电工导 C 第十三次实验报告

姓名: 宋士祥　　学号:521030910013　　班级:F2103001

2022 年 12 月 15 日

1 实验概览

本次实验介绍了 PyTorch 等相关知识。

1.1 PyTorch 与神经网络的搭建

神经网络，可以简单理解为利用加权求和等方式对函数进行拟合。机器学习的目的即为寻找这一个函数 f。对于一个神经网络，其层数和结点数是确定的，需要训练的部分即为节点间的权值 w。对于下一个结点，简要来说，有 $z = g(\mathrm{w}^T A)$. g 为调整函数。

对于模型的训练，我们采用损失函数（即比较得到的函数与原函数进行比较，一个常用的损失函数为 $L = \frac{1}{2}(f(x) - F(x))^2$，$F(x)$ 为神经网络拟合的函数。）的方式进行处理。对于神经网络的更新，一个常用的处理方法为梯度下降法（然而本次实验不使用）。

在本部分中，我们需要完成：

1. 拟合一个初等函数，并给出分析
2. 使用 CIFAR-10 进行图片分类

1.2 利用卷积神经网络完成图片特征提取和以图搜图

早期的图像检索技术研究主要是基于文本的图像检索 (TBIR)，通过对图片的文本描述来查找图像库中具有相应标签的图像。随着技术的发展，基于内容的图像检索 (CBIR)，即以图搜图也成为了各大网站的标配功能。

以图搜图的核心其实就是在被检索的图像库 (Library) 中找到和参与检索的图片 (Query) 最相似的图片集合。

深度学习模型通过一种端到端学习的范式，使得模型能够学习到图像的这种特征。所以我们可以使用模型来将图像转换成一个能够代表图像的向量，从而用向量之间的相似程度代表图像之间的相似程度，从而解决以图搜图的任务。

卷积神经网络是一类包含卷积计算且具有深度结构的前馈神经网络（Feedforward Neural Networks），是深度学习（deep learning）的代表算法之一。其在图像处理的领域有着非常广泛的应用。

2 实验环境

本次实验采用所需的实验环境如下：

- Docker 中的`sjtumic/ee208` 镜像
- Python3（使用 VSCode 编译）

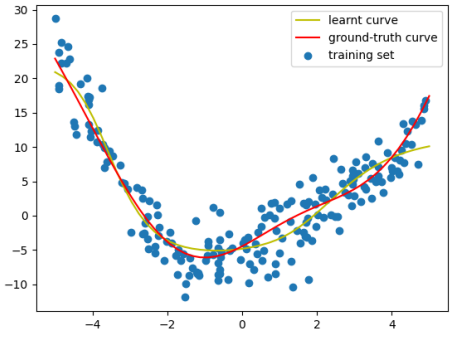
- `numpy` 扩展。
- `PyTorch` 环境。(均在 SJTUEE208 镜像中给出)

