss**） 作为损失函数：

```
criterion = nn.MSELoss()
```

优化器

使用 **Adam** 优化器 进行梯度更新：

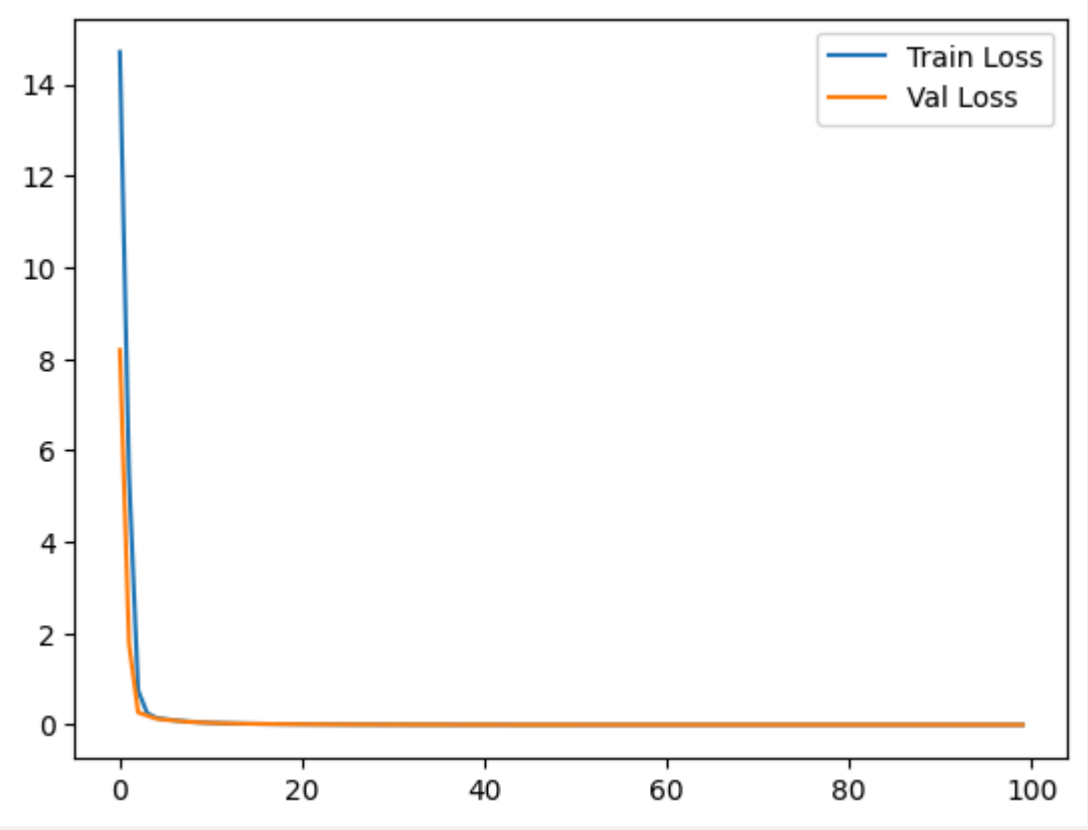
```
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
```

训练循环

```
def train_one_epoch(model, train_loader, optimizer):
    model.train()
    train_loss = 0
    for X_train, y_train in train_loader:
```

```
        optimizer.zero_grad()
        y_pred = model(X_train)
        loss = criterion(y_pred, y_train)
        loss.backward()
        optimizer.step()
        train_loss += loss.item()
    return train_loss / len(train_loader)

def train(model, train_loader, val_loader, optimizer, epochs):
    train_losses = []
    val_losses = []
    for epoch in range(epochs):
        train_loss = train_one_epoch(model, train_loader,
optimizer)
        val_loss = validate(model, val_loader)
        train_losses.append(train_loss)
        val_losses.append(val_loss)
        print(f"Epoch {epoch+1}/{epochs}, Train Loss: {train_loss},
Val Loss: {val_loss}")
    return train_losses, val_losses
```

