

1. 解釋程式碼

➤ select_user() :

```
def select_users(self, round, num_users):
    """
    Randomly select {num_users} users from all users
    Args:
        round: current round
        num_users: number of users to select
    Return:
        List of selected clients objects

    Hints:
        1. Default 10 users to select, you can modify the args {--n
        2. Note that {num_users} can not be larger than total users
    """

    if(num_users == len(self.users)):
        return self.users

    num_users = min(num_users, len(self.users))

    np.random.seed(round)

    return np.random.choice(self.users, num_users, replace=False)
```

若輸入的 num_users 大於 self.users，就將全部的 self.users 回傳。反之，就隨機挑選 num_users 個 users 回傳。

➤ aggregate_parameters() :

```
def aggregate_parameters(self):
    """
    Weighted sum all the selected users' model parameters by number of samples

    Args: None
    Return: None

    Hints:
        1. Use self.selected_users, user.train_samples.
        2. Replace the global model (self.model) with the aggregated model.
    """
    for param in self.model.parameters():
        param.data = torch.zeros_like(param.data)
    total = 0
    for user in self.selected_users:
        total += user.train_samples
    for user in self.selected_users:
        model = self.model.parameters()
        for server_param, user_param in zip(self.model.parameters(), user.get_parameters()):
            server_param.data = server_param.data + user_param.data.clone() * (user.train_samples / total)
```

首先將原本 server 的 model parameters 清 0，方便之後直接拿 users 的 parameters 取代。再來就將每個 users 的 parameters 乘上「users 所使用的 train_samples 在全部 samples 當中所佔的比例」，並加到 server 的 model parameters 當中，完成 aggregate。

➤ set_parameters():

```
def set_parameters(self, model, beta=1):
    """
    Replace the user's local model with the global model
    Args:
        model: the global model parameters
        beta: moving average model,
            i.e., user's model parameters = beta * global model parameters + (1 - beta) * user's mc
    Return:
        None

    Hint:
        1. You can use self.model (the user's model), model (global model parameters).
    """

    for user_param, global_param in zip(self.model.parameters(), model.parameters()):
        user_param.data = beta * global_param.data.clone() + (1 - beta) * user_param.data.clone()
```

根據 beta，將 global parameters 與 local parameters 做結合。結合的方式為：beta *

global model parameters + (1 - beta) * local model parameters。

2. 探討問題

I. data distribution

a. alpha=0.1

```
TRAIN #sample by user: [7517, 5817, 4245, 2533, 4726, 5121, 5122, 1664, 8721, 4534]
Dumping train data => ./u10c10-alpha0.1-ratiol.0/train/train.pt
7517 samples in total
c=2,n=1015| c=4,n=2966| c=5,n=4| c=6,n=47| c=8,n=3485|
5 Labels/ 7517 Number of training samples for user [0]:
5817 samples in total
c=2,n=240| c=5,n=4559| c=7,n=24| c=8,n=994|
4 Labels/ 5817 Number of training samples for user [1]:
4245 samples in total
c=2,n=2539| c=3,n=412| c=5,n=390| c=8,n=492| c=9,n=412|
5 Labels/ 4245 Number of training samples for user [2]:
2533 samples in total
c=1,n=817| c=2,n=58| c=3,n=52| c=7,n=1495| c=9,n=111|
5 Labels/ 2533 Number of training samples for user [3]:
4726 samples in total
c=1,n=284| c=2,n=209| c=6,n=753| c=7,n=3479| c=9,n=1|
5 Labels/ 4726 Number of training samples for user [4]:
5121 samples in total
c=0,n=4973| c=2,n=100| c=3,n=9| c=5,n=29| c=6,n=3| c=8,n=7|
6 Labels/ 5121 Number of training samples for user [5]:
5122 samples in total
c=4,n=2018| c=6,n=3102| c=7,n=1| c=9,n=1|
4 Labels/ 5122 Number of training samples for user [6]:
1664 samples in total
c=1,n=15| c=3,n=1489| c=4,n=15| c=6,n=1| c=8,n=21| c=9,n=123|
6 Labels/ 1664 Number of training samples for user [7]:
8721 samples in total
c=2,n=838| c=3,n=3034| c=6,n=498| c=9,n=4351|
4 Labels/ 8721 Number of training samples for user [8]:
4534 samples in total
c=0,n=27| c=1,n=3884| c=2,n=1| c=3,n=4| c=4,n=1| c=5,n=18| c=6,n=596| c=7,n=1| c=8,n=1| c=9,n=1|
10 Labels/ 4534 Number of training samples for user [9]:
100%| 10/10
TEST #sample by user: [500, 400, 500, 500, 500, 600, 400, 600, 400, 1000]
```

```
Average Global Accuracy = 0.3726, Loss = 1.91.
Best Global Accuracy = 0.4030, Loss = 1.81, Iter = 144.
Finished training.
```

