深度学习 Lab1

PB20111686 黄瑞轩

1 实验目标

使用 pytorch 手写一个前馈神经网络，用于近似函数 $f(x) = \log_2(x) + \cos\left(\frac{\pi x}{2}\right), x \in [1, 16]$，并研究网络深度、学习率、网络宽度、激活函数对模型性能的影响。

2 实验过程和关键代码展示

注：为了展示核心功能，某些代码中一些无关紧要的部分在报告中被删去了。

2.1 数据生成

使用 numpy 按 $f(x)$ 生成三个数据集 train, verify, test ，保存到三个单独的 csv 文件中。超参数 n 控制 train 数据集的大小，verify 和 test 的大小分别是 $0.5n$ 和 $0.3n$ 。

> **注**：为了防止数据泄露，先随机均匀地生成 $1.8n$ 个数据，然后使用 `drop_duplicate` 函数去重，再根据比例划分数据集并存储。

```
def generate_data(x_min: float, x_max: float, n: int):
    x = np.random.uniform(x_min, x_max, size=int(n*sum(GD_RATES)))
    y = np.log2(x) + np.cos(np.pi * x/2)
    df = pd.DataFrame({'x': x, 'y': y})
    df.drop_duplicates(keep='first', inplace=True)

    length = df.shape[0]
    df1 = df.iloc[range(int(n*GD_RATES[0]))]
    df2 = df.iloc[range(int(n*GD_RATES[0]) + 1, int(n*GD_RATES[0]) + int(n*GD_RATES[1]))]
    df3 = df.iloc[range(int(n*GD_RATES[0]) + int(n*GD_RATES[1]) + 1, length)]
    df1.to_csv('data_train.csv', index=False)
    df2.to_csv('data_verify.csv', index=False)
    df3.to_csv('data_test.csv', index=False)
```

2.2 网络搭建

继承 `nn.Module` 搭建神经网络，采用 **输入层—多个隐藏层—输出层** 的结构。比如，采取一组特定超参数的网络结构如下：

```
class MyModel(nn.Module):
    def __init__(self, hidden_size: int):
        super(MyModel, self).__init__()
        self.layers = nn.Sequential(
            nn.Linear(1, hidden_size),
```

```
            nn.Sigmoid(),
            nn.Linear(hidden_size, hidden_size),
            nn.Sigmoid(),
            nn.Linear(hidden_size, hidden_size),
            nn.Sigmoid(),
            nn.Linear(hidden_size, 1)
        )

    def forward(self, x):
        return self.layers(x)
```

