华中科技大学

计算机视觉实验报告

专　　业：	计算机科学与技术
班　　级：	CS2005 班
学　　号：	U202090063
姓　　名：	董玲晶
指导教师：	刘　康

分数	
教师签名	

2023 年 3 月 21 日

1 任务要求

设计一个前馈神经网络，对一组数据实现回归任务。

在[-10,10]×[-10,10]的 2-D 平面内，以均匀分布随机生成 5000 个数据点(x,y). 令 f(x,y) = x^2 +xy+y^2。至少含有一层隐藏层的前馈神经网络以预测给定数据点(x,y)的 f(x,y)函数值。在随机生成的数据点中，随机抽取 90%用于训练，剩下的 10%用于测试。

框架任选，尝试不同的网络层数、不同的神经元个数、使用不同的激活函数等，观察网络性能。

2 任务设计

2.1 数据集准备

使用 numpy.random.uniform() 函数均匀分布随机生成 5000 个点，范围为 (-10, 10)，size 为 (5000, 2)，即 5000 个 2D 平面上的点。得到点后计算对于所有点 (x, y) 的 f(x,y) = x^2 +xy+y^2 值。使用 sklearn.model_delection 包下的 train_test_split()函数，将生成的点和对应的函数值按 9: 1 分为训练集和测试集。

对划分好两个数据集中的元素按元组（数据，标签）的格式重整，以便后续使用 dataloader。

将上述整个过程抽象为函数 data_generation(num of data, size, scales, test_ratio)，对应代码如下：

代码 1 数据集生成 data_generation()

```
def data_generator(num, size, scales, test_size):
  scale_down, scale_up = scales
  x_y = np.random.uniform(scale_down, scale_up, (num, size))
  f_x_y = x_y[:,0]**2 + x_y[:,0]*x_y[:,1] + x_y[:,1]**2
  f_x_y = f_x_y.reshape(-1, 1)
   xy_train, xy_test, fxy_train, fxy_test = train_test_split(x_y, f_x_y,
test_size=test_size)
  train_dataset = []
  test_dataset = []
  for idx in range(xy_train.shape[0]):
    train_dataset.append((xy_train[idx], fxy_train[idx]))
  for idx in range(xy_test.shape[0]):
    test_dataset.append((xy_test[idx], fxy_test[idx]))
  return train_dataset, test_dataset
```

2.2 网络结构

由于是点的回归任务，INPUT_SIZE 为 2，OUTPUT_SIZE 为 1；输入的变量还有隐藏层神经元个数 HIDDEN_SIZE；用 toggle 来控制神经网络的层数，默认值为 1；激活函数参数默认值为 nn.ReLU()。