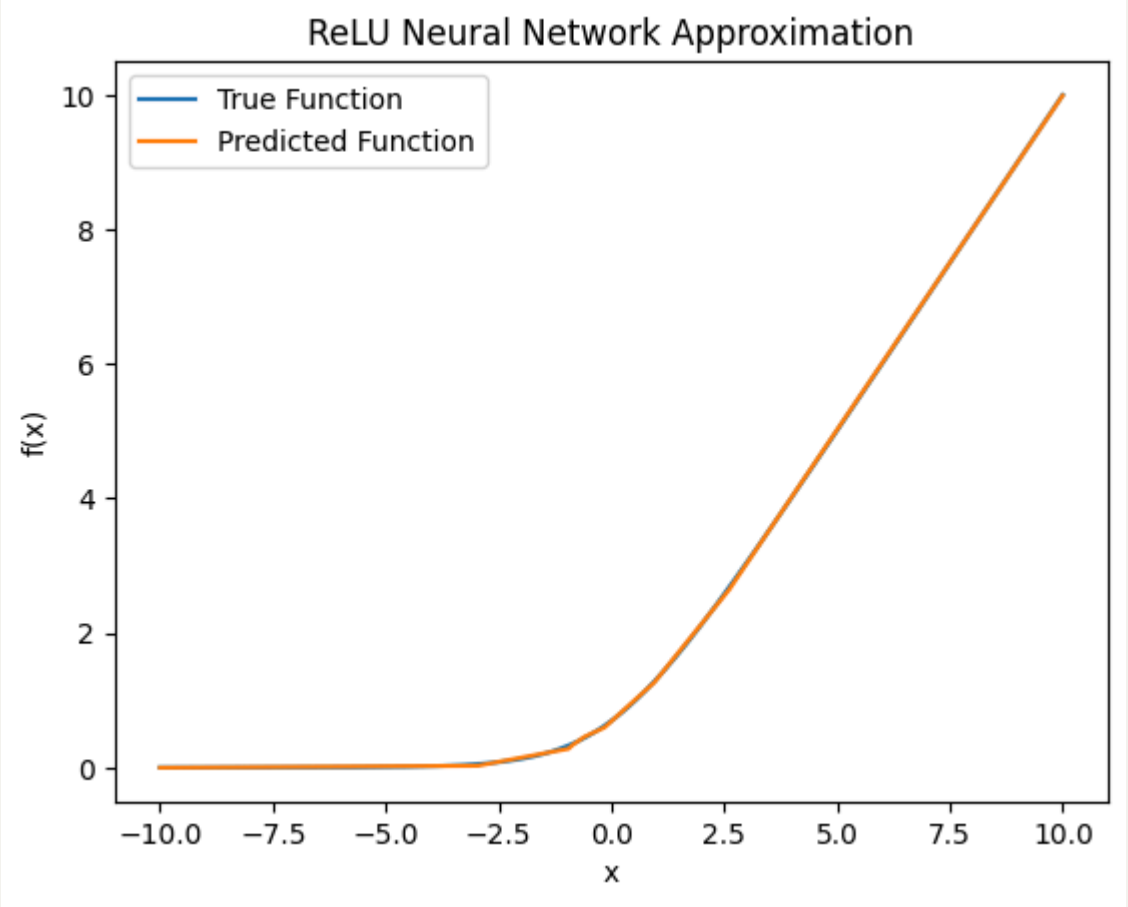


6. 训练结果与拟合效果

我们绘制了 **训练后的神经网络拟合结果**，与原始函数对比：

```
# Test the model
test_loader = DataLoader.DataLoader(test_dataset, batch_size=32,
shuffle=False)
model.eval()
with torch.no_grad():
    y_pred = model(X)
    #多种评价指标拟合效果，包括MSE, MAE, R2, RMSE
    mse = criterion(y_pred, y)
    mae = F.l1_loss(y_pred, y)
    r2 = 1 - mse / torch.var(y)
    rmse = torch.sqrt(mse)
    print(f"MSE: {mse}, MAE: {mae}, R2: {r2}, RMSE: {rmse}")

plt.plot(X.numpy(), y.numpy(), label='True Function')
plt.plot(X.numpy(), y_pred.numpy(), label='Predicted Function')
plt.legend()
plt.xlabel("x")
plt.ylabel("f(x)")
plt.title("ReLU Neural Network Approximation")
plt.show()
```



实验结论

1. 拟合效果良好：
    - 训练的 **ReLU 神经网络** 能够很好地拟合 `log(1 + exp(x))` 函数
2. **ReLU 局限性**：
    - 由于 **ReLU 会导致负数部分梯度为 0**
    - 可能的改进方法：
        - 增加 **隐藏层** 或 **神经元数量**
        - 使用 **Leaky ReLU** 代替 ReLU 以防止梯度消失
3. 实验成功验证理论：
    - 通过实验，我们验证了 **两层 ReLU 网络能够近似任意函数**。

---

7. 结论

本实验成功实现了 **两层 ReLU 神经网络** 对 **log(1 + exp(x))** 函数的拟合，验证了 ReLU 网络的**通用逼近能力**。实验结果表明，适当的网络架构和训练方式可以使神经网络逼近任意非线性函数。未来可以尝试 **更深的网络结构** 以提升拟合效果，并进一步优化 ReLU 的局限性。