这个网络激活函数采用的是 Sigmoid，网络宽度由 `hidden_size` 参数决定。

2.3 训练框架

整个实验过程大致如下：

- 使用网络在训练集上训练
- 使用训练的模型在验证集上验证，查看 MSE 等指标曲线
- 调参，获得较好效果后在测试集上测试，得到 MSE 指标

训练集上的训练框架如下：

```
model = MyModel(hidden_size=64)                    # 实例化模型
criterion = nn.MSELoss()                           # 损失函数
optimizer = SGD(model.parameters(), lr=0.01)       # 优化函数
epochs = 100                                       # 训练轮数

for epoch in range(epochs):
    for x, y in data_train_loader:                 # 使用 DataLoader 进行数据加载
        y_pred = model(x)                          # 获取预测值，然后传播误差进行参数更新
        loss = criterion(y_pred, y)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

2.4 训练指标可视化

使用 Matplotlib 进行可视化，主要有两部分：

- 训练过程中 Loss 随训练轮数的变化，以及验证集上每个 Batch 的 Loss 情况
- 最终测试集上的预测情况，即将测试集的 y 和预测的 y 值画在一张图上

3 超参数的调节

本实验采用单一变量的对照法，即在一组训练结果不错的超参数上每次选择一个超参数进行调参，然后将每个超参数最好的取值组合在一起。

3.1 基础超参数

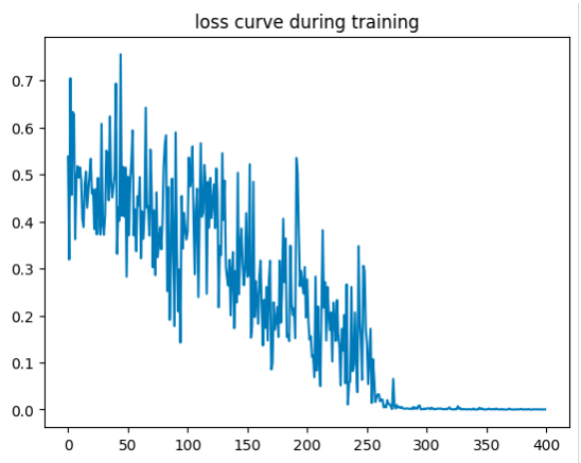
用作调参基础的各超参数配置如下所示，分别展示了网络结构、Epoch 大小、Batch 大小和数据集大小。

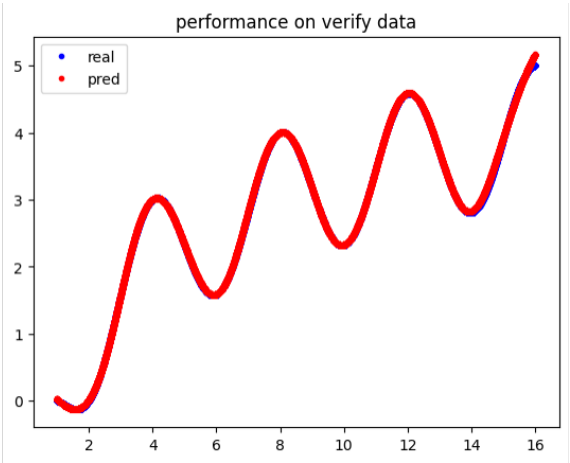
```
Sequential(
  (0): Linear(in_features=1, out_features=64, bias=True)
  (1): Sigmoid()
  (2): Linear(in_features=64, out_features=64, bias=True)
  (3): Sigmoid()
  (4): Linear(in_features=64, out_features=64, bias=True)
  (5): Sigmoid()
  (6): Linear(in_features=64, out_features=1, bias=True)
)
epochs=400, batches=32, lr=0.01
dataset cases:
  train set: 50000 cases
  verify set: 25000 cases
  test set: 15000 cases
```

其训练结果如下，300 轮训练后验证集上的 Loss 基本收敛。

```
MSE:
  average MSE of trainning is 0.213553
  minimum MSE of trainning is 0.000037
  MSE of verifying is 0.000378
trainning time: 159.8037450313568 seconds
```

为了方便观察，将训练过程中的 Loss 变化、验证集原始样本点和预测点利用 Matplotlib 绘图进行可视化如下：





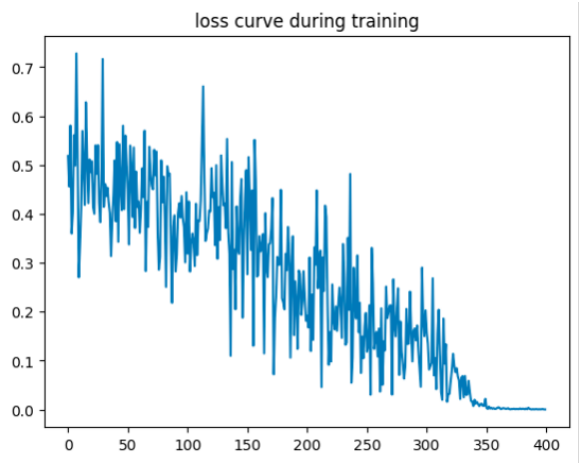
下面各种参数下的结果以最终 MSE 为指标，在 MSE 相近时训练的耗时（训练 MSE 大约收敛的 epoch 序号）也将作为参考。

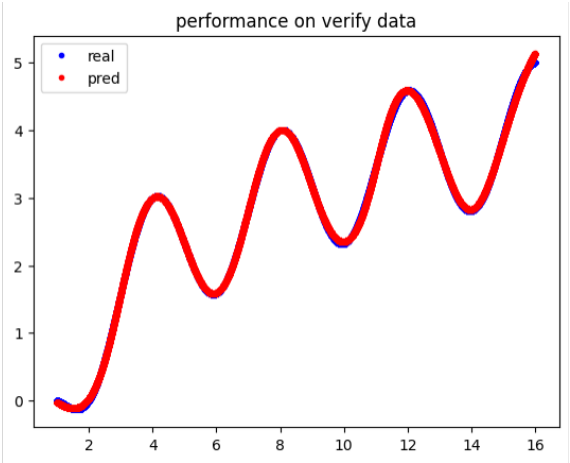
3.2 网络深度的影响

基础超参数具有两个全连接隐藏层，隐藏层的数量会影响训练的速度，可以尝试一个全连接隐藏层和三个全连接隐藏层的效果。

3.2.1 一层隐藏层的结果

