lab1 实验报告

- 刘兆宸
- PB21061373

实验目的

使用 pytorch 完成一个简单的DNN,近似函数$y = log_2(x) + \cos\left(\frac{\pi x}{2}\right)$,研究数据量、网络深度、学习率、网络宽度、激活函数对模型性能的影响.

关键代码展示

```
class my_net(nn.Module):
    def __init__(self, input_size: int, hidden_size: int,
output_size: int, hidden_depth: int = 1):
        super(my_net, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.fc2 = nn.Linear(hidden_size, output_size)
        self.fc_mid = nn.Linear(hidden_size, hidden_size)
        self.hidden_layer_depth = hidden_depth
        self.active_func = nn.LeakyReLU()

    def forward(self, x):
        out = self.fc1(x)
        out = self.active_func(out)

        for _ in range(self.hidden_layer_depth):
            out = self.fc_mid(out)
            out = self.active_func(out)

        out = self.fc2(out)
        out = self.active_func(out)
        return out
```

```
model  = my_net(input_size, hidden_size, output_size)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(),
lr=learning_rate)
# Train the model
for epoch in range(num_epochs):
    # Forward pass
    outputs = model(train_x)
    loss = criterion(outputs, train_y)
    # Backward and optimize
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

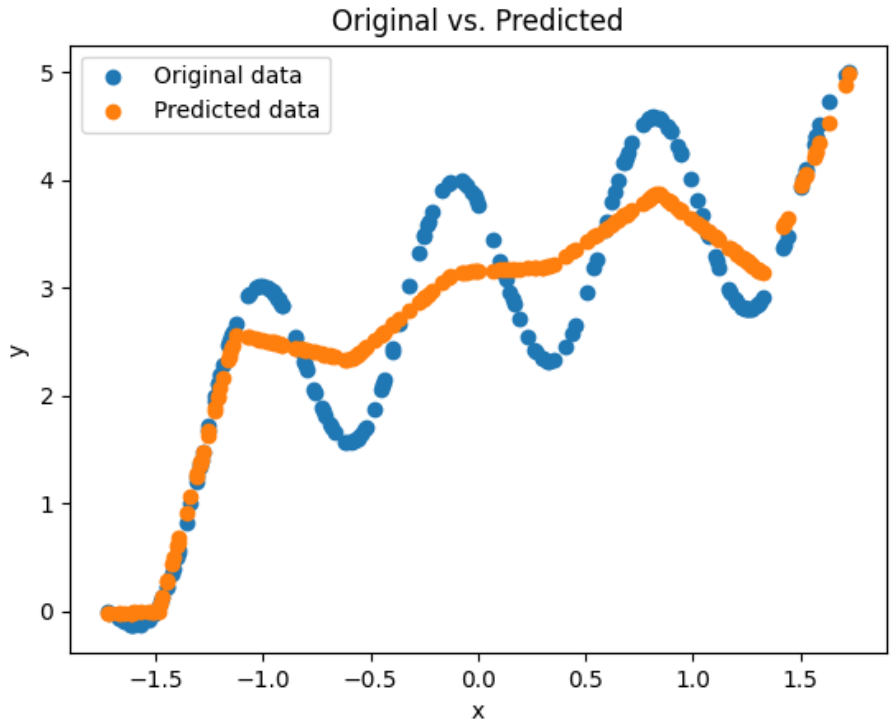
实验过程

先是设计两层的线性层的一个简单的神经网络(输入层、隐藏层、输出层)，然后通过不同的参数进行训练，初始参数设计结果如下。

> *epoches=200,loss选择MSELoss(), 注意由于为了防止激活函数失效，将x标准化,i.e.* $x = \frac{x-x.mean}{x.std}$.