3 任务 1：初等函数拟合

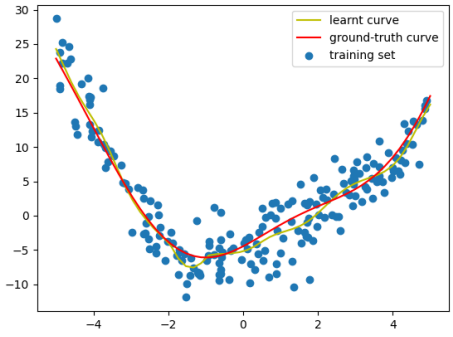
3.1 练习 A: 训练轮次

我们分别将训练轮次定位 100，1000，10000，50000 等。由于 Matplotlib 拥有保存图片的功能，因此我们不对代码进行修改。

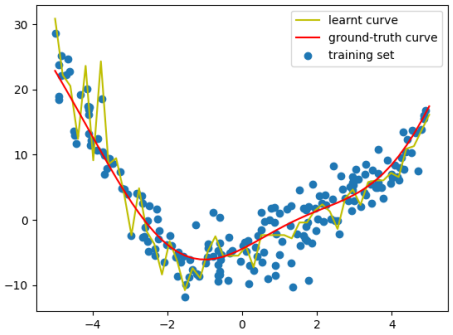
验证结果如下图：



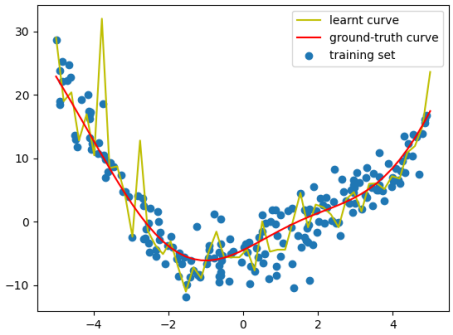
(a) 100



(b) 1000



(c) 10000



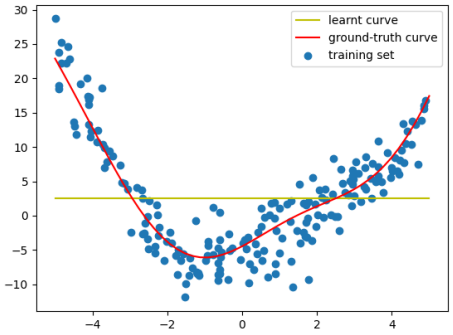
(d) 50000

我们预期的结果为训练轮数越高，得到函数拟合的精度越高。然而我们对比发现，结果并不相符。

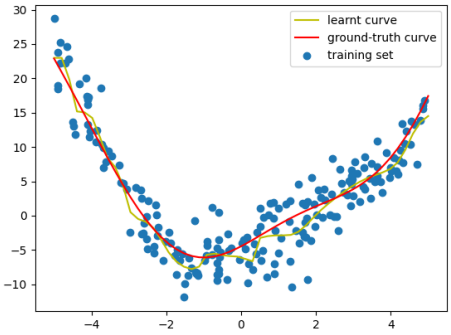
笔者查阅资料发现，对于神经网络的训练，存在一个“过拟合”的问题，过拟合的本质其实就是模型的表达能力过强，样本数不足导致模型记住了每一个样本的特征。这也是为什么函数都开始失真了。

3.2 练习 B: 学习率

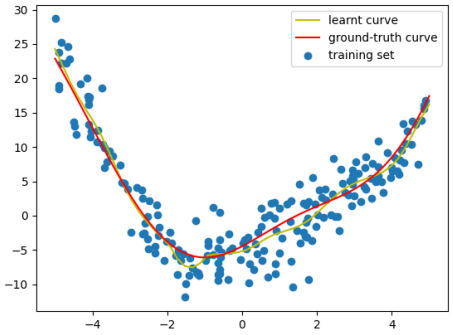
我们将训练轮次固定为 1000,，学习率则分别定为 1，0.1，0.01，0.001，对应结果如下：



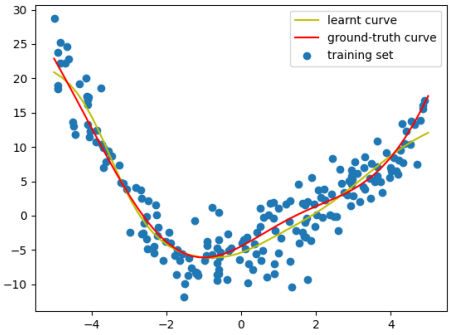
(e) 1



(f) 0.1



(g) 0.01



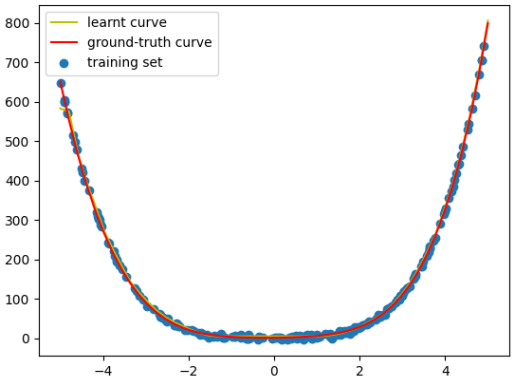
(h) 0.001

这个结果和我们的预期基本相符。它说明，在学习率过高的情况下，我们的神经网络起不到比较好的训练效果，而在学习率过低的情况下，我们会出现步长不足的问题。

3.3 练习 C：自己训练一个函数

我们选取（编）了一个函数 $f(x) = x^2 + x^4 + e^x + |\sin x|$。这个函数的特点在于它是一个凸函数，而且导函数非常大。这就意味着一开始我们的损失函数会非常大，我们需要调整参数以符合要求。

