pytorch版本报告

一、函数定义

1. 目标函数

代码中定义了一个目标函数 `target_function`，其数学表达式为 $y=3x^3+x^2+2x+1$。该函数接收一个输入 x，并返回对应的 y 值。在代码中，它的作用是生成训练和测试数据，作为模型需要学习和拟合的真实数据。以下是其代码实现：

```
def target_function(x):
    return 3*x**3+ x**2 + 2*x + 1
```

2. 神经网络模型

代码中定义了一个名为 `ReLUNet` 的两层 ReLU 神经网络模型，继承自 `nn.Module`。该模型包含两个全连接层：

- `self.fc1`：输入维度为 1，输出维度为 10，用于将输入特征映射到 10 个隐藏单元。
- `self.fc2`：输入维度为 10，输出维度为 1，用于将隐藏单元的输出映射到最终的预测值。

在 `forward` 方法中，输入数据首先通过 `self.fc1` 层，然后经过 ReLU 激活函数进行非线性变换，最后通过 `self.fc2` 层得到最终的预测结果。以下是模型的代码实现：

```
class ReLUNet(nn.Module):
    def __init__(self):
        super(ReLUNet, self).__init__()
        self.fc1 = nn.Linear(1, 10)
        self.fc2 = nn.Linear(10, 1)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

二、数据采集

1. 训练数据

使用 `np.random.uniform(-1, 1, 100)` 函数从 [−1,1] 区间内均匀随机采样 100 个点作为输入特征 *x**tr**ain*，然后将这些点输入到目标函数 `target_function` 中，得到对应的输出值 *y**tr**ain*。这样就生成了 100 个训练样本 (*x**tr**ain*,*y**tr**ain*)。

2. 测试数据

使用 `np.linspace(-1, 1, 100)` 函数在 [−1,1] 区间内均匀生成 100 个点作为输入特征 *x**tes**t*，同样将这些点输入到目标函数 `target_function` 中，得到对应的输出值 *y**tes**t*。这些样本用于评估模型在未见过的数据上的泛化能力。

3. 数据转换

将生成的 `numpy` 数组转换为 `PyTorch` 张量，并增加一个维度，以便于输入到神经网络模型中。

```
x_train_torch = torch.tensor(x_train, dtype=torch.float32).unsqueeze(1)
y_train_torch = torch.tensor(y_train, dtype=torch.float32).unsqueeze(1)
x_test_torch = torch.tensor(x_test, dtype=torch.float32).unsqueeze(1)
y_test_torch = torch.tensor(y_test, dtype=torch.float32).unsqueeze(1)
```

三、模型描述

1. 模型结构

模型是一个简单的两层全连接神经网络，包含一个输入层、一个隐藏层和一个输出层。输入层接收一个特征，隐藏层有 10 个神经元，输出层输出一个预测值。隐藏层使用 ReLU 激活函数引入非线性，使得模型能够学习到更复杂的函数关系。

2. 损失函数

使用均方误差损失函数 `nn.MSELoss()` 来衡量模型预测值与真实值之间的差异。均方误差损失函数的数学表达式为：*MSE=n*1∑*i*=1*n*(*y***i*−*y*^*i*)2，其中 *y***i* 是真实值，*y*^*i* 是预测值，*n* 是样本数量。

3. 优化器

使用 Adam 优化器来更新模型的参数，学习率设置为 0.02。Adam 优化器结合了 AdaGrad 和 RMSProp 的优点，能够自适应地调整每个参数的学习率，使得模型在训练过程中更加稳定和高效。

4. 训练过程

模型训练了 1000 个 epoch。在每个 epoch 中，首先将优化器的梯度清零，然后通过模型进行前向传播得到预测值，计算预测值与真实值之间的损失，接着进行反向传播计算梯度，最后使用优化器更新模型的参数。每 100 个 epoch 打印一次当前的损失值，并记录每个 epoch 的损失值到 `loss_history` 列表中。

```
epochs = 1000
loss_history = []

for epoch in range(epochs):
    optimizer.zero_grad()
    y_pred = model(x_train_torch)
    loss = criterion(y_pred, y_train_torch)
    loss.backward()
    optimizer.step()

    if epoch % 100 == 0:
        print(f'Epoch {epoch}, Loss: {loss.item():.4f}')

    loss_history.append(loss.item())
```

四、拟合效果

1. 真实值与预测值对比

在训练完成后，使用训练好的模型对测试数据进行预测，并将预测结果与真实值进行可视化对比。通过绘制真实值和预测值的曲线，可以直观地观察到模型的拟合效果。

```
plt.figure(figsize=(10, 6))
plt.plot(x_test, y_test, label='True Values', color='blue')
plt.plot(x_test, y_pred_torch, label='Predicted Values', color='green')
plt.title('True vs Predicted Values')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```

2.可视化结果

从可视化结果可以看出，模型的预测值与真实值较为接近，说明模型能够较好地拟合目标函数。