电类工程学导论 C 实验报告 13

518030910406 郑思榕

1 实验准备

1.1 实验环境介绍

1. 环境：在 Windows 系统中进行本次实验
2. 语言：python 3.7
3. 工具：本实验使用了开源的深度学习框架 pytorch

1.2 实验目的

1. 利用深度学习实现对非线性函数 $f(x) = x^2 + 2sin(x) + cos(x-1) - 5$ 的拟合，进行以下三个练习：
   (a) 不对代码其他部分进行改动，更改参数 NUM_TRAIN_EPOCHS 为 100,1000,10000,50000，你发现拟合得到的曲线有什么变化?
   (b) 固定 NUM_TRAIN_EPOCHS=1000，更改参数 LEARNING_RATE 为 1,0.1,0.01,0.001，你发现拟合得到的曲线有什么变化?
   (c) 自定义一个函数 f(x)，调整合适的参数，使得模型拟合效果尽可能好。
2. 利用深度学习对 CIFAR-10 图片进行分类，完成下面几个练习。
   (a) 补充 exp2.py 的代码，使得程序可以完整运行。
   (b) 使用 resnet20 模型，训练一个 cifar-10 的分类器。(推荐训练策略：以 0.1 的学习率 (learning rate, lr) 训练 5 个 epoch，再以 0.01 的 lr 训练 5 个 epoch。)
   (c) (可选) 思考：Train acc 和 Test acc 有什么关联和不同? 在 lr 从 0.1 变到 0.01 后，acc 发生了什么变化？为什么?

