

复现报告

首先在本机上配置Anaconda+PyTorch(GPU版)+CUDA+cuDNN的代码环境,中间尽力经历了一系列版本不匹配和配置失败的问题,最后成功配置,不加赘述

clone项目到本地

阅读论文的代码文件以及readme.md文件,第一次尝试运行代码出现了OMP报错

```
OMP: Error #15: Initializing libiomp5md.dll, but found libiomp5md.dll already initialized.
```

上网查阅资料发现可能是因为电脑的torch版本太新(2.0.1+cu118),而本项目的运行环境是torch==1.7.0+cu101,根据错误提示在main.py中加入俩行代码后成功运行

```
import os
os.environ['KMP_DUPLICATE_LIB_OK']='True'
```

1.基本复现部分

按照论文内容分别在三个不同的数据集进行攻击:

ml-100数据集:

```
python main.py --dataset=ml-100k/ --attack=FedRecAttack --clients_limit=0.05 --items_limit=60 --part_percent=1
```

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=FedRecAttack --clients_limit=0.05 --items_limit=60 --part_percent=1
Arguments: attack=FedRecAttack, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.05, items_limit=60, part_percent=1, attack_lr=0.01, attack_batch_size=256
Load data done [0.9 s]. #user=990, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0 (init), (0.0848) on test, (0.0021, 0.0054, 0.0025) on target. [2.8s]
Iteration 1, loss = 72.84028 [3.9s], (0.0785) on test, (0.3923, 0.4062, 0.3809) on target. [1.4s]
Iteration 2, loss = 72.82955 [2.3s], (0.0732) on test, (0.5145, 0.5209, 0.5106) on target. [1.5s]
Iteration 3, loss = 72.82317 [3.0s], (0.0774) on test, (0.6174, 0.6206, 0.6139) on target. [1.4s]
Iteration 4, loss = 72.81738 [2.9s], (0.1145) on test, (0.7170, 0.7203, 0.7152) on target. [1.4s]
Iteration 5, loss = 72.80764 [3.2s], (0.2110) on test, (0.7867, 0.7867, 0.7861) on target. [1.4s]
Iteration 6, loss = 72.78014 [3.4s], (0.2736) on test, (0.8628, 0.8628, 0.8628) on target. [1.5s]
Iteration 7, loss = 72.69856 [3.0s], (0.2990) on test, (0.9121, 0.9121, 0.9121) on target. [1.7s]
Iteration 8, loss = 72.44743 [3.4s], (0.3234) on test, (0.9561, 0.9561, 0.9561) on target. [1.5s]
Iteration 9, loss = 71.69528 [3.0s], (0.3319) on test, (0.9775, 0.9775, 0.9775) on target. [1.7s]
Iteration 10, loss = 69.61645 [3.3s], (0.3362) on test, (0.9829, 0.9829, 0.9829) on target. [1.5s]
Iteration 11, loss = 64.96125 [3.0s], (0.3446) on test, (0.9861, 0.9871, 0.9864) on target. [1.6s]
Iteration 12, loss = 58.11625 [2.8s], (0.3383) on test, (0.9882, 0.9882, 0.9872) on target. [1.5s]
Iteration 13, loss = 52.26982 [2.7s], (0.3372) on test, (0.9904, 0.9904, 0.9900) on target. [1.5s]
Iteration 14, loss = 48.69690 [2.8s], (0.3457) on test, (0.9936, 0.9936, 0.9930) on target. [1.5s]
Iteration 15, loss = 46.58629 [2.7s], (0.3457) on test, (0.9936, 0.9946, 0.9939) on target. [1.5s]
Iteration 16, loss = 45.16842 [2.9s], (0.3796) on test, (0.9957, 0.9957, 0.9948) on target. [1.5s]
```

