

```

def generate_sigma(self):
    assert (len(self.uploaded_models) > 0)
    # 计算上传参数的标准差
    if self.model_sigma == None:
        return
    for param in self.model_sigma.parameters():
        param.data.zero_()
    for w, client_model in zip(self.uploaded_weights, self.uploaded_models):
        for sigma, client_param, global_param in zip(self.model_sigma.parameters(), client_model.pa
            sigma.data += pow((client_param.data - global_param.data), 2)*w
    for sigma in self.model_sigma.parameters():
        sigma.data = torch.sqrt(sigma.data)

def update_bias(self):
    for uploaded_model, id in zip(self.uploaded_models, self.uploaded_ids):
        for bias, local_param, global_param, sigma in zip(self.local_bias[id].parameters(), uploade
            bias.data += self.sigma_lr*self.div(local_param.data-global_param.data, sigma.data)

def send_models_ME(self):
    assert (len(self.clients) > 0)
    if self.model_sigma == None:
        self.model_sigma = copy.deepcopy(self.global_model)
        self.send_models()
    else:
        for client in self.clients:
            start_time = time.time()
            local_model = copy.deepcopy(self.global_model)
            for param, bias, sigma in zip(local_model.parameters(), self.local_bias[client.id].para
                param.data += bias.data*sigma.data
            client.set_parameters(local_model)

            client.send_time_cost['num_rounds'] += 1
            client.send_time_cost['total_cost'] += 2 * (time.time() - start_time)

```

每轮更新当中

服务器端：

(1) generate_sigma()根据上传的模型参数，计算模型参数标准差sigma

(2) update_bias()更新bias(bias即为下传模型时使用的参数)

(3) send_model_BS()下传模型。第一轮时使用原本的send_model()，第二轮开始使用bias和sigma下传个性化的模型

当前问题：第二轮参数出现Nan
sigma为0?

SCINet 文件 编辑 查看 运行 内核 标签页 设置 帮助

按文件名过滤

/ PFLlib / system /

名称	修改时间
flcore	2 个月前
models	1 小时前
utils	2 个月前
debug	5 分钟前
examples.sh	2 个月前
FedRep-Cifar...	2 个月前
FedRep-Cifar...	2 个月前
get_mean_st...	2 个月前
main.py	2 小时前
mnist_fedme...	5 分钟前

emil_h@worker-0:~/PFLlib/sj X debug X serverME.py X examples.sh X

```
490 4.0451e-05]], device= cuda:0 , requires_grad=True)+
491 Parameter containing:
492 tensor([4.1562e-04, 2.1190e-04, 2.3163e-04, 4.8921e-04, 1.9452e-04, 1.0990e-05,
493 3.1514e-04, 9.6834e-05, 2.3156e-04, 1.1721e-04, 2.4279e-04, 2.8014e-04,
494 2.6394e-06, 3.2319e-04, 2.7811e-04, 2.0706e-04, 1.1611e-10, 2.1103e-04,
495 0.0000e+00, 4.8099e-04, 4.6734e-04, 3.8155e-04, 1.4386e-07, 1.3431e-04,
496 2.5202e-05, 1.3576e-06, 3.2290e-04, 2.5317e-04, 4.0639e-04, 8.1501e-05,
497 2.3885e-04, 3.4244e-04, 2.8789e-04, 2.5255e-04, 3.7936e-04, 2.3959e-05,
498 2.3567e-04, 2.6481e-04, 5.5904e-04, 4.1447e-04, 9.0482e-05, 7.0857e-05,
499 4.8138e-04, 3.7047e-07, 2.2364e-04, 2.7503e-04, 1.2192e-04, 4.5575e-04,
500 6.0357e-05, 3.3060e-04, 7.1796e-05, 3.6963e-04, 2.2920e-05, 2.5948e-04,
501 2.2989e-04, 3.4907e-04, 3.8703e-04, 2.2032e-04, 3.1288e-04, 1.6196e-04,
502 1.2568e-04, 2.7879e-04, 0.0000e+00, 1.3676e-04, 4.9069e-04, 3.4060e-04,
503 1.3578e-04, 2.7766e-04, 2.3221e-10, 3.7127e-04, 3.1434e-06, 9.1614e-05,
504 5.7879e-04, 2.5872e-04, 3.8580e-04, 3.3634e-04, 3.5908e-04, 6.1710e-04,
505 1.2129e-04, 3.2281e-04, 6.2418e-06, 0.0000e+00, 1.1590e-06, 2.0975e-04,
506 1.0229e-04, 4.5472e-04, 3.1062e-04, 6.9113e-06, 1.5587e-04, 2.1996e-04,
507 1.1418e-04, 2.8495e-04, 4.5543e-04, 7.1432e-07, 3.8523e-04, 3.9963e-04,
508 0.0000e+00, 3.5542e-04, 2.9948e-04, 5.7137e-05, 1.3117e-04, 3.1811e-04,
509 7.6250e-05, 4.7958e-04, 2.4511e-04, 2.1924e-04, 3.3768e-04, 3.2602e-04,
510 6.3980e-05, 1.6166e-04, 4.1844e-04, 1.3369e-04, 4.0782e-04, 3.8823e-04,
511 3.5182e-04, 5.4785e-04, 4.0176e-04, 4.0205e-04, 9.3677e-05, 2.3720e-06,
512 2.7887e-04, 3.5830e-04, 2.9933e-04, 3.0654e-04, 4.7530e-05, 3.1235e-04,
513 4.6443e-10, 4.1108e-04, 0.0000e+00, 6.7269e-05, 5.2433e-04, 1.7376e-04,
514 1.9126e-04, 5.2653e-04, 2.3858e-04, 4.8645e-04, 9.8119e-07, 1.1659e-06,
515 1.0208e-04, 1.0714e-04, 3.4035e-04, 6.3532e-07, 4.1464e-04, 2.7381e-05,
516 6.7914e-05, 2.8727e-04, 4.2494e-05, 3.5454e-04, 3.2554e-04, 4.0555e-04,
517 6.5003e-05, 2.5708e-04, 1.7789e-04, 3.5514e-04, 4.3140e-04, 2.8858e-04,
518 3.5608e-05, 2.8918e-04, 2.8023e-04, 6.4579e-05, 3.8777e-04, 4.7606e-04,
519 5.5664e-07, 4.2651e-04, 2.8428e-04, 3.1750e-04, 6.2102e-05, 2.7637e-04,
```

0.0000 1/10

简易界面 1 \$ 0 Plain Text 行 495, 列 15 空格: 4 debug

简化版本：未使用sigma,
直接更新并使用bias

强于FedAvg与FedProx
(0.98)

```
Evaluate global model
Averaged Train Loss: 0.0089
Averaged Test Accuracy: 0.9884
Averaged Test AUC: 0.9984
Std Test Accuracy: 0.0056
Std Test AUC: 0.0010
----- time cost ----- 67.

Best accuracy.
0.9892049348869089

Average time cost per round.
47.99732605171204
File path: ../results/mnist_FedAvg_test_0.h5

Average time cost: 96051.19s.
Length: 2001
std for best accuracy: 0.0
mean for best accuracy: 0.9892049348869089
All done!

Storage on cpu
-----
Total Tensors: 36085612          Used Memory: 93.31M
-----
```