```
Sequential(
  (0): Linear(in_features=1, out_features=64, bias=True)
  (1): Sigmoid()
  (2): Linear(in_features=64, out_features=64, bias=True)
  (3): Sigmoid()
  (4): Linear(in_features=64, out_features=1, bias=True)
)
MSE:
  average MSE of trainning is 0.251362
  minimum MSE of trainning is 0.000347
  MSE of verifying is 0.000627
trainning time: 138.47537803649902 seconds
```

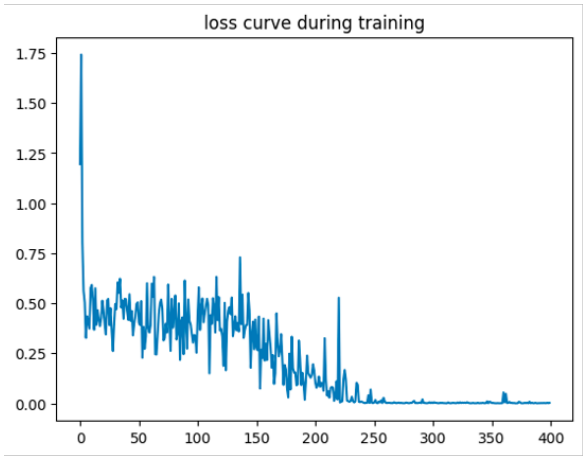


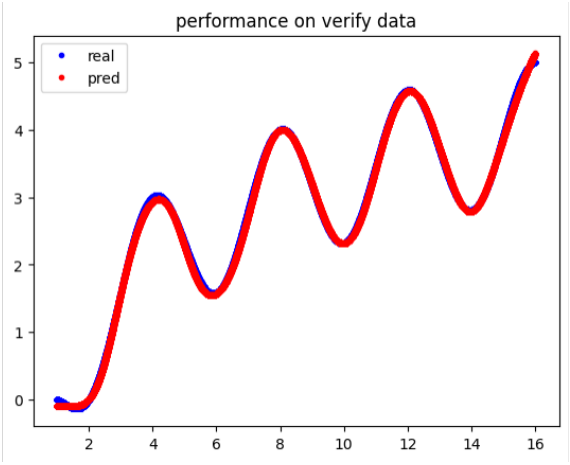


3.2.2 三层隐藏层的结果

```
Sequential(
  (0): Linear(in_features=1, out_features=64, bias=True)
  (1): Sigmoid()
  (2): Linear(in_features=64, out_features=64, bias=True)
  (3): Sigmoid()
  (4): Linear(in_features=64, out_features=64, bias=True)
  (5): Sigmoid()
  (6): Linear(in_features=64, out_features=64, bias=True)
  (7): Sigmoid()
  (8): Linear(in_features=64, out_features=1, bias=True)
)
MSE:
  average MSE of trainning is 0.200664
  minimum MSE of trainning is 0.000261
  MSE of verifying is 0.001651
```