```

Iteration 179, loss = 22.62211 [2.9s], (0.5811) on test, (0.9496, 0.9550, 0.9479) on target. [1.5s]
Iteration 180, loss = 22.57600 [3.1s], (0.5673) on test, (0.9507, 0.9561, 0.9493) on target. [1.6s]
Iteration 181, loss = 22.54507 [3.6s], (0.5758) on test, (0.9528, 0.9593, 0.9518) on target. [1.3s]
Iteration 182, loss = 22.50189 [3.4s], (0.5769) on test, (0.9518, 0.9593, 0.9513) on target. [1.4s]
Iteration 183, loss = 22.46988 [2.7s], (0.5663) on test, (0.9496, 0.9539, 0.9477) on target. [1.5s]
Iteration 184, loss = 22.42142 [3.0s], (0.5822) on test, (0.9496, 0.9539, 0.9484) on target. [1.5s]
Iteration 185, loss = 22.38664 [3.3s], (0.5769) on test, (0.9507, 0.9571, 0.9495) on target. [1.3s]
Iteration 186, loss = 22.34509 [3.2s], (0.5779) on test, (0.9507, 0.9561, 0.9491) on target. [1.2s]
Iteration 187, loss = 22.31050 [3.2s], (0.5716) on test, (0.9464, 0.9528, 0.9439) on target. [1.3s]
Iteration 188, loss = 22.26075 [3.0s], (0.5864) on test, (0.9507, 0.9539, 0.9481) on target. [1.5s]
Iteration 189, loss = 22.21812 [3.0s], (0.5832) on test, (0.9486, 0.9539, 0.9478) on target. [1.6s]
Iteration 190, loss = 22.18132 [2.9s], (0.5737) on test, (0.9475, 0.9539, 0.9455) on target. [1.4s]
Iteration 191, loss = 22.14873 [2.7s], (0.5779) on test, (0.9507, 0.9582, 0.9498) on target. [1.6s]
Iteration 192, loss = 22.13088 [3.5s], (0.5779) on test, (0.9486, 0.9550, 0.9479) on target. [1.6s]
Iteration 193, loss = 22.07107 [3.2s], (0.5779) on test, (0.9539, 0.9603, 0.9516) on target. [1.5s]
Iteration 194, loss = 22.03251 [3.1s], (0.5832) on test, (0.9507, 0.9582, 0.9500) on target. [1.7s]
Iteration 195, loss = 21.99907 [3.0s], (0.5801) on test, (0.9507, 0.9561, 0.9486) on target. [1.7s]
Iteration 196, loss = 21.97591 [3.0s], (0.5843) on test, (0.9486, 0.9539, 0.9476) on target. [1.4s]
Iteration 197, loss = 21.93448 [2.9s], (0.5779) on test, (0.9475, 0.9539, 0.9455) on target. [1.6s]
Iteration 198, loss = 21.90582 [2.8s], (0.5811) on test, (0.9475, 0.9539, 0.9456) on target. [1.5s]
Iteration 199, loss = 21.86037 [3.1s], (0.5822) on test, (0.9453, 0.9518, 0.9443) on target. [1.1s]
Iteration 200, loss = 21.82274 [3.2s], (0.5822) on test, (0.9443, 0.9539, 0.9448) on target. [1.2s]

```

可以看到看到经过200轮攻击,对于最后一条输出结果解释如下

- 迭代次数: 第200次迭代。
- 损失值: 损失值为21.82274, 表示在该次迭代中模型的训练损失。
- 测试集性能: 在测试集上的性能指标为0.5822, 表示在Top-K推荐中命中率为0.5822。
- 目标物品性能: 性能指标有三项, ER@5和ER@10分别表示在前5, 10个推荐结果中, 用户实际交互的物品所占的比例。NDCG@10表示在前10个推荐结果中, 用户实际交互的物品的平均排名的归一化值, 本次结果在目标物品上的性能指标分别为0.9443、0.9539和0.9448
- 总时间: 整个迭代过程的运行时间为1.2秒。

这些性能指标用于评估推荐系统的效果, 损失值用于衡量模型的训练效果。根据这些指标的数值, 可以判断推荐系统在特定迭代中的性能和训练进展情况。

ml-1m数据集:

```
python main.py --dataset=ml-1m/ --attack=FedRecAttack --clients_limit=0.05 --items_limit=60 --part_percent=1
```

