chpt_4 - 2 Relu函数拟合报告

2154071 张博文

1、函数定义

- train

```
'''
    @ brief         训练神经网络
    @ param
        model       模型
        criterion   损失函数
        optimizer   优化器
        x_train     特征
        y_train     标签
        epochs      训练轮次
'''
def train(model, criterion, optimizer, x_train, y_train, epochs):
    model.train()
    for epoch in range(epochs):
        inputs = torch.tensor(x_train, dtype=torch.float32).unsqueeze(1)
        targets = torch.tensor(y_train, dtype=torch.float32).unsqueeze(1)

        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, targets)
        loss.backward()
        optimizer.step()

        # 每100轮打印一次训练结果
        if (epoch+1) % 100 == 0:
            print(f"Epoch [{epoch+1}/{epochs}], Loss: {loss.item():.4f}")
```

- test

```
'''
    @ brief         得到验证结果
    @ param
```

```
        model       模型
        x_test      特征
    @ return        预测值
'''
def test(model, x_test):
    model.eval()
    with torch.no_grad():
        inputs = torch.tensor(x_test,
dtype=torch.float32).unsqueeze(1)
        outputs = model(inputs)
        predictions = outputs.squeeze(1).numpy()
    return predictions
```

2、数据采集

在本实验中使用该神经网络模型拟合 Sin 函数，在 -2*pi ~ 2*pi 范围内均匀生成 1000 个点，随机划分 800 个点为训练集，200 个点为测试集。

- generate_data 函数

```
'''
    @ brief     生成样本数据
    @ param     样本规模
    @ return
'''
def generate_data(n_samples):
    np.random.seed(0)
    x = np.random.uniform(-2*np.pi, 2*np.pi, n_samples)
    y = np.sin(x)
    return x, y
```

- 生成数据集（1000），随机划分测试集（800）和验证集（200）

```
# 生成数据集(1000)
x_data, y_data = generate_data(1000)

# 随机划分训练集(800)和测试集(200)
indices = np.random.permutation(len(x_data))
train_indices, test_indices = indices[:800], indices[800:]
x_train, y_train = x_data[train_indices], y_data[train_indices]
x_test, y_test = x_data[test_indices], y_data[test_indices]
```

3、模型描述

按照题目要求，使用 3 个线性层 + 2 个 ReLU 激活函数做隐藏层来构成神经网络。使用 pytorch 框架实现。

- ReLU_Net

```
class ReLU_Net(nn.Module):
    # 定义三个线性层，两个ReLU层连接作隐藏层
    def __init__(self):
        super(ReLU_Net, self).__init__()
        self.fc1 = nn.Linear(1, 64)
        self.fc2 = nn.Linear(64, 16)
        self.fc3 = nn.Linear(16, 1)
        self.relu = nn.ReLU()

    # 前向传播
    def forward(self, x):
        h1 = self.relu(self.fc1(x))
        h2 = self.relu(self.fc2(h1))
        y = self.fc3(h2)
        return y
```

损失函数采用均方误差损失函数，优化器使用 Adam 优化器。

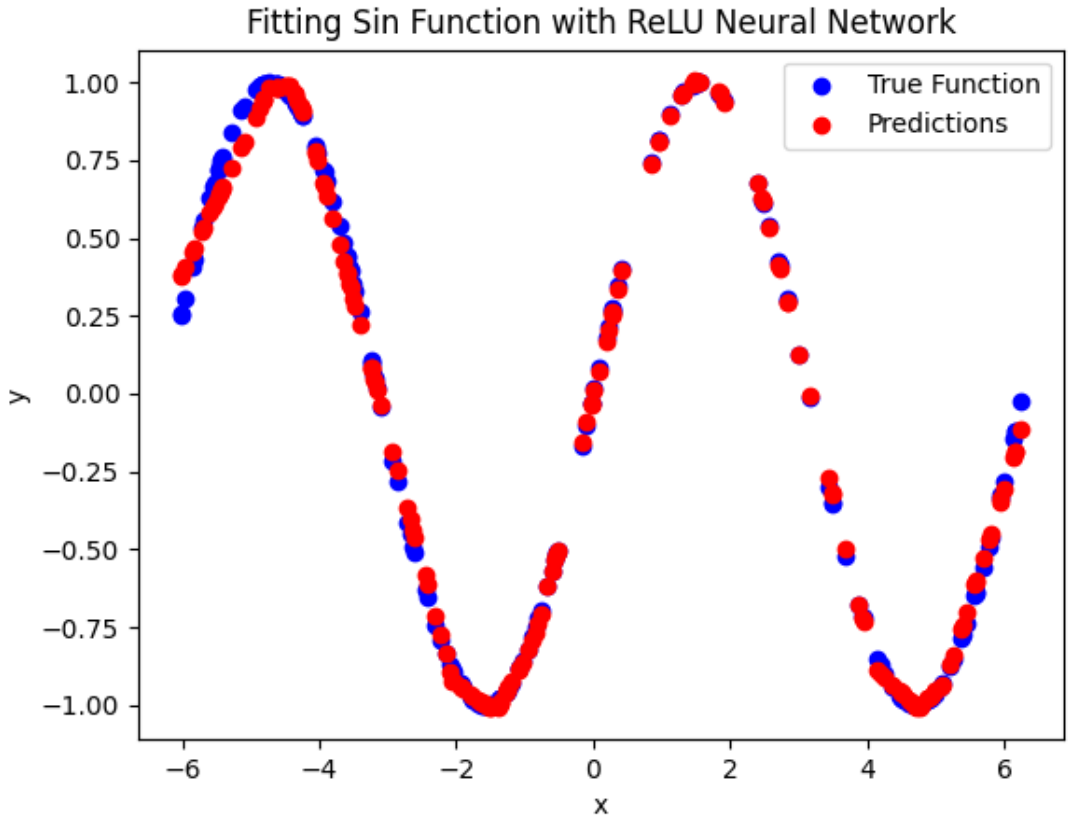
```
# 定义神经网络模型、损失函数和优化器
model = ReLU_Net()
criterion = nn.MSELoss()                                    # 均方误差
optimizer = optim.Adam(model.parameters(), lr=0.001)    # Adam
```

4、拟合效果

拟合效果较好，训练一千轮次后的损失值在 0.004 以下。

```
Epoch [100/1000], Loss: 0.1343
Epoch [200/1000], Loss: 0.1082
Epoch [300/1000], Loss: 0.0879
Epoch [400/1000], Loss: 0.0631
Epoch [500/1000], Loss: 0.0415
Epoch [600/1000], Loss: 0.0265
Epoch [700/1000], Loss: 0.0157
Epoch [800/1000], Loss: 0.0082
Epoch [900/1000], Loss: 0.0040
Epoch [1000/1000], Loss: 0.0023
```

拟合图像：


Fitting Sin Function with ReLU Neural Network
True Function
Predictions
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y
−6
−4
−2
0
2
4
6
x