采用全连接层+激活函数结构，神经网络的结构图如下所示：

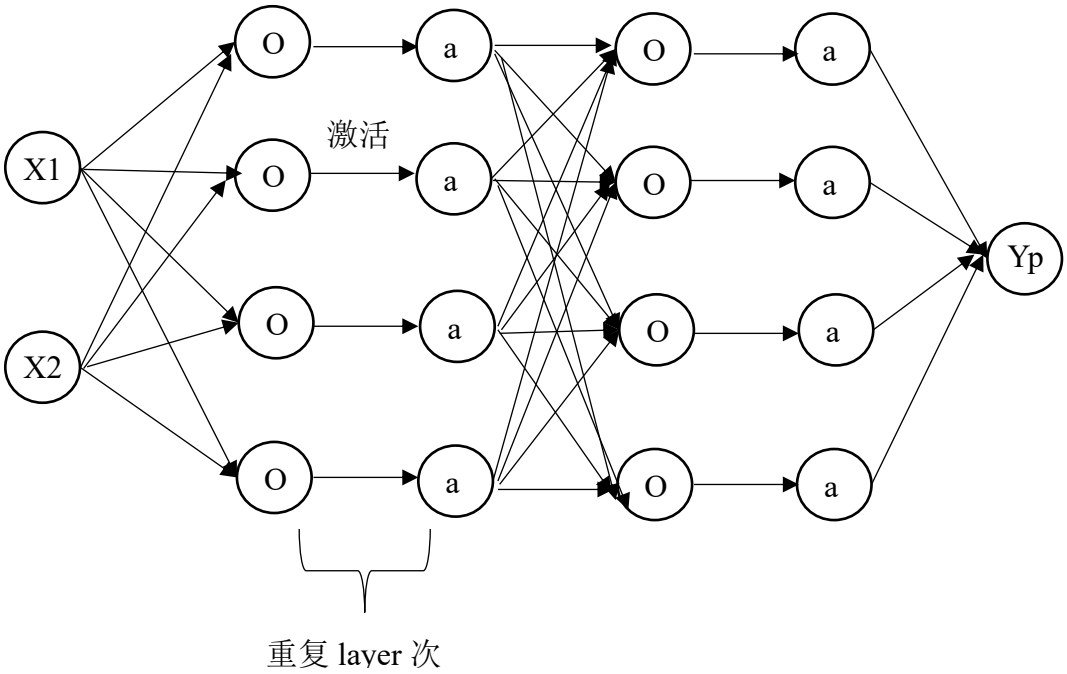

图 1 神经网络结构

2.3 架构及固定参数

总体采用 pytorch 架构，训练、测试、绘图、网络等全体代码详情见开源仓库：https://github.com/Elubrazione/cv_labs_hust/tree/main/lab1。

2.3.1 优化器

选用 torch.optim.Adam()，其是随机梯度下降（SGD）算法的一种变体，具有自适应学习率和动量参数，更改初始学习率为 0.002

2.3.2 损失函数

选用 nn.MSELoss()，其是回归问题中最常用的损失函数之一，计算简单；具有连续性和光滑性，求解过程相对稳定。

2.3.3 其它参数

BATCH_SIZE = 20， EPOCH = 50；

3 实验

本实验分为两大部分，意在探究不同的网络层数、神经元个数、激活函数对神经网络性能的影响。观察在不同的参数下随着训练次数的增加，loss 的变化趋势及训练时间的长短。

3.1 激活函数

本次实验一共选择了三种不同的激活函数，分别为 ReLU、Sigmoid 和 Tanh，在隐藏层数和神经元个数相同的情况下进行实验，改变隐藏层数和神经元个数的组合进行多次实验，每组实验及对应参数如下表所示：

表 1 探究激活函数对神经网络性能的影响

实验编号	隐藏层数目	神经元个数	激活函数
0	1	16	ReLU
			Sigmoid
			Tanh
1	2	8	ReLU
			Sigmoid
			Tanh
2	4	4	ReLU
			Sigmoid
			Tanh

通过实验可以发现在神经网络结构较为简单时，ReLU 可以让模型的损失函数快速收敛；但相对来说 Sigmoid 和 Tanh 的效果则较差，在训练中梯度过小，导致收敛的速度更慢，如图 2 所示。与此同时，实验表明 Tanh 的效果更优于 Sigmoid。

分析可得，上述原因是用激活函数本身的性质造成的。虽然三者都为非线性函数，但 ReLU 较为简单，输出为 MAX(0, X)，梯度为为 0 或 1，尽管它也有其它缺点，但其所具有的非饱和性和稀疏性可以有效避免梯度消失的问题，且计算更加高效和快速；Sigmoid 函数的输出为(0, 1)，两端的输出值较密集，因此对于很大或者很小的值其输出值都很接近 1 或 0，梯度会特别小，从而导致了模型参数更新十分缓慢；与 Sigmoid 相比，Tanh 的输出范围更大，为[-1, 1]，且两端的输出饱和度相较于 Sigmoid 更低一些，因此在梯度消失的问题上性能由于

Sigmoid。

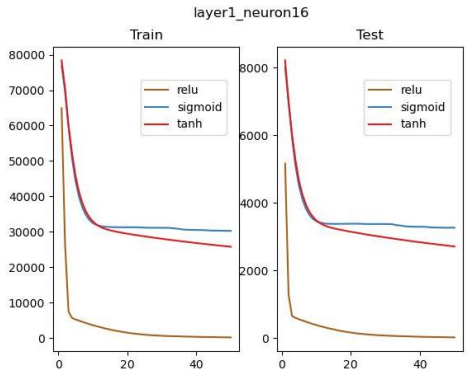

图 2-a 不同激活函数对 loss 的影响(1)