```

D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-1m/ --attack=FedRecAttack --clients_limit=0.05 --items_limit=60 --part_percent=1
Arguments: attack=FedRecAttack, dim=32, path=Data/, dataset=ml-1m/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.05, items_limit=60, part_percent=1, attack_lr=0.01, attack_batch_size=256
Load data done [8.7 s]. #user=6342, #item=3706, #train=994169, #test=6040
Target items: [3683].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0 (init), (0.1000) on test, (0.0040, 0.0055, 0.0027) on target. [11.0s]
Iteration 1, loss = 114.11534 [30.7s], (0.0786) on test, (0.5388, 0.5397, 0.5383) on target. [10.1s]
Iteration 2, loss = 114.10184 [23.3s], (0.1366) on test, (0.6361, 0.6363, 0.6349) on target. [9.8s]
Iteration 3, loss = 114.08768 [19.7s], (0.2914) on test, (0.7721, 0.7728, 0.7705) on target. [10.0s]
Iteration 4, loss = 114.00285 [20.7s], (0.3964) on test, (0.8783, 0.8793, 0.8777) on target. [10.0s]
Iteration 5, loss = 113.08117 [24.0s], (0.4219) on test, (0.9806, 0.9811, 0.9801) on target. [10.6s]
Iteration 6, loss = 104.91805 [21.3s], (0.4306) on test, (0.9964, 0.9964, 0.9962) on target. [10.0s]
Iteration 7, loss = 84.89798 [21.2s], (0.4416) on test, (0.9980, 0.9980, 0.9979) on target. [10.4s]
Iteration 8, loss = 75.31504 [22.9s], (0.4427) on test, (0.9985, 0.9985, 0.9985) on target. [10.3s]
Iteration 9, loss = 71.56753 [20.3s], (0.4475) on test, (0.9985, 0.9985, 0.9985) on target. [10.2s]
Iteration 10, loss = 69.49508 [22.7s], (0.4470) on test, (0.9987, 0.9987, 0.9987) on target. [9.9s]
Iteration 11, loss = 68.14760 [23.9s], (0.4500) on test, (0.9988, 0.9988, 0.9988) on target. [11.0s]
Iteration 12, loss = 67.16764 [21.9s], (0.4503) on test, (0.9988, 0.9988, 0.9988) on target. [11.2s]
Iteration 13, loss = 66.42269 [22.6s], (0.4517) on test, (0.9988, 0.9988, 0.9988) on target. [10.8s]
Iteration 14, loss = 65.80674 [23.3s], (0.4533) on test, (0.9988, 0.9988, 0.9988) on target. [10.9s]

```

管理员: C:\Windows\system32\cmd.exe

```
Iteration 173, loss = 41.22952 [32.6s], (0.5808) on test, (0.9660, 0.9722, 0.9626) on target. [17.7s]
Iteration 174, loss = 41.17556 [31.9s], (0.5877) on test, (0.9657, 0.9709, 0.9618) on target. [16.1s]
Iteration 175, loss = 41.15312 [31.6s], (0.5874) on test, (0.9669, 0.9709, 0.9632) on target. [15.3s]
Iteration 176, loss = 41.11272 [33.2s], (0.5841) on test, (0.9684, 0.9728, 0.9658) on target. [14.6s]
Iteration 177, loss = 41.06984 [31.9s], (0.5873) on test, (0.9667, 0.9713, 0.9626) on target. [14.8s]
Iteration 178, loss = 41.03724 [33.6s], (0.5844) on test, (0.9622, 0.9677, 0.9584) on target. [13.5s]
Iteration 179, loss = 40.99641 [27.7s], (0.5851) on test, (0.9675, 0.9710, 0.9630) on target. [9.4s]
Iteration 180, loss = 40.96635 [21.9s], (0.5887) on test, (0.9664, 0.9705, 0.9618) on target. [10.2s]
Iteration 181, loss = 40.94516 [21.5s], (0.5833) on test, (0.9690, 0.9737, 0.9662) on target. [10.4s]
Iteration 182, loss = 40.89756 [27.3s], (0.5833) on test, (0.9659, 0.9704, 0.9612) on target. [14.8s]
Iteration 183, loss = 40.87053 [31.4s], (0.5808) on test, (0.9664, 0.9709, 0.9624) on target. [14.7s]
Iteration 184, loss = 40.83575 [31.7s], (0.5861) on test, (0.9626, 0.9670, 0.9573) on target. [14.4s]
Iteration 185, loss = 40.79877 [30.4s], (0.5849) on test, (0.9656, 0.9700, 0.9616) on target. [15.1s]
Iteration 186, loss = 40.75543 [31.3s], (0.5877) on test, (0.9659, 0.9699, 0.9610) on target. [15.0s]
Iteration 187, loss = 40.73563 [31.7s], (0.5909) on test, (0.9665, 0.9712, 0.9626) on target. [14.5s]
Iteration 188, loss = 40.68999 [31.9s], (0.5869) on test, (0.9692, 0.9740, 0.9648) on target. [15.0s]
Iteration 189, loss = 40.66944 [29.9s], (0.5866) on test, (0.9649, 0.9690, 0.9601) on target. [12.5s]
Iteration 190, loss = 40.63970 [24.2s], (0.5876) on test, (0.9659, 0.9707, 0.9620) on target. [10.0s]
Iteration 191, loss = 40.59528 [20.1s], (0.5856) on test, (0.9656, 0.9697, 0.9602) on target. [10.2s]
Iteration 192, loss = 40.54930 [21.6s], (0.5932) on test, (0.9654, 0.9704, 0.9612) on target. [10.5s]
Iteration 193, loss = 40.52530 [21.4s], (0.5882) on test, (0.9695, 0.9728, 0.9647) on target. [10.0s]
Iteration 194, loss = 40.49235 [20.5s], (0.5868) on test, (0.9670, 0.9707, 0.9620) on target. [10.1s]
Iteration 195, loss = 40.48721 [22.4s], (0.5886) on test, (0.9664, 0.9710, 0.9622) on target. [9.9s]
Iteration 196, loss = 40.44938 [20.3s], (0.5869) on test, (0.9647, 0.9689, 0.9598) on target. [10.1s]
Iteration 197, loss = 40.40689 [20.2s], (0.5929) on test, (0.9619, 0.9669, 0.9564) on target. [10.1s]
Iteration 198, loss = 40.38150 [22.2s], (0.5851) on test, (0.9642, 0.9680, 0.9583) on target. [9.9s]
Iteration 199, loss = 40.34276 [20.0s], (0.5859) on test, (0.9641, 0.9685, 0.9587) on target. [10.0s]
Iteration 200, loss = 40.32190 [20.7s], (0.5902) on test, (0.9614, 0.9654, 0.9560) on target. [10.1s]
```

D:\Users\Administrator\Desktop\大作业3\FedRecAttack>

steam数据集:

