编程题报告

191300087 左之睿 1710670843@qq.com

Problem 1

1.1

DNNs often fit target functions from low to high frequencies during the training.
即深度神经网络在训练过程中通常从低频到高频拟合目标函数。

1.2

F-principle让我们理解到DNN的拟合过程是从低频向高频进行，这样意味着在使用DNN处理高维信息更加重要的问题时，我们可以明确给出更高的优先级从而提高效率。

1.3

input frequency是指表示输入的不同位置上的强度的函数的频率，例如在图像识别问题中，input frequency就是表示图像在不同位置的像素点处的强度的函数的频率，这个频率对应着相邻像素的强度变化率，比如一张纯红的图片就对应着0频率
response frequency是指输入输出之间的一个一般的映射$f(x_1, x_2, ..., x_n)$的频率，记f作傅里叶变换得到$\hat{f}(k_1, ..., k_n), k_j$中的频率就代表了f关于$x_j$的变化率

F_princicle中使用的是response frequency

1.4

与1.3中的定义类似，但是使用离散非均匀傅里叶变换(NUDFT)代替原本的傅里叶变换
理由：
1、数据集是离散的，即目标函数的包含的信息仅在$\{x_i\}_{i=1}^n$处
2、NUDFT允许我们在训练过程中通过计算不同的$\mathbf{k}$下的$\Delta_F(\mathbf{k}) = |\hat{h}_k - \hat{y}_k|/|\hat{y}_k|$从而进行收敛性分析
3、$\hat{y}_k$能够有效的反映训练数据的频率结构特征。简单说来就是$\hat{y}_k$的高频对应对训练数据中附近点的输出的剧烈变化，然后使用滤波器我们就可以获得数据的低频部分作进一步分析

1.5

在F-principle中，作者分低维和高维两个方面进行例证。
对于低维情况中的例1，DNN在spatial domain上从整体向细节拟合，在Fourier domain上先拟合低频部分，再拟合高频部分。对于低维情况中的例2，DNN也是在训练中先记忆了coarse-grained landscape(即低频部分)，随后才记忆细节(即高频部分)。
在高维情况中(与论文类似，就不分开说了)，作者利用真实数据集，如MNIST，CIFAR10，首先是用投影方法进行试验，表明对于单个方向DNN表现出了F-principle。接下来为了证明整体上的F-principle，使用了较粗糙的过滤方法：将频域分为低频高频两个部分，在训练中不断检查二者的相对误差并考虑其收敛性，最终发现低频部分收敛的更快，与F-principle相符合。

Problem 2

2.1

2.1.1

我在low_dim.py的main函数中的ep循环中添加了如下代码：

```
        if ep % 2500 == 0:
            print(ep)
            plt.plot(x, y, color='r')
            plt.plot(x, pred_y, color='b', linestyle='--')
            plt.show()
```

即每训练2500次就输出一次拟合的图像，此处先贴出ep=0,2500时的图像
ep=0

ep=2500

可见当训练2500次时主体部分就已经拟合的很好了，故我选取ep=250,500,1000,1500,2000时的拟合情况再次进行绘图
以下依次是ep=250,500,1000,1500,2000绘制出的拟合图像
ep=250

ep=500

ep=1000

ep=1500

ep=2000

可见，随着ep的递增，在spatial domain上图像拟合确实是从主体向细节进行的

2.1.2

首先生成频率坐标

```
w = np.linspace(0, 20, num=100)
_w = w / 5 * np.pi * 2
freq=np.fft.rfftfreq(_w.shape[-1],d=1/240)
```

然后将目标函数值也进行FT

```
y_value=y.detach().numpy().flatten()
y_value_w=np.fft.rfft(y_value)
```

最后在循环中对计算的函数值进行FT，并且绘图

```
    if ep ==:
        y1=pred_y.flatten()
        pred_y_w=np.fft.rfft(y1)
        plt.plot(freq,abs(pred_y_w),color='b',linestyle='--') # 拟合曲线
        plt.plot(freq,abs(y_value_w),color='r') # 待拟合的
        plt.show()
```

本次直接选取ep=250,500,1000,1500,2000,5000,10000的情况进行绘图，图片如下

ep=250

ep=500

ep=1000

ep=1500

ep=2000

ep=5000

ep=10000

从图像可见，CNN确实是从低维部分开始拟合，逐步向高维继续。

2.2

这一题只完成了求解NUDFT的部分，没有进行e_{low}, e_{high}的计算，选择的是filter方法，分别取$\delta = 1，3，5$，要是在训练过程中对不同的δ都有$e_{low} < e_{high}$，那么f-principle成立

2.2.1

见mnist.py

2.2.2

$$e_{low} = (\frac{\sum_i |y_i^{low,\delta} - h_i^{low,\delta}|^2}{\sum_I |y_I^{low,\delta}|^2})^{1/2}$$

$$e_{high} = (\frac{\sum_i |y_i^{high,\delta} - h_i^{high,\delta}|^2}{\sum_I |y_I^{high,\delta}|^2})^{1/2}$$

添加的代码位于train函数中

```
def train(args, model, ...):
    ...
    for delta =1,3,5:
        for batch_idx, (data, target) in enumerate(train_loader):
            调用pynufft中的NUDFT计算all_ys,all_pred_ys的NUDFT
            依照上述公式计算e_low,e_high并进行比较
            若一直有e_low<e_high，那么f-principle成立
```

2.2.3

低频收敛更快，高频收敛慢

2.3

2.3.1

各种模型在训练过程中都有着一定的特性，我们需要把握住这些特性并善加利用以达成更好的效果。如DNN拟合函数时从低频到高频进行，那面对高频信息更为重要的问题我们就可以主动的赋予优先级，从而避免这个特性带来的不利影响

2.3.2 遇到的问题

(1) 在绘制低维情况的频谱图时，使用torch.fft.fft绘制出的图像严重错误，也不是单边频谱

解决方法：使用np.fft.rfft进行绘制，并且单独对频率轴进行FT，即如下代码

```
w = np.linspace(0, 20, num=100)
_w = w / 5 * np.pi * 2
freq = np.fft.rfftfreq(_w.shape[-1], d=1 / 240)
```

(2) 在绘制频谱图时，有时傅里叶变换会报出shape错误

解决：对于tensor张量，先用.detach().numpy()转换为numpy形式
在使用numpy.flatten()将变量转换为1维