我们最终选取的参数为采样 200，训练轮数 2700，学习率 0.1，得到的结果如图：



3.4 问题思考

为什么训练轮数过高后反而效果并不好？

这里引入一个“过拟合”的概念。如果训练轮数过高，而神经网络记住了这些取样点的结果，那么它会采取尽可能接近这几个点的方式来完成这个函数，而不是靠接近这个函数。

而过这 200 个点的函数有很多，我们的初始函数只是其中一个。这就意味着我们的神经网络会尽可能接近这些点，但是最后的训练效果可能会很离谱。

解决方式：如果在允许的范围内，扩大训练集。

4 任务 2：CIFAR-10 图片分类

4.1 代码说明

由于助教已经写好了代码框架，实际上我们只需要写一下测试的部分就好了。由于测试部分的本质即为比较，和 train 部分的代码大同小异，因此我们实际只需要把 train 部分的代码略作改动即可。

对应代码：

```
def test(epoch):
    print('==> Testing...')
    model.eval()
    with torch.no_grad():
        correct = 0
        total = 0
        for batch_idx, (inputs, targets) in enumerate(testloader):
            # Just copy the code in "test"
            optimizer.zero_grad()
            outputs = model(inputs)
            _, predicted = outputs.max(1)
            total += targets.size(0)
            correct += predicted.eq(targets).sum().item()
        acc = float(correct) / total
        ########################################
```

同时，考虑 ppt 中的建议，我我们对于 verb|lr| 需要进行适当的修改。在学习次数在 0-4 时 lr 为 0.1，5-9 时为 0.01.

4.2 结果分析

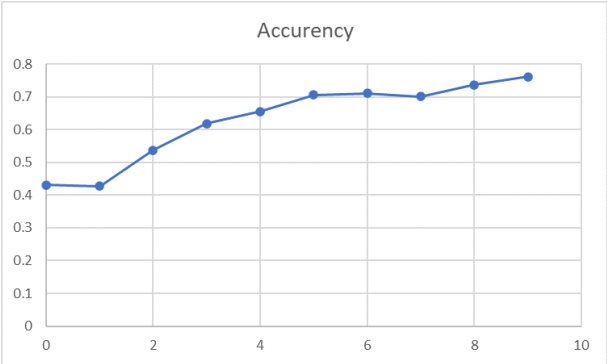
我们为了提高模型的准确率，我们将训练次数改为 11 次，得到的准确率有 70%。匹配效果相对较好。

具体数值：

```
Test 0: Acc: 0.430500
Test 1: Acc: 0.427500
Test 2: Acc: 0.537400
Test 3: Acc: 0.618400
Test 4: Acc: 0.655700
Test 5: Acc: 0.706100
```

```
Test 6: Acc: 0.711500
Test 7: Acc: 0.701300
Test 8: Acc: 0.736700
Test 9: Acc: 0.761200
```

我们利用 Excel 进行折线图的绘制，如下：



4.3 问题思考

4.3.1 关于 Test Acc 和 Train Acc

首先应当强调的是，在神经网络的训练中，训练集和测试集是应当完全分开的，不然神经网络会过度依赖训练集而无法做到真正的区分 (举个例子假设要识别路牌, 而我们的训练集中只有”上海” 的路牌, 那么神经网络可能会认为含” 上海” 的就是路牌, 而非长成路牌的样子的是路牌, 这就失去了意义). 因此,Test Acc 一定是会低于 Train Acc 的。但是 Test Acc 会相对更加客观。

4.3.2 lr 变小后

lr 变小后，Acc 相较上一组变化率下降，但是其总体仍保持上升的趋势，使得最后的 Accurency 相对较好。

5 任务 3：使用 Pytorch 提取图像特征

5.1 代码说明

我们使用微软的图片库 (https://www.microsoft.com/en-us/research/project/image-understanding/) 随机抽取 100 个图片作为我们的数据集。另外抽取十个图片作为我们的测试集。

对于特征提取的程序，我们仅略作改动（如将输出的向量拉平为一个一维数组），但是保证其完整性，我们 import 到另一个程序对其进行测试。

我们采取的取余弦的方式对其进行测试，向量的余弦越趋近 1，说明对比效果越好。

代码

```python
# Train
for i in range(1, 101):
    test_image = default_loader('./data/ ({}).bmp'.format(i))
    resnet50.append(extract_feature.main(test_image))


def comparison(img1, img2):
```

```
    sim = np.sum(img1 * img2) / (np.linalg.norm(img1) * np.linalg.norm(img2))
    return sim


for i in range(1, 11):
    print("Test {}!".format(i))
    test_image = default_loader('./test/ ({}).bmp'.format(i))
    out = extract_feature.main(test_image)
    compare = list()
    for j in range(100):
        compare.append(comparison(out, resnet50[j]))
    sort = sorted(compare)
    print(
        "The five most fitted images are:\n{} Score:{}\n{} Score:{}\n{} Score:{}\n{}
                                                Score:{}\n{} Score:{}"
        .format(
            compare.index(sort[-1]) + 1, sort[-1],
            compare.index(sort[-2]) + 1, sort[-2],
            compare.index(sort[-3]) + 1, sort[-3],
            compare.index(sort[-4]) + 1, sort[-4],
            compare.index(sort[-5]) + 1, sort[-5]))
```

