函数拟合实验报告

高宽 2252708

2025 年 3 月 6 日

目录

第 1 部分　代码作用

使用一个双层 ReLU 网络拟合一个基本初等函数，从而验证通用近似定理。其中，网络搭建只使用 numpy，利用 numpy 实现自动梯度。这次任务非常简单，读了文档的原理，看看代码，很容易完成。在进行实验的过程中，我还深入推理了 softmax 函数的反向传播公式，对反向获得了更加深刻的理解。

第 2 部分　函数定义

待拟合的目标函数为:

$$f(x) = \frac{\sin(x) + \cos(x)}{\exp(x)} - x^4 \tag{1}$$

第 3 部分　数据采集

使用 numpy 即可快速生成数据。代码如下：

```
x = np.linspace(-np.pi, np.pi, 1000).reshape(-1, 1)
y = (np.sin(x) + np.cos(x)) / np.exp(x) - x ** 4
```

第 4 部分　模型描述

本实验采用双层 ReLU 网络，输入和输出层维度均为 1，隐藏层维度可以自定义。图 4-1 为网络架构图。

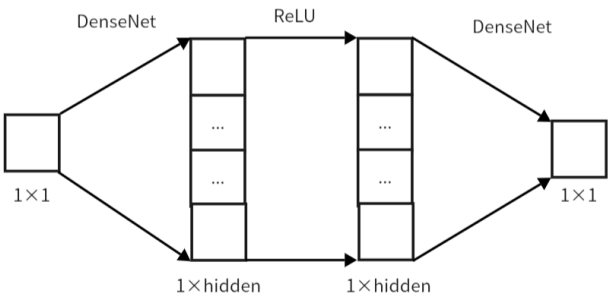

图 4-1：网络架构图

实现该模型的过程中，需要自行定义正向传播和反向传播的计算函数。下面给出其推导的原理。

对于神经网络中的任意一次计算或变形操作，均有输入和输出。设函数为 f，输入为 x，输出为 h，待求偏导的变量为 w。w 为向量或矩阵。在一次前向传播后，为了更新参数，我们需要求出损失函数对参数的梯度: $\frac{\phi L}{\phi w}$，其形状于 w 相同。根据矩阵分析，有:

$$\frac{\phi L}{\phi w} = \frac{\phi L}{\phi h}\frac{\dot{\phi h}}{\phi w}$$

这里的乘法是不一定矩阵乘法。在计算中，我们会根据具体情况，在确保遵守链式法则的前提下，使用尽可能简洁地方法将 $\frac{\phi L}{\phi h}$ 与 $\frac{\phi h}{\phi w}$ 结合算出 $\frac{\phi L}{\phi w}$ 算出。$\frac{\phi L}{\phi h}$ 对于中间层来说，可以直接从后一层的反向传播中获得，对于最后一层来说，可以直接根据求导法则和损失函数，利用预测值与真实值求出，因此 $\frac{\phi L}{\phi h}$，总是可以当作反向传播函数的参数传入。$\frac{\phi L}{\phi w}$ 则可以根据矩阵分析中学过的求导法则求出。综上，在一次训练中，我们总是可以通过反向传播计算出损失函函数对任意参数的梯度! 以乘法计算为例，其代码为：

```
class Mul():
def __init__(self):
    self.mem = {}

def forward(self, inp, w):
    # inp.shape: (N, num_features)
    # w.shape: (in_dim, out_dim)
    # outp.shape: (N, out_dim)
    self.mem['inp'] = inp
    self.mem['w'] = w
    # print(inp.shape, w.shape)
    outp = inp @ w
    return outp
```

```
14
15 def backward(self, grad_outp):
16     # grad_outp.shape: (N, out_dim)
17     # grad_inp.shape: (N, num_features)
18     # grad_w.shape: (in_dim, out_dim)
19     grad_inp = np.matmul(grad_outp, self.mem['w'].T)
20     grad_w = np.matmul(self.mem['inp'].T, grad_outp)
21     return grad_inp, grad_w
```

ReLU 操作的正反向传播与之类似。

我们利用上述原理即可实现基于 numpy 的自动梯度。

第 5 部分　拟合效果

训练时使用 MES 作为拟合的损失函数。由于待拟合的函数比较简单，样本点也比较少，所以训练时不分批次，每一轮都直接拿整个数据集做训练（也可以认为批大小为 1000）。经过多次实验，我们选取隐藏层维度为 400、学习率为 0.0000005、训练 30000 轮的经验值。图 5-1 训练所得结果。

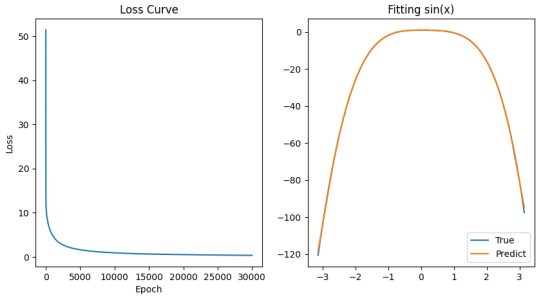

图 5-1：拟合结果