```
trainning time: 180.10336899757385 seconds
```





3.2.3 网络层数的影响总结

MSE：二层 < 一层 < 三层，而且二层训练时间适中，依然选择二层隐藏层作为新的超参数选择。

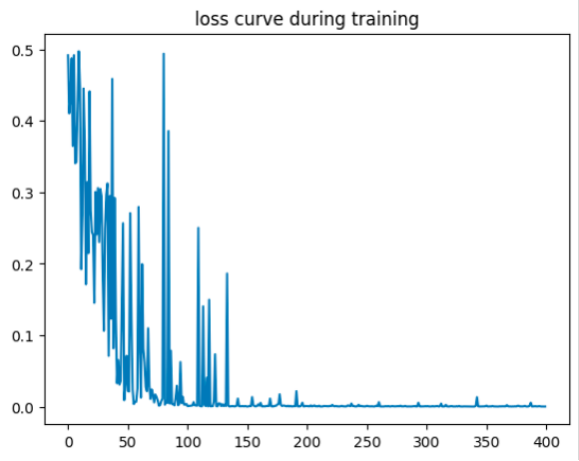
网络层数	测试集上的 MSE	训练时间（秒）	收敛轮数
一层	0.000627	138.47	350
二层（默认）	0.000378	159.80	300
三层	0.001651	180.10	250

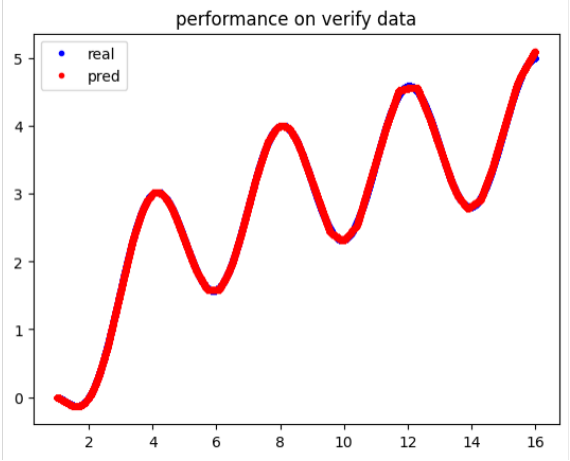
3.3 激活函数的影响

基础超参数使用的是 Sigmoid 作为激活函数，在此基础上，分别使用 ReLU 和 tanh 两个激活函数来尝试效果。

3.3.1 使用 ReLU 的结果

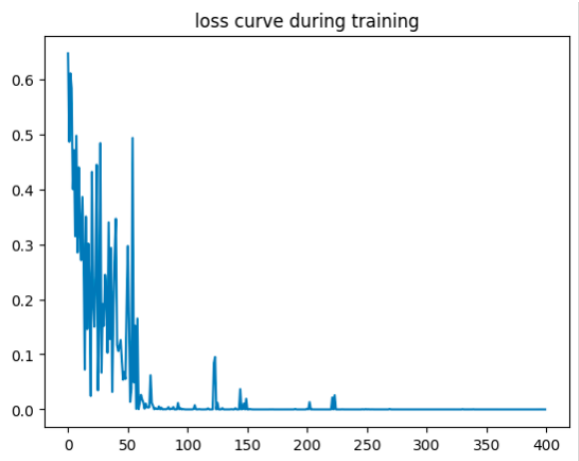
```
Sequential(
  (0): Linear(in_features=1, out_features=64, bias=True)
  (1): ReLU()
  (2): Linear(in_features=64, out_features=64, bias=True)
  (3): ReLU()
  (4): Linear(in_features=64, out_features=64, bias=True)
  (5): ReLU()
  (6): Linear(in_features=64, out_features=1, bias=True)
)
MSE:
  average MSE of trainning is 0.041999
  minimum MSE of trainning is 0.000037
  MSE of verifying is 0.000216
trainning time: 150.20449900627136 seconds
```

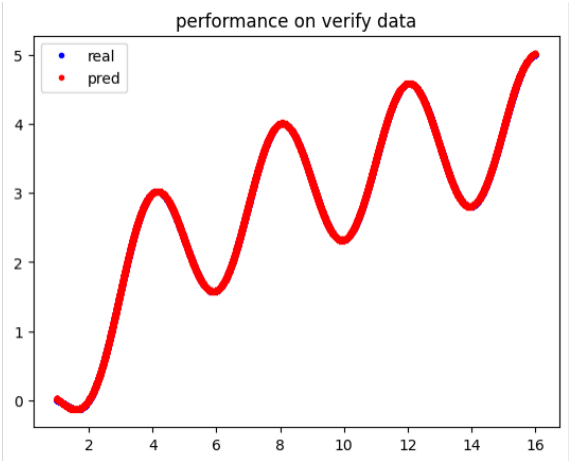




3.3.2 使用 tanh 的结果