5.2 对应结果

```
Test 1!
Prepare image data!
Extract features!
Time for extracting features: 0.15
Save features!
The five most fitted images are:
12 Score:0.9958114624023438
35 Score:0.995583176612854
38 Score:0.9955723285675049
37 Score:0.9955607652664185
13 Score:0.9955446720123291
Test 2!
Prepare image data!
Extract features!
Time for extracting features: 0.15
Save features!
The five most fitted images are:
5 Score:0.9960855841636658
6 Score:0.9960440993309021
4 Score:0.9959630370140076
3 Score:0.995910108089447
22 Score:0.9958860874176025
Test 3!
Prepare image data!
```

```
Extract features!
Time for extracting features: 0.16
Save features!
The five most fitted images are:
42 Score:0.9959242343902588
37 Score:0.9958661198616028
35 Score:0.9958024621009827
39 Score:0.9957846403121948
33 Score:0.995755672454834
Test 4!
Prepare image data!
Extract features!
Time for extracting features: 0.15
Save features!
The five most fitted images are:
57 Score:0.995871901512146
52 Score:0.9958577752113342
59 Score:0.9958552718162537
51 Score:0.9958348274230957
62 Score:0.9958264231681824
Test 5!
Prepare image data!
Extract features!
Time for extracting features: 0.15
Save features!
The five most fitted images are:
65 Score:0.9963147640228271
69 Score:0.9961764812469482
64 Score:0.9961698651313782
66 Score:0.9960054159164429
67 Score:0.9959712028503418
Test 6!
Prepare image data!
Extract features!
Time for extracting features: 0.15
Save features!
The five most fitted images are:
64 Score:0.9961156249046326
65 Score:0.995826005935669
67 Score:0.9958125948905945
68 Score:0.9956989288330078
69 Score:0.9956857562065125
Test 7!
Prepare image data!
Extract features!
```

```
Time for extracting features: 0.16
Save features!
The five most fitted images are:
84 Score:0.996376097202301
96 Score:0.9962178468704224
79 Score:0.9960218667984009
78 Score:0.9959855079650879
82 Score:0.9959573149681091
Test 8!
Prepare image data!
Extract features!
Time for extracting features: 0.14
Save features!
The five most fitted images are:
84 Score:0.9962936639785767
96 Score:0.9962146878242493
88 Score:0.9961373805999756
100 Score:0.9961228370666504
79 Score:0.996116042137146
Test 9!
Prepare image data!
Extract features!
Time for extracting features: 0.14
Save features!
The five most fitted images are:
81 Score:0.9961386919021606
91 Score:0.9959754347801208
98 Score:0.9959292411804199
83 Score:0.9959042072296143
99 Score:0.9958341121673584
Test 10!
Prepare image data!
Extract features!
Time for extracting features: 0.18
Save features!
The five most fitted images are:
84 Score:0.9961538910865784
78 Score:0.9960600137710571
1 Score:0.9960307478904724
92 Score:0.9959763288497925
98 Score:0.995930016040802
```

结果相对较吻合，不会出现一个草地上出现一只羊的图片被匹配到一个自行车这种情况的出现。

5.3 问题思考

5.3.1 关于 features 的写法

代码中的 features，其实是对 resnet50 的模拟。需要注意的是，由于我们已经预先加载好了 Resnet50，所以实际上我们只是简要模拟了关键步骤，如卷积，池化等步骤。逐步利用已经训练好的网络提取图像特征。

5.3.2 其他可用的图像特征提取网络

事实上，可供使用的神经网路有很多。在 ppt 中也略有介绍。

此处简要介绍一个 VGG 网络，它是 2014 年牛津大学的一篇论文中提到的，提取过程非常之复杂，此处不叙。

VGG 其实是对 AlexNet 的一个改进，其重点在将 5×5 的卷积层改为了两层 3×3 的卷积层。

6 拓展思考

深度学习的应用如今已经非常广阔，在计算机视觉，自然语音处理等方面取得了长足的进步。神经网络目前也有及其广泛的研究前景。

然而，神经网络最令人诟病的一点在于其过于“黑箱”化，很多时候只是调了几个参数，加了几个连接层而已。