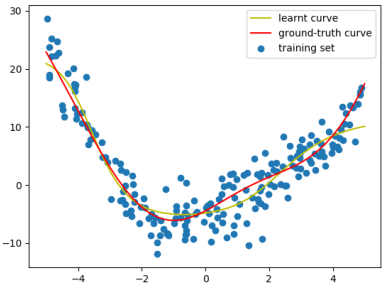
1.3 实验原理

本实验以神经网络的工作模型为实验原理，每层神经网络根据上一层的损失 loss 更改神经网络的参数，经过多层网络的计算得到较好的拟合模型。

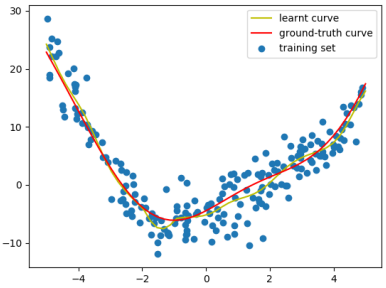
2 实验过程

2.1 实验 1：初等函数拟合

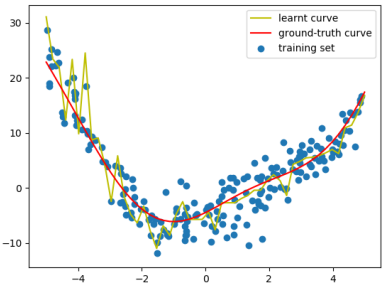
2.1.1 练习 A: 改变参数 NUM_TRAIN_EPOCHS



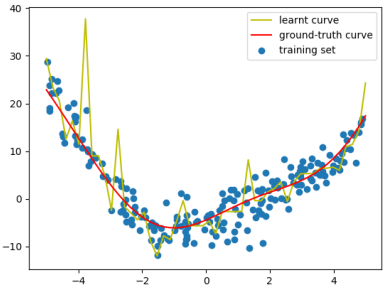
NUM_TRAIN_EPOCHS=100



NUM_TRAIN_EPOCHS=1000



NUM_TRAIN_EPOCHS=10000



NUM_TRAIN_EPOCHS=50000

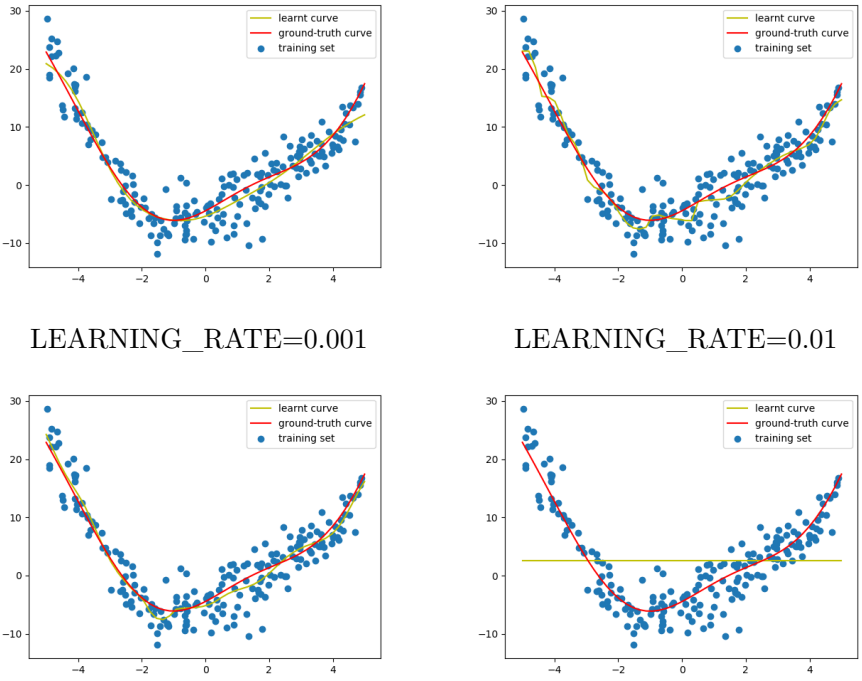
图 1: 练习 A：改变 NUM_TRAIN_EPOCHS

保持 NUM_TRAIN_SAMPLES = 200 不变，更改 NUM_TRAIN_EPOCHS 分别为 100，1000，10000，50000 得到以上四张图。图中红色曲线为真实函数图像，黄色曲线为经过神经网络学习生成的函数曲线。

由四张图中黄色和红色曲线拟合程度可知，NUM_TRAIN_EPOCHS = 1000 时神经网络的学习情况最好，而当增大 EPOCHS 的数量为 10000 和

50000，学习曲线反而拟合得更不好。这是因为当 NUM_TRAIN_EPOCHS 增加到 10000 和 50000 时，出现了过拟合的情况。

2.1.2 **练习** B: **改变** LEARNING_RATE



LEARNING_RATE=0.001　　LEARNING_RATE=0.01

LEARNING_RATE=0.1　　LEARNING_RATE=1

图 2: 练习 B：改变 LEARNING_RATE

由以上四张图可知，LEARNING_RATE=0.001 时学习曲线的拟合程度最好，而更大或更小的 LEARNING_RATE 的学习曲线结果更不好。这是因为使用过大的 LEARNING_RATE，或者过小的 LEARNING_RATE，都很有可能导致神经网络过早陷入一个较差的局部最小值难以自拔（梯度为零），但此时的局部最小值并不是全局的最优解。

2.1.3 **练习** C: **自定义函数** $f(x) = 0.5x^3 + sin(x) - 1$

在练习 C 中，我自定义了函数 $f(x) = 0.5x^3 + sin(x) - 1$，调整参数 LEARNING_RATE=0.001 和 NUM_TRAIN_EPOCHS=8000 得到学习曲线拟合效果最好的神经模型如下图：