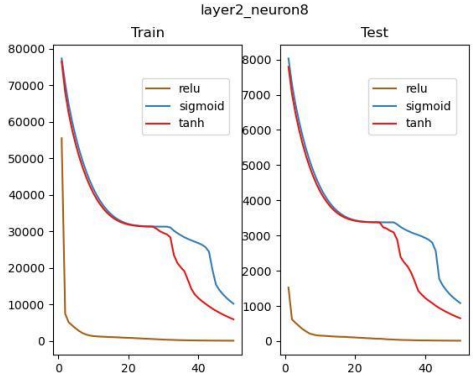

图 2-b 不同激活函数对 loss 的影响(2)

通过实验结果也可以发现，在参数合适的情况下，训练集和测试集的损失都随着训练次数的增加不断下降，直到拟合。

3.2 神经元个数

控制神经网络的非线性层层数和激活函数不变，改变每层的神经元个数。如图 3-a 和 3-b，可以看到在还未收敛时，增加神经元个数有助于加速损失下降，使模型更快达到收敛。

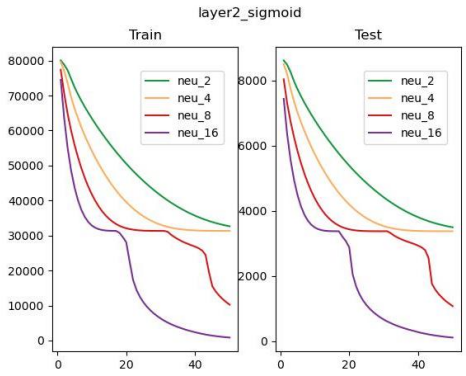

图 3-a 神经元个数对 loss 的影响(1)

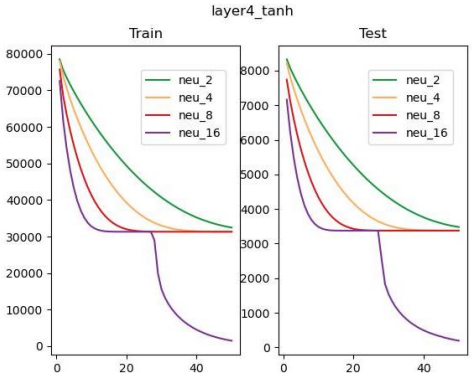

图 3-b 神经元个数函数对 loss 的影响(2)

而当神经元个数继续增加时，整体网络的结构也会趋于复杂。如图 4-a，可以观察到在层数为 4 的网络中，每层神经元的个数达到 16 时，尽管训练集的损失仍在下降，测试集的损失已经出现了轻微的震荡；而随着神经元个数的增加，训练集的损失也出现了震荡，且个数越多，震荡似乎更明显且密集，如图 4-b。

同时在实验的过程中，可以感受到随着神经元的个数增加，训练的时间也越来越长。因此在简单的任务中，过度神经元个数会增加模型的复杂度，使训练的时间增加，可以导致模型拟合化。

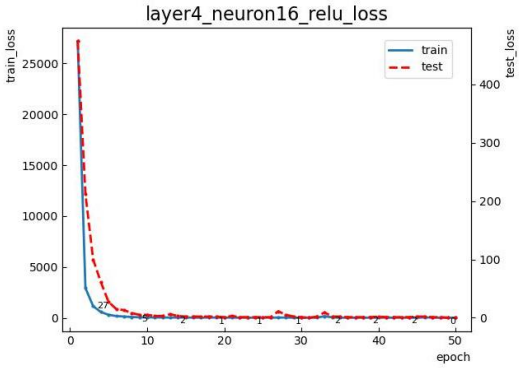

图 4-a 增加神经元个数后的震荡现象(1)

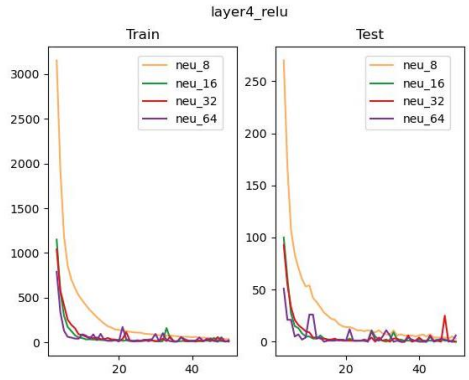

图 4-b 增加神经元个数后的震荡现象(2)