参数	N	h_1	学习率	激活函数
值	2000	500	0.01	LeakyRelu

验证集上结果如下:

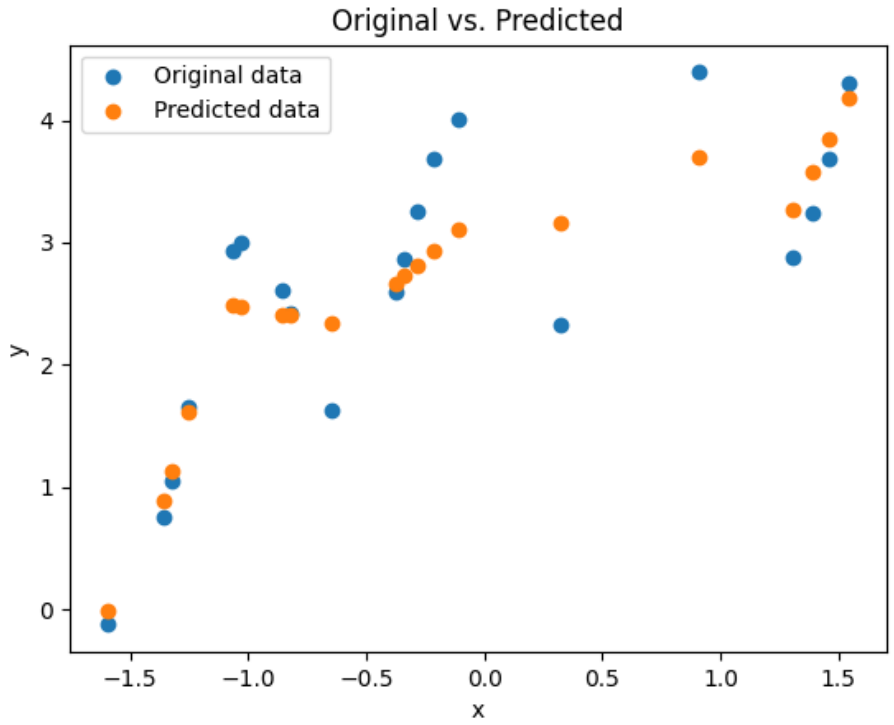


此时$MSE(=\frac{1}{n}\sum(y_{pred}-y)^2)=0.0.2111246$

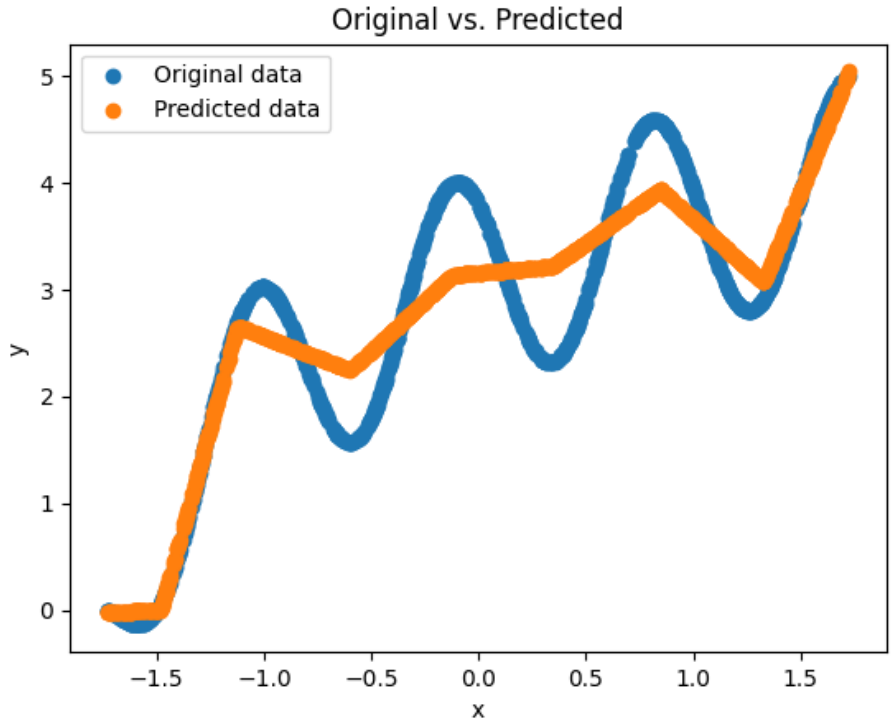
为了研究不同超参数对模型的影响，我们控制变量调整数据量N、网络深度、学习率、网络宽度、激活函数，分别进行实验。

一. 调整数据量N

保持网络结构及学习率不变我们得到结果如下:



N = 200,MSE= 0.20631718635559082

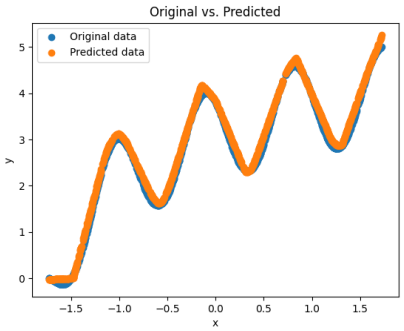


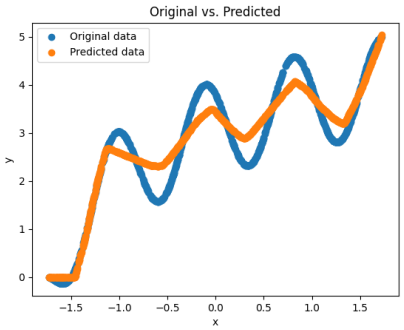
N = 10000, MSE = 0.20180949568748474