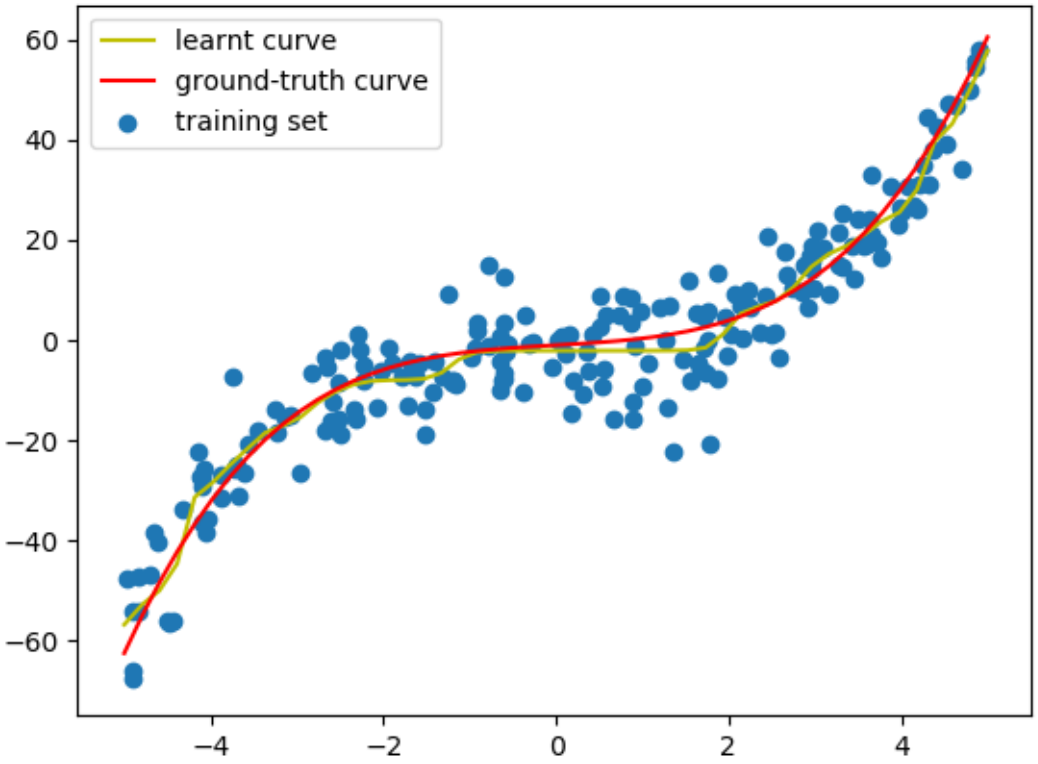


图 3: 练习 C：自定义函数 $f(x) = 0.5x^3 + sin(x) - 1$

```
for batch_idx, (inputs, targets) in enumerate(testloader):
    optimizer.zero_grad()#Sets gradients of all model parameters to
    outputs = model(inputs)                #outputs:   tensor([[-0.84

    _, predicted = outputs.max(1) # the predicted tensors having 1 t
    total += targets.size(0)#targets.size(0)都是128,即batch_size
    correct += predicted.eq(targets).sum().item()
    #eq对两个tensor中每个元素判断是否相等，以bool tensor的形式输出/
    #sum结果是如tensor(34),item后是34
    print('Epoch [%d] Batch [%d/%d]| Traininig Acc: %.3f%% (%d/%d)'
          % (epoch, batch_idx + 1, len(testloader),
             100. * correct / total, correct, total))
acc = 100. * correct / total
```

图 4: test 函数空缺部分

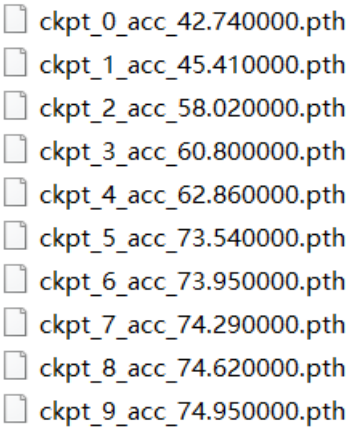


图 5: 10 个 epoch 的 test acc 及训练模型

2.2 实验 2：CIFAR-10 图片分类

填补 test 函数空缺部分代码如图 4：运行 ex2.py 程序可得到 10 个训练模型, 如图 5：（可选）思考：Train acc 和 Test acc 有什么关联和不同? 在 lr 从 0.1 变到 0.01 后，acc 发生了什么变化？为什么?

Answer：

1. train acc 是神经网络针对 50000 个训练图片进行分类得出来的正确率，期间神经网络的参数根据 learning rate 和 loss 不断更改；而 test acc 是使用 train 阶段训练完成的神经网络进行分类，期间神经网络参数不再改变。相同点是二者都是神经网络对图片的分类得出的准确率。

2. 当 lr 从 0.1 变到 0.01，acc 变化幅度下降了，原因是减小 lr 相当于减

小优化模型寻找最优解的步长，因此 acc 的变化幅度减小。当训练到一定阶段减小 lr 有助于模型更好地找到真实的最优解。

3 实验总结

3.1 实验概述

本实验通过使用基于深度学习框架 pytorch 的神经网络，实现了复杂函数的绘制和 CIFAR-10 图片的分类。

3.2 实验心得

本实验我收获良多，主要有以下几点

1. 了解神经网络的大致工作流程，知道了 pytorch 中一些函数的意义及使用方法
2. 学会训练一个神经网络的基本流程
3. 了解了图片分类时训练精度 train acc 与检测精度 test acc 的关联与不同，知道了 loss 和 lr 的关系和意义

最后，衷心感谢实验过程中各位老师和助教的帮助！