```
Sequential(
  (0): Linear(in_features=1, out_features=64, bias=True)
  (1): Tanh()
  (2): Linear(in_features=64, out_features=64, bias=True)
  (3): Tanh()
  (4): Linear(in_features=64, out_features=64, bias=True)
  (5): Tanh()
  (6): Linear(in_features=64, out_features=1, bias=True)
)
MSE:
  average MSE of trainning is 0.036709
  minimum MSE of trainning is 0.000003
  MSE of verifying is 0.000029
trainning time: 162.52362275123596 seconds
```





3.3.3 激活函数的影响总结

MSE：Sigmoid > ReLU > Tanh。使用 Tanh 的效果非常好，收敛快，在区间边界也保持了良好的性质；而 Sigmoid 在边界处的性质稍有瑕疵，ReLU 小范围局部变化较大的位置稍有瑕疵。

选择 Tanh 作为新的超参数选择。

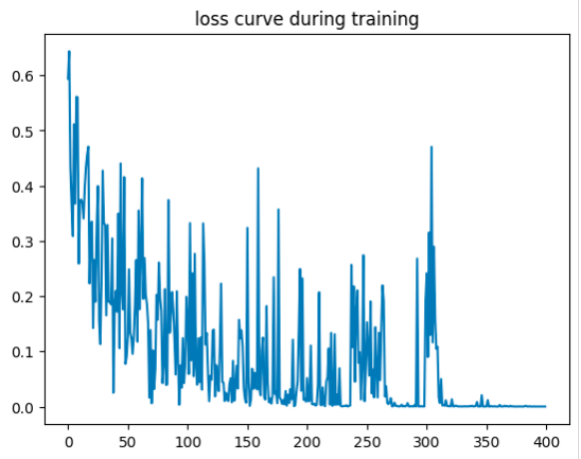
激活函数	测试集上的 MSE	训练时间（秒）	收敛轮数
ReLU	0.000216	150.20	150
Sigmoid（默认）	0.000378	159.80	300
Tanh	0.000029	162.52	80

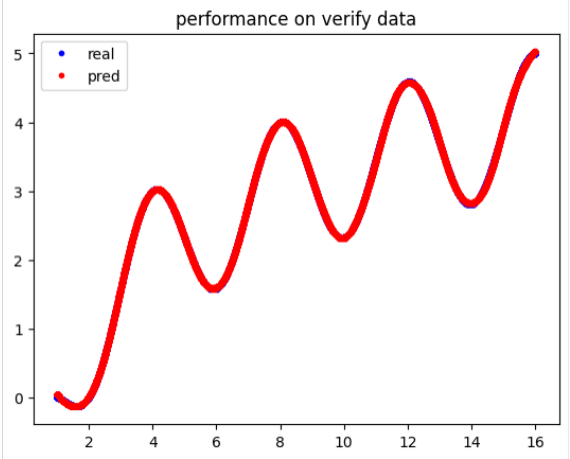
3.4 网络宽度的影响

基础超参数使用的网络宽度是 64，网络深度也会影响训练的速度。在新的超参数选择基础上，分别使用网络宽度 8、16、32、128 进行尝试。

3.4.1 网络宽度 8 的结果

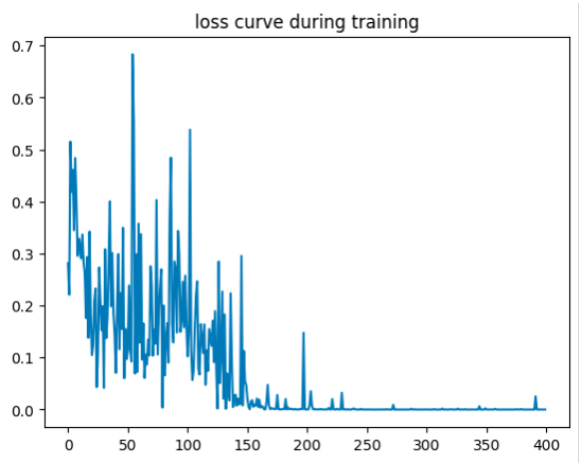
```
Sequential(
  (0): Linear(in_features=1, out_features=8, bias=True)
  (1): Tanh()
  (2): Linear(in_features=8, out_features=8, bias=True)
  (3): Tanh()
  (4): Linear(in_features=8, out_features=8, bias=True)
  (5): Tanh()
  (6): Linear(in_features=8, out_features=1, bias=True)
)
MSE:
  average MSE of trainning is 0.094547
  minimum MSE of trainning is 0.000046
  MSE of verifying is 0.000096
trainning time: 132.63091778755188 seconds
```

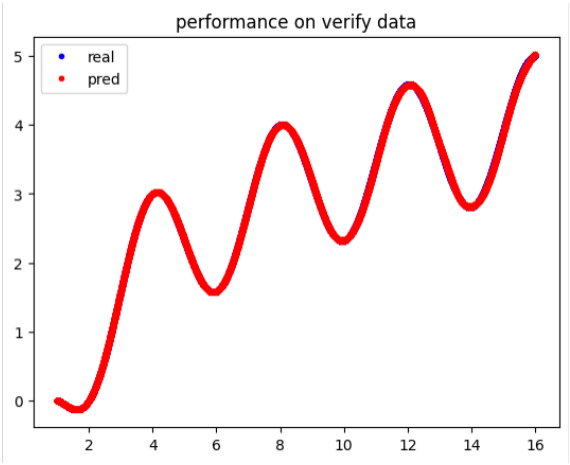




3.4.2 网络宽度 16 的结果

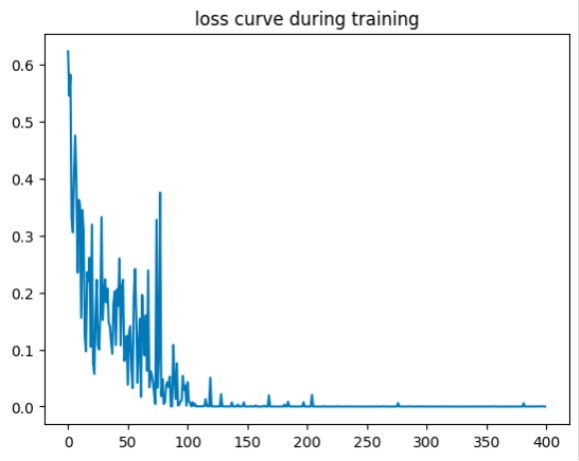
```
Sequential(
  (0): Linear(in_features=1, out_features=16, bias=True)
  (1): Tanh()
  (2): Linear(in_features=16, out_features=16, bias=True)
  (3): Tanh()
  (4): Linear(in_features=16, out_features=16, bias=True)
  (5): Tanh()
  (6): Linear(in_features=16, out_features=1, bias=True)
)
MSE:
  average MSE of trainning is 0.070662
  minimum MSE of trainning is 0.000013
  MSE of verifying is 0.000188
trainning time: 150.99685788154602 seconds
```

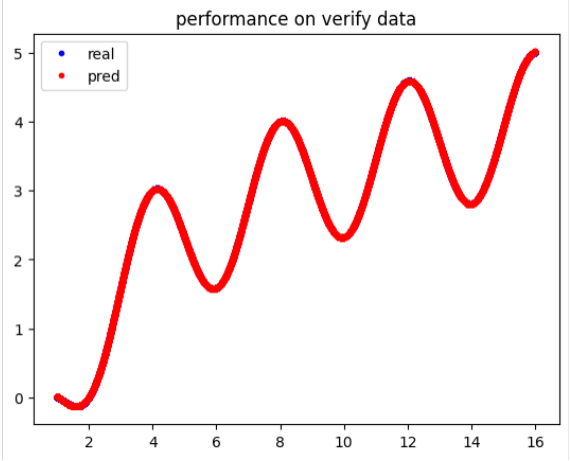




3.4.3 网络宽度 32 的结果