对比可知N=10000时，这种参数结构表现最好,因为数据量越大，模型越容易拟合数据，且由于网络参数够多，并没有过拟合

二. 调整网络深度

我们调整网络深度为3层,6层,结果分别如下(左 h=3 ,右 h=6):
(中间层参数为150x150)





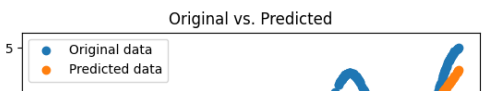
h=3, MSE: 0.40837058424949646

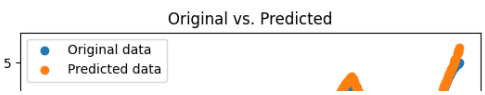
h=6结构,MSE: 0.1355595886707306

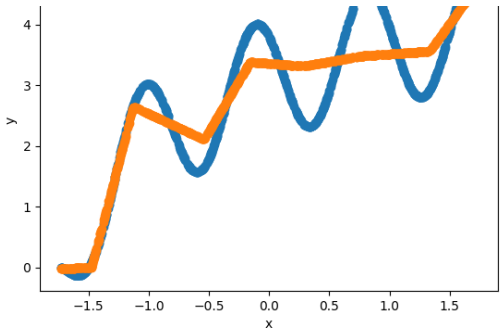
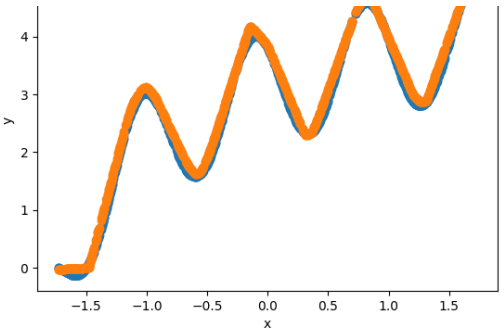
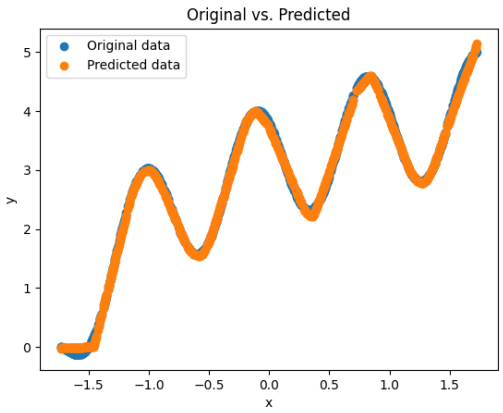
对比初始发现h=3层时拟合效果较好，相较于2层时可能结构更复杂更容易拟合该曲线，但h=6层时拟合效果反而变差,可能是因为过拟合。
因此我们下面实验就用3层结构进行实验.