當 alpha 為 0.1 時，資料在每個 user 上面分佈的非常不平均，所以的 users 都只有特定 label 的資料，導致每個 local model 都無法訓練的非常全面，無法平衡地學習到全部的 features，因此也導致最後的 global accuracy 只有 0.3 左右。

b. alpha=50

```
TRAIN #sample by user: [4901, 5481, 4907, 4629, 5449, 4889, 4770, 4601, 5073, 5300]
Dumping train data => ./u10c10-alpha50.0-ratiol.0/train/train.pt
4901 samples in total
c=0,n=504| c=1,n=489| c=2,n=491| c=3,n=510| c=4,n=415| c=5,n=500| c=6,n=489| c=7,n=606| c=8,n=447| c=9,n=450|
10 Labels/ 4901 Number of training samples for user [0]:
5481 samples in total
c=0,n=483| c=1,n=538| c=2,n=696| c=3,n=386| c=4,n=454| c=5,n=568| c=6,n=684| c=7,n=528| c=8,n=573| c=9,n=571|
10 Labels/ 5481 Number of training samples for user [1]:
4907 samples in total
c=0,n=554| c=1,n=456| c=2,n=452| c=3,n=513| c=4,n=458| c=5,n=439| c=6,n=552| c=7,n=482| c=8,n=576| c=9,n=425|
10 Labels/ 4907 Number of training samples for user [2]:
4629 samples in total
c=0,n=453| c=1,n=457| c=2,n=444| c=3,n=525| c=4,n=486| c=5,n=471| c=6,n=485| c=7,n=482| c=8,n=378| c=9,n=448|
10 Labels/ 4629 Number of training samples for user [3]:
5449 samples in total
c=0,n=502| c=1,n=569| c=2,n=544| c=3,n=483| c=4,n=482| c=5,n=599| c=6,n=510| c=7,n=547| c=8,n=661| c=9,n=552|
10 Labels/ 5449 Number of training samples for user [4]:
4889 samples in total
c=0,n=509| c=1,n=502| c=2,n=457| c=3,n=416| c=4,n=560| c=5,n=464| c=6,n=472| c=7,n=415| c=8,n=589| c=9,n=505|
10 Labels/ 4889 Number of training samples for user [5]:
4770 samples in total
c=0,n=522| c=1,n=484| c=2,n=430| c=3,n=564| c=4,n=505| c=5,n=454| c=6,n=399| c=7,n=429| c=8,n=405| c=9,n=578|
10 Labels/ 4770 Number of training samples for user [6]:
4601 samples in total
c=0,n=477| c=1,n=460| c=2,n=479| c=3,n=475| c=4,n=526| c=5,n=449| c=6,n=479| c=7,n=367| c=8,n=494| c=9,n=395|
10 Labels/ 4601 Number of training samples for user [7]:
5073 samples in total
c=0,n=450| c=1,n=525| c=2,n=457| c=3,n=582| c=4,n=507| c=5,n=551| c=6,n=466| c=7,n=553| c=8,n=392| c=9,n=590|
10 Labels/ 5073 Number of training samples for user [8]:
5300 samples in total
c=0,n=546| c=1,n=520| c=2,n=550| c=3,n=546| c=4,n=607| c=5,n=505| c=6,n=464| c=7,n=591| c=8,n=485| c=9,n=486|
10 Labels/ 5300 Number of training samples for user [9]:
100%| 10/10 [00:00<00:00, 12.01it/s]
TEST #sample by user: [1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000]
Dumping train data => ./u10c10-alpha50.0-ratiol.0/test/test.pt
Finish Generating User samples
```

```
Average Global Accuracy = 0.7834, Loss = 0.86.
Best Global Accuracy = 0.7931, Loss = 0.79, Iter = 143.
```

當 alpha 為 50 時，資料在每個 user 上面分佈的相對平均，所以的 users 都擁有

全部 label 的資料，因此每個 local model 可以平衡地學習到全部的 features，最後的 global accuracy 也達到 0.78 左右。

II. Number of users in a round

a. num_users=2

```
Average Global Accuracy = 0.6084, Loss = 1.19.  
Best Global Accuracy = 0.6680, Loss = 0.96, Iter = 145.  
Finished training.
```

b. num_users=10

```
Average Global Accuracy = 0.7869, Loss = 0.82.  
Best Global Accuracy = 0.7955, Loss = 0.80, Iter = 146.  
Finished training.
```

資料在每個 user 上面分佈的相對平均的情況下，可以看到每一輪更新 10 個 users 對 global accuracy 有著更想顯著的幫助。這是因為每個 users 都可能擁有獨特的數據，因此綜合每個 users 的參數更新都可以讓 global model 在整體上學習的更好。至於收斂的速度，我的實測結果為這兩種訓練方式都在 120~130 個 epochs 收斂。

3. 最終 acc 的輸出

```
Average Global Accuracy = 0.7869, Loss = 0.79.  
Best Global Accuracy = 0.7988, Loss = 0.77, Iter = 144.  
Finished training.
```

4. 此次作業中學到的重點

在 trace code 的過程當中，使我對 federated learning 有更深刻的了解。

- 我了解到 server 如何選擇要更新的 user model，也學習到 server 與 users 如何根據收到的參數，更新自身的 model。
- 並了解每個 users 所擁有的資料分布，對於最後的 global accuracy 會有很大的影響。server 需要適應不同的數據分佈，並綜合考慮各個使用者的貢獻，以獲得更好的全域準確度。
- 每一輪更新的 users 數量，對 global accuracy 也會有影響。如果每輪僅選擇少數 users 與模型更新，可能會造成部分 users 的資料未被充分利用，導致模型在表現不佳。