3.3 隐藏层层数

控制神经网络隐藏层的神经元个数和激活函数不变，改变隐藏层的层数。如图 5-b，对于八个神经元个数、使用 Sigmoid 激活函数的神经网络结构，一层的效果过差，随着迭代次数增加，梯度消失、模型收敛过慢甚至可能难以收敛，因此适当增加隐藏层的层数可以增加神经网络的性能。综合 5-a 可以观察到，层数的增加可以使损失下降的速度更快，从而加速模型的收敛，这样就使学习的效率大大提升。

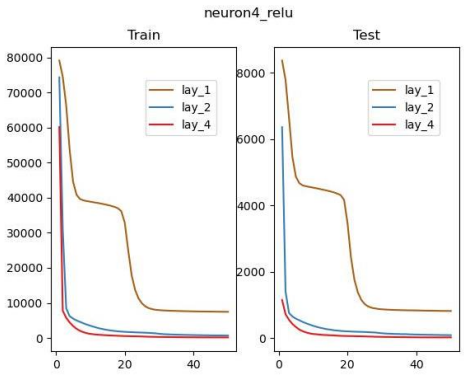

图 5-a 隐藏层的层数对 loss 的影响(1)

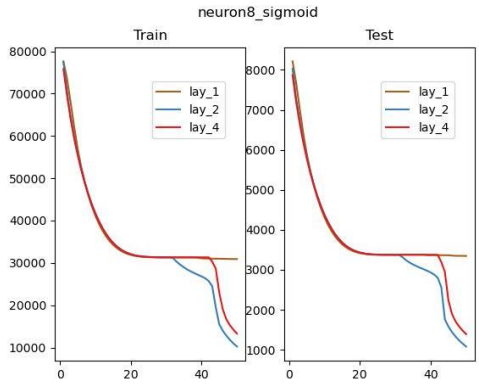

图 2-b 隐藏层的层数对 loss 的影响(2)

但同神经元个数对网络性能的影响相似，仍需考虑的另一个问题是，无限增加层数对网络的性能是否也会产生不好的影响。设计了如下实验，使用特别少的神经元个数，排除神经元个数对模型复杂度的影响，使用 50 层的层数进行实验，可以观察到测试集的损失也出现了振荡现象，如图 6-a 所示。这说明层数的增加给模型带来的影响确实如我们自然推导的那样：会增加模型的复杂度从而可能产生过拟合的现象。

为了放大这种现象从而进行更好的观察，将神经元的个数增加至 16，改变层

数，如图 6-b 所示。随着层数的增加，先是只有测试集上出现了损失震荡，而后训练集上出现了此种现象，且层数越大震荡幅度呈上升趋势。这也说明了层数的增加对模型复杂度的影响可能会导致梯度的爆炸，使训练更加困难。

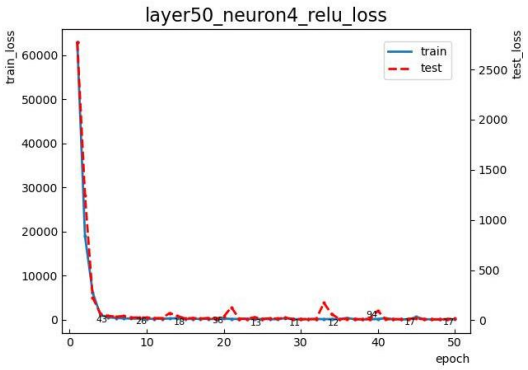

图 6-a 增加隐藏层数后的振荡现象(1)

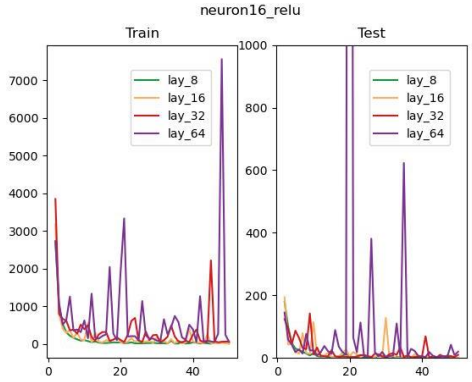

图 6-b 增加隐藏层数后的振荡现象(2

4 结论

激活函数、神经元个数、隐藏层层数都会对神经网络的性能造成影响。随着神经元个数和隐藏层层数的增加，模型的收敛速度和学习效率会随之增加，但若过于复杂则会出现模型过拟合的现象。因此在模型搭建时，除了选择合适的学习率和 BATCH_SIZE 大小，激活函数和神经网络层数、每层神经元个数的选择也尤为重要。