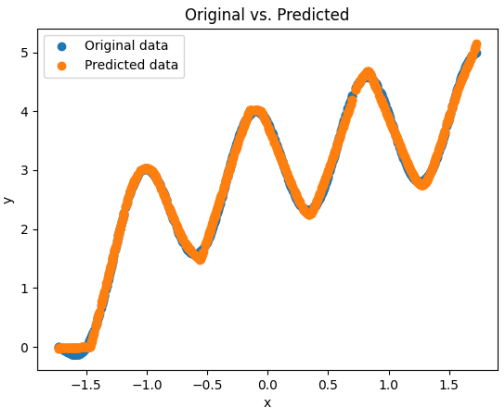
三. 调整网络宽度

以三层为例,1xw, wxw, wx1的结构为例,我们分别尝试了w=50,w=100,w=250,w=500,结果如下:
(左w=50,右w=250)





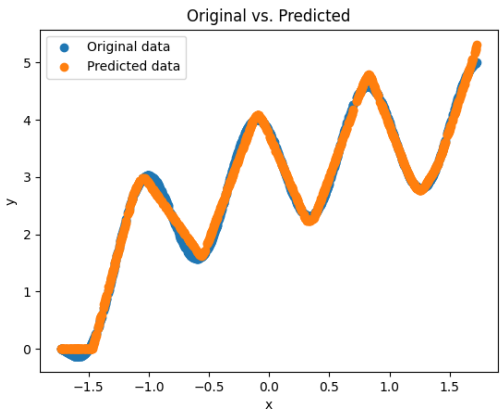


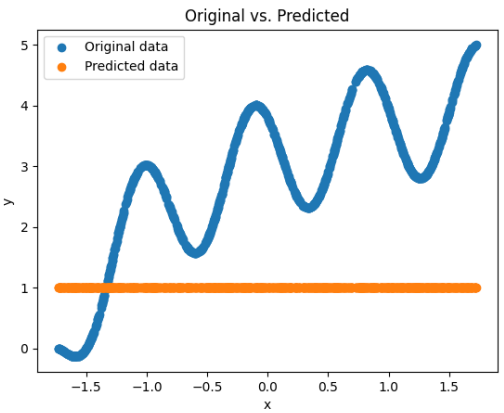
- w50 : MSE: 0.2593955099582672
- w150: 0.011002691462635994
- w250: MSE: 0.0032969526946544647
- w500 MSE: 0.0026266041677445173

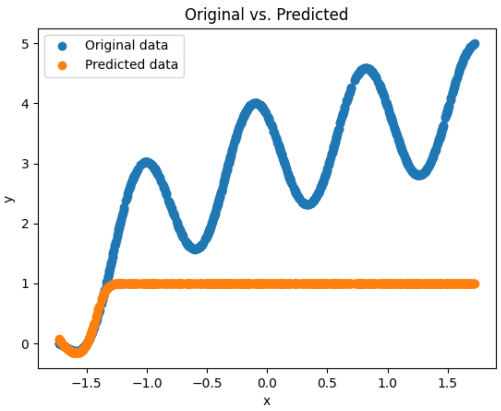
由上可见，网络的宽度越大，拟合效果越好，但是w=250时已经可以达到很好的效果,再大可能过拟合了,因此我们选择w=250作为下面的实验参数.

