HW5 report

1. Explanation

severbase.py

select_users(self, model, beta)

```python
def select_users(self, round, num_users):
    ## TODO
    '''
    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 {--num_users} to change this hyper-parameter
        2. Note that {num_users} can not be larger than total users (i.e., num_users <= len(self.user))
    '''
    # ADD
    np.random.seed(round)  # use round as random seed
    user_indices = np.random.choice(range(len(self.users)), num_users, replace=False)
    self.selected_users = [self.users[i] for i in user_indices]
    return self.selected_users
    # END
```

使用當前 round 作為 random seed，隨機選擇 users 並且將結果存入 self.selected_users 中。

aggregate_parameters(self)

```python
def aggregate_parameters(self):
    ## TODO
    '''
    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.
    '''
    # ADD
    # Initializa new parameter, whose shape is same as model parameter
    new_params = [torch.zeros_like(param) for param in self.model.parameters()]

    # Count all selected user's samples
    total_train_samples = sum(user.train_samples for user in self.selected_users)
    # traverse all selected user，accumulate every user's parameter based on their weight
    for user in self.selected_users:
        user_params = user.get_parameters()
        for new_param, user_param in zip(new_params, user_params):
            new_param.data += user_param.data * (user.train_samples / total_train_samples)

    # update global model parameter
    for model_param, new_param in zip(self.model.parameters(), new_params):
        model_param.data.copy_(new_param.data)
    # END
```

首先先宣告一個參數列表，其形狀與 model parameter 的形狀一樣；接著，我們宣告 total_train_samples 去紀錄樣本總數，以利待會加權平均之計算；再來，我們計算各個 user 其相對應的加權平均並累加至 new_params 裡頭；最後將更新後的 global model parameter 更新至 model_param

userbase.py

set_parameters(self, mode, beta)

```
def set_parameters(self, model, beta=1):
    ## TODO
    '''
    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 model paramete
    Return:
        None

    Hint:
        1. You can use self.model (the user's model), model (global model parameters).
    '''
    # ADD
    for param, new_param in zip(self.model.parameters(), model.parameters()):
        param.data = beta * new_param.data + (1 - beta) * param.data
    # END
```

將 server 的 model parameter 送至 user 端的 model 中並使用 beta 控制 parameter 的更新方式。

2. 探討問題的原因

- Data Distribution

  alpha 0.1

```
-------------Round number: 149 -------------


Average Global Accurancy = 0.3850, Loss = 1.64.
Best Global Accurancy = 0.4146, Loss = 1.66, Iter = 144.
Finished training.
```

  當 alpha 為 0.1 時，其數據分佈較不均勻，每個 user 端的 data distribution 更傾向於某個類別，使得在 global model 在不同類別上的 training data 不足，進而導致正確率低落。

alpha 50.0

```
-------------Round number: 149 -------------

Average Global Accurancy = 0.8000, Loss = 0.79.
Best Global Accurancy = 0.8000, Loss = 0.79, Iter = 149.
Finished training.
```

當 alpha 為 50.0 時，其數據分佈較 alpha 為 0.1 時均勻，每個 user 端的 data distribution 更加貼近整體數據分佈，在訓練 global model 有更好的效果，使準確率大為提升。

- Number of users in a round

  num_users 2：

```
-------------Round number: 149 -------------

Average Global Accurancy = 0.5842, Loss = 1.25.
Best Global Accurancy = 0.6505, Loss = 1.03, Iter = 148.
Finished training.
```

每輪訓練的 user 端數量較少，model parameters 更新不夠充分，導致收斂速度較慢，最終準確率較低

num_users 10：

```
-------------Round number: 149 -------------

Average Global Accurancy = 0.8000, Loss = 0.79.
Best Global Accurancy = 0.8000, Loss = 0.79, Iter = 149.
Finished training.
```

每輪訓練的 user 端數量較多，model parameters 更新較 num_users 為 2 時充分，收斂速度較快，最終準確率較高

3. 最終 acc 的輸出截圖（--num_users 10, --alpha 100.0）

```
-------------Round number: 149 -------------

Average Global Accurancy = 0.8035, Loss = 0.76.
Best Global Accurancy = 0.8067, Loss = 0.75, Iter = 148.
Finished training.
```

4. 此次作業習得重點

   在這次的作業中，讓我更熟悉 Horizontal Federated Learning 的運作原理，並且在實作 select_users, aggregate_parameters, set_parameters 時，更了解這些 function 與各項參數之間是如何互相影響著 global model 的準確率與訓練效果。