```
Sequential(
  (0): Linear(in_features=1, out_features=32, bias=True)
  (1): Tanh()
  (2): Linear(in_features=32, out_features=32, bias=True)
  (3): Tanh()
  (4): Linear(in_features=32, out_features=32, bias=True)
  (5): Tanh()
  (6): Linear(in_features=32, out_features=1, bias=True)
)
MSE:
  average MSE of trainning is 0.038991
  minimum MSE of trainning is 0.000005
  MSE of verifying is 0.000010
  MSE of testing is 0.000010
trainning time: 156.254380941391 seconds
```



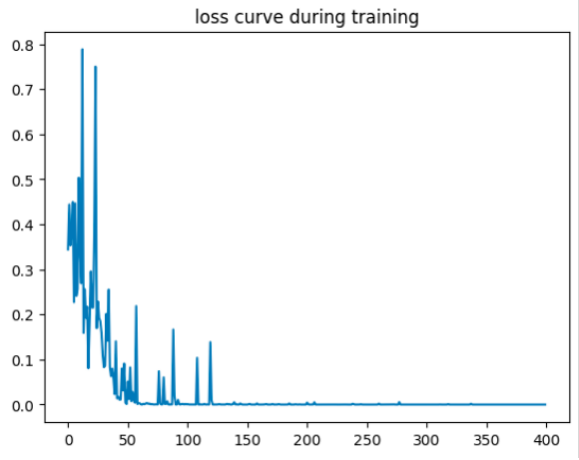


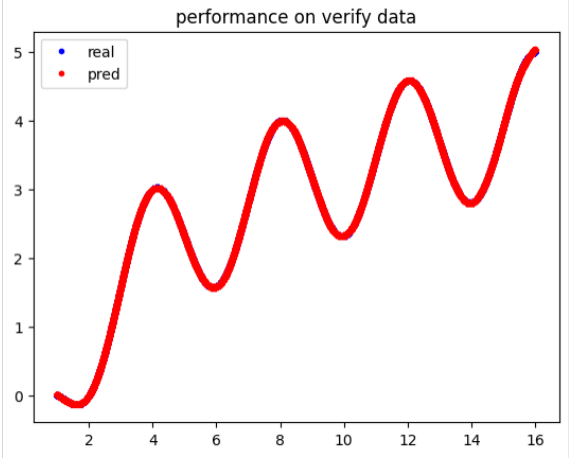
3.4.4 网络宽度 64 的结果

见 3.3.2 使用 tanh 的结果

3.4.5 网络宽度 128 的结果

