ML HW13 Meta Learning Report

學號：b06901066　系級：電機三　姓名：孟妍

1. 請描述如何將助教的程式碼 (包含 classification 與 regression) 從二階的 MAML 改成一階的 MAML (作答請盡可能詳盡，以免助教誤會)，並且比較其最後在 classification 上的 accuracy (5-way-1-shot)。因此你的 GitHub 上會有 p1_classification.py 和 p1_regression.py 兩個檔案，分別是 classification 和 regression 的一階版本。(**配分**: classification 修改 (1) regression 修改 (1) report 一階做法在 classification 上的 accuracy (0.5))

   (1) regression 一階：backward的時候把create_graph設為False

```
146             # meta learning
147
148             y_tilde = sub_models[model_num](x[model_num][sample[:5],:])
149             little_l = F.mse_loss(y_tilde, y[model_num][sample[:5],:])
150             little_l.backward(create_graph = False)
151             sub_models[model_num].update(lr = 1e-2, parent = meta_model.model)
152             meta_optimizer.zero_grad()
153
154             y_tilde = sub_models[model_num](x[model_num][sample[5:],:])
155             meta_l =  meta_l + F.mse_loss(y_tilde, y[model_num][sample[5:],:])
```

   (2) classification 一階：在inner loop計算loss對 θ 微分的時候把create_graph設為False

```
91    for inner_step in range(inner_train_step):
92      train_label = create_label(n_way, k_shot).cuda()
93      logits = model.functional_forward(train_set, fast_weights)
94      loss = criterion(logits, train_label)
95      grads = torch.autograd.grad(loss, fast_weights.values(), create_graph = False)
96      fast_weights = OrderedDict((name, param - inner_lr * grad)
97                                 for ((name, param), grad) in zip(fast_weights.items(), grads))
```

   (3) 一階做法在 classification 上的 accuracy:

```
100%|██████████| 6/6 [00:04<00:00,  1.34it/s]  Testing accuracy:  0.8200000000000002
```

   改成一階做法後，在其餘參數不變的情況下，validation accuracy大約可以到0.85-0.86，最後testing accuracy為0.82

2. 請將 classification 的程式碼改成 inner loop 更新 5 次，並且使用 adagrad 與二階的 MAML，寫出其 pseudo code 與回報 accuracy (5-way-1-shot omniglot分類任務)。並且以 outer loop 的更新次數為橫軸，分類的準確率為縱軸作圖，比較其差異。因此你的 GitHub 上要有 p2.py，對應本題的程式碼。(**配分**:pseudo code (1) 作圖(1) report accuracy (0.5))

(1)pseudo code （modify inner_train_step and change optimizer）

原本training inner loop只update一次，改成update 5次如下：

```
meta_loss, acc = MAML(meta_model, optimizer, x, n_way, k_shot, q_query, loss_fn, inner_train_step = 5)
```

Adagrad的update如下:

$$W \leftarrow W - \eta \frac{1}{\sqrt{n+\epsilon}} \frac{\partial L}{\partial W}$$

$$n = \sum_{r=1}^{t} (\frac{\partial L_r}{\partial W_r})^2$$

$$W \leftarrow W - \eta \frac{1}{\sqrt{\sum_{r=1}^{t} (\frac{\partial L_r}{\partial W_r})^2 + \epsilon}} \frac{\partial L}{\partial W}$$

按照上面修改MAML這個function裡update參數的部份，這題我設ϵ=0.7。跟SGD update的不同在於learning rate可調整，變成lr $\times \frac{1}{\sqrt{n+\epsilon}}$，n為前面所有梯度值的平方和。

pseudo code:

1: initialize **n**(list)

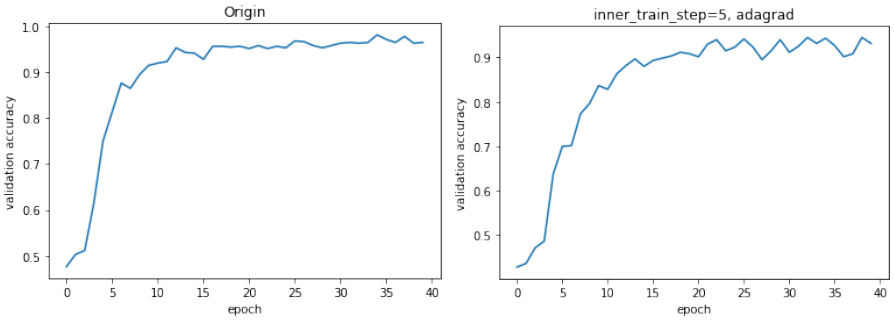
2: for inner_step in range(inner_train_step=5):

3: compute ∇loss (grads)

4: for every item in **n**, **n[i] += *grads*²**

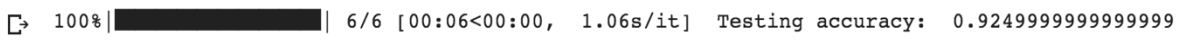
5: update $\theta = \theta - inner\ loop\ lr \times grads \times \frac{1}{\sqrt{n+\epsilon}}$

(2)作圖



inner loop的optimizer換成adagrad後，validation accuracy上升比較不陡，但收斂的震盪也較大。

(3) report accuracy



3. 實作論文 "How to train your MAML" (https://arxiv.org/abs/1810.09502) 中的一個 tip，解釋你使用的 tip 並且比較其在 5-way-1-shot 的 omniglot 分類任務上的 accuracy。助教其實已經實作了一個，請找出是哪一個 tip 並且不要重複。因此你的 GitHub 上要有 p3.py，對應本題的程式碼。(配分：report 實作 tip 後的 accuracy (1) 解釋你使用的 tip (1) 找出助教實作的 tip (0.5))

(1)實作tip後的accuracy

```
Testing accuracy:  0.9583333333333334
```

(2)使用的tip

這題我實作的tip是Second Order Derivative Cost → Derivative-Order Annealing，根據paper，training時我讓outer loop更新150次，前50次用一階做法更新，剩餘的epoch才用二階做法。

(3) 找出助教實作的tip

Absence of Batch Normalization Statistic Accumulation → Per-Step Batch Normalization Running Statistics (BNRS)

4. 請實作 reptile 在 omniglot dataset 上，訓練 5-way-1-shot 的分類任務，並且回報其 accuracy。這題應該在 GitHub 上會有 reptile.sh 的 shell script，執行方式詳見投影片。 (配分：程式碼 (2) 回報acc (0.5))