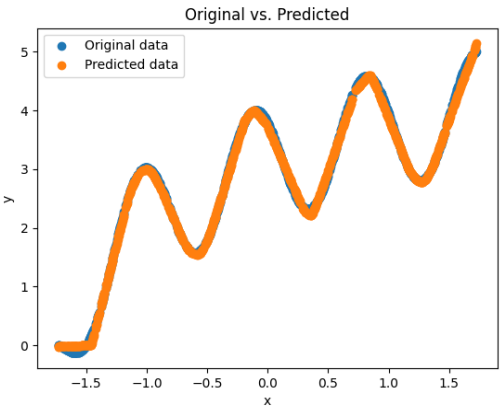
四. 调整激活函数

我们分别尝试了Sigmoid,Tanh,Relu,LeakyReLU激活函数,结果如下:
(以1x250,250x250,250x1的结构为例)









Relu : 0.00852164812386036

sigmoid: 4.795731544494629,由于不是分类任务,不适合用sigmoid

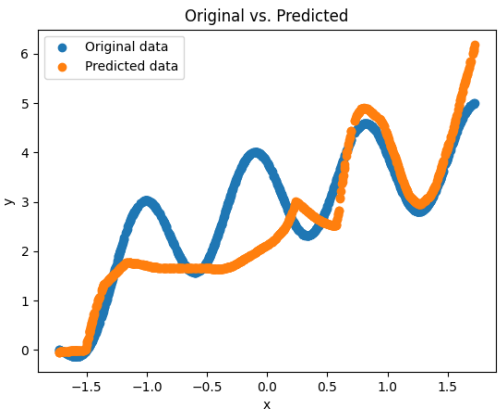
tanh: 0.00852164812386036, $tanh(x) = 2sigmoid(x) - 1$,和sigmoid函数出现了类似的问题.

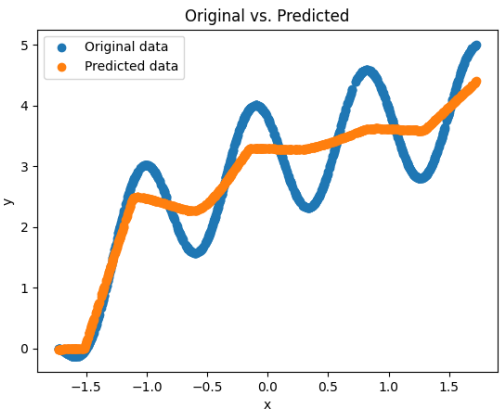
LeakyRelu: 0.0032969526946544647,相比较Relu,LeakyRelu的效果更好，图像显得更加平滑,不会出现"死亡神经元",效果较好

五. 调整学习率

我们分别尝试了`0.1`,`0.01`,`0.001`的学习率(上面的结果均为lr=0.01),结果如下:

(以1x150,150x150,150x1的结构为例)



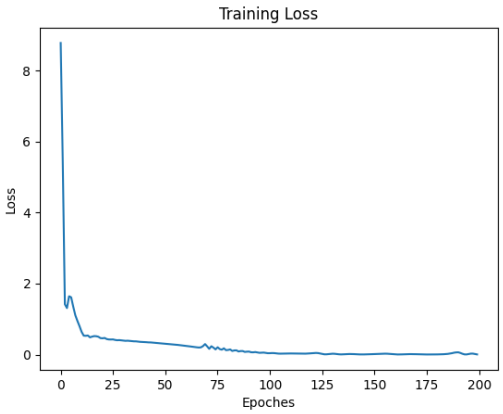


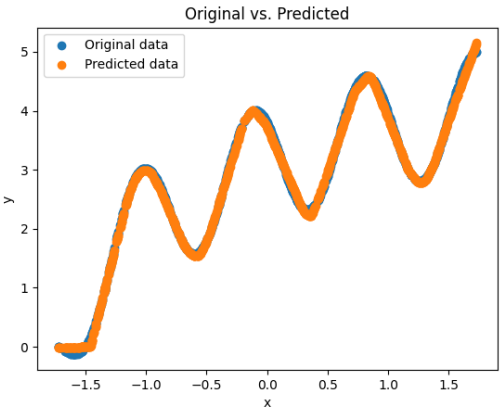
测试集表现

参数	N	学习率	激活函数
值	2000	0.01	LeakyRelu

网络结构 : 1x250,250x250,250x1 的三层结构

Loss和拟合表现如下





test set MSE: 0.003418794833123684