```
Sequential(
  (0): Linear(in_features=1, out_features=128, bias=True)
  (1): Tanh()
  (2): Linear(in_features=128, out_features=128, bias=True)
  (3): Tanh()
  (4): Linear(in_features=128, out_features=128, bias=True)
  (5): Tanh()
  (6): Linear(in_features=128, out_features=1, bias=True)
)
MSE:
  average MSE of trainning is 0.029077
  minimum MSE of trainning is 0.000003
  MSE of verifying is 0.000023
trainning time: 201.7103133201599 seconds
```





3.4.6 网络宽度的影响总结

MSE 都表现很好。综合 MSE 和训练时间，我们将默认网络宽度调整至 32。

网络宽度	测试集上的 MSE	训练时间（秒）	收敛轮数
8	0.000096	132.63	350
16	0.000188	150.99	200
32	0.000010	156.25	150
64（默认）	0.000029	162.52	80
128	0.000023	201.71	100

3.5 学习率的影响

基础超参数学习率为 0.01，在此基础上尝试 0.1、0.05、05.005，主要关注收敛性和训练轮数的表现。

3.5.1 学习率为 0.01 的结果

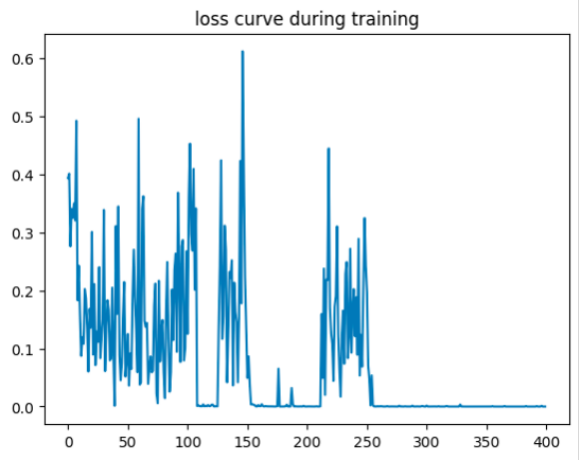
见 3.4.3 网络宽度 32 的结果

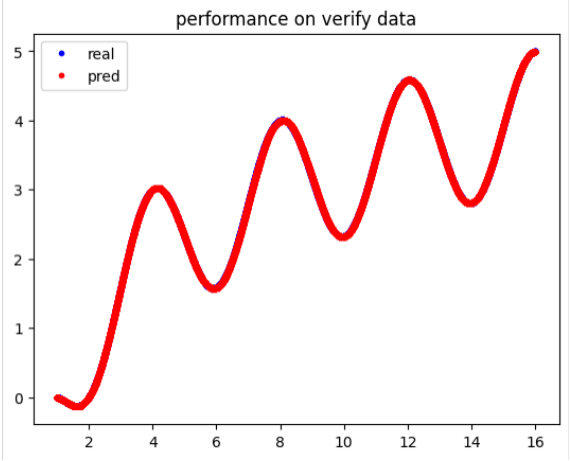
3.5.2 学习率为 0.1 的结果

这种情况下，可能因为学习率太大，无法收敛。

3.5.3 学习率为 0.05 的结果

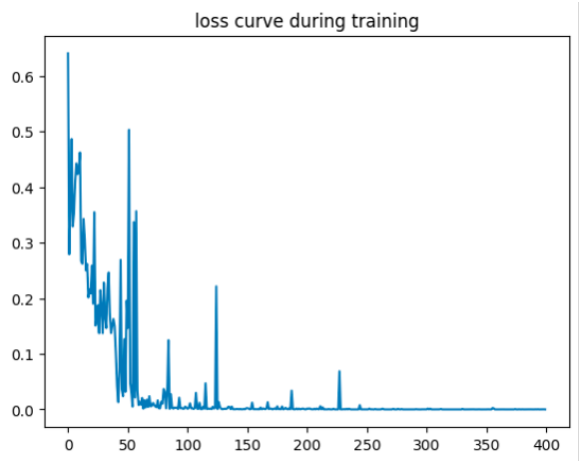
```
Sequential(
  (0): Linear(in_features=1, out_features=32, bias=True)
  (1): Tanh()
  (2): Linear(in_features=32, out_features=32, bias=True)
  (3): Tanh()
  (4): Linear(in_features=32, out_features=32, bias=True)
  (5): Tanh()
  (6): Linear(in_features=32, out_features=1, bias=True)
)
MSE:
  average MSE of trainning is 0.077823
  minimum MSE of trainning is 0.000007
  MSE of verifying is 0.000067
trainning time: 154.54519200325012 seconds
```

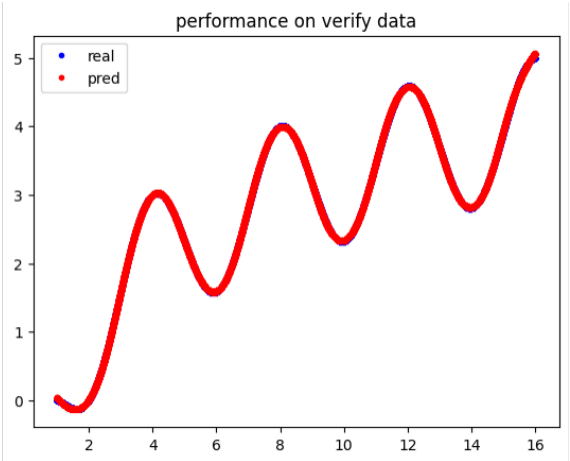




3.5.4 学习率为 0.005 的结果

```
Sequential(
  (0): Linear(in_features=1, out_features=32, bias=True)
  (1): Tanh()
  (2): Linear(in_features=32, out_features=32, bias=True)
  (3): Tanh()
  (4): Linear(in_features=32, out_features=32, bias=True)
  (5): Tanh()
  (6): Linear(in_features=32, out_features=1, bias=True)
)
MSE:
  average MSE of trainning is 0.035650
  minimum MSE of trainning is 0.000025
  MSE of verifying is 0.000088
trainning time: 153.22086024284363 seconds
```





3.5.5 学习率的影响总结

学习率的影响总结如下，从 0.05 的情况可以看出学习率过大会导致 Loss 震荡。可见，学习率 $lr = 0.01$ 是最合适的。

学习率	测试集上的 MSE	是否收敛	收敛轮数	训练时间（秒）
0.1	—	否	—	—
0.05	0.000067	是	275	154.54
0.01（默认）	0.000010	是	150	156.25
0.005	0.000088	是	150	153.22

4 最终参数及测试

经过调参，最终选定的参数如下：

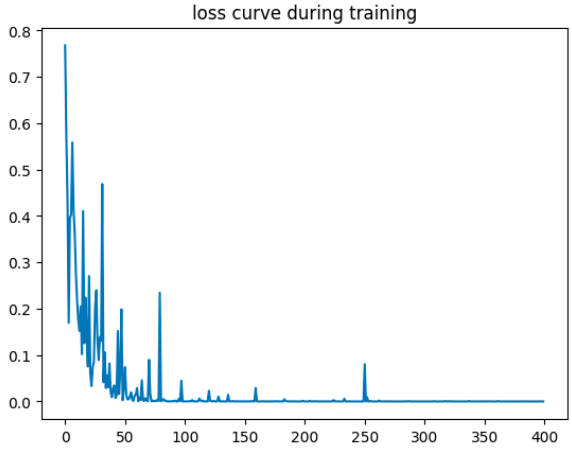
网络结构：

```
Sequential(
  (0): Linear(in_features=1, out_features=32, bias=True)
  (1): Tanh()
  (2): Linear(in_features=32, out_features=32, bias=True)
  (3): Tanh()
  (4): Linear(in_features=32, out_features=32, bias=True)
  (5): Tanh()
  (6): Linear(in_features=32, out_features=1, bias=True)
)
```

其他超参数：

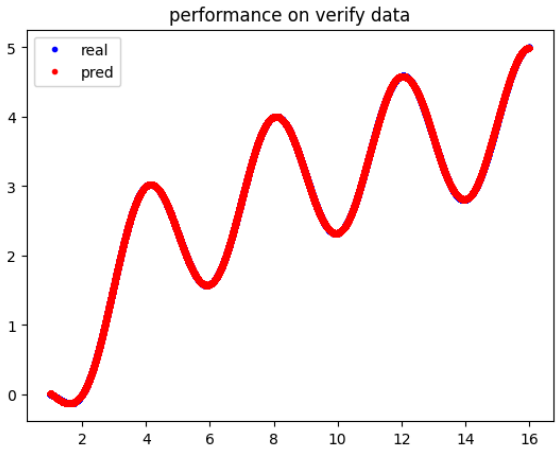
```
epochs=400, batches=32, lr=0.01
dataset cases:
  train set: 50000 cases
  verify set: 25000 cases
  test set: 15000 cases
```

训练过程 Loss 的变化：



验证集上的结果：

```
MSE：
  MSE of verifying is 0.000022
```



测试集上的结果：

```
MSE：
  MSE of testing is 0.000022
```