```
python main.py --dataset=steam/ --attack=FedRecAttack --clients_limit=0.05 --
items_limit=60 --part_percent=1
```

选择管理员: C:\Windows\system32\cmd.exe

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=steam/ --attack=FedRecAttack --clients_limit=0.05 --items_limit=60 --part_percent=1
Arguments: attack=FedRecAttack, dim=32, path=Data/, dataset=steam/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.05, items_limit=60, part_percent=1, attack_lr=0.01, attack_batch_size=256
Load data done [2.3 s]. #user=3940, #item=5134, #train=110960, #test=3753
Target items: [3312].
output format: ((Sampled HR@10)), ((ER@5), (ER@10), (NDCG@10))
Iteration 0 (init), (0.1015) on test, (0.0101, 0.0165, 0.0084) on target. [7.8s]
Iteration 1, loss = 20.50143 [21.0s], (0.0738) on test, (0.4897, 0.4924, 0.4884) on target. [6.8s]
Iteration 2, loss = 20.49870 [15.9s], (0.0594) on test, (0.5001, 0.5028, 0.4971) on target. [7.4s]
Iteration 3, loss = 20.49666 [17.3s], (0.0568) on test, (0.5156, 0.5191, 0.5141) on target. [7.2s]
Iteration 4, loss = 20.49528 [14.6s], (0.0621) on test, (0.5356, 0.5369, 0.5324) on target. [7.3s]
Iteration 5, loss = 20.49394 [15.2s], (0.0645) on test, (0.5543, 0.5580, 0.5530) on target. [6.5s]
Iteration 6, loss = 20.49234 [16.2s], (0.0922) on test, (0.5786, 0.5812, 0.5771) on target. [7.3s]
Iteration 7, loss = 20.48987 [15.2s], (0.1407) on test, (0.6143, 0.6175, 0.6100) on target. [6.5s]
Iteration 8, loss = 20.48539 [15.3s], (0.2411) on test, (0.6492, 0.6516, 0.6452) on target. [7.0s]
Iteration 9, loss = 20.47629 [15.4s], (0.3360) on test, (0.7061, 0.7111, 0.7028) on target. [6.8s]
Iteration 10, loss = 20.45687 [17.6s], (0.4349) on test, (0.7207, 0.7266, 0.7175) on target. [9.9s]
Iteration 11, loss = 20.41459 [18.6s], (0.4985) on test, (0.7885, 0.7935, 0.7858) on target. [9.8s]
Iteration 12, loss = 20.32359 [20.6s], (0.5430) on test, (0.8074, 0.8168, 0.7997) on target. [9.6s]
Iteration 13, loss = 20.13309 [21.4s], (0.5710) on test, (0.8589, 0.8666, 0.8509) on target. [10.3s]
Iteration 14, loss = 19.75849 [22.3s], (0.5918) on test, (0.8789, 0.8914, 0.8594) on target. [9.9s]
Iteration 15, loss = 19.11579 [22.5s], (0.6006) on test, (0.9170, 0.9264, 0.8990) on target. [10.1s]
Iteration 16, loss = 18.21278 [20.1s], (0.6107) on test, (0.9467, 0.9536, 0.9309) on target. [9.6s]
Iteration 17, loss = 17.20161 [18.3s], (0.6256) on test, (0.9664, 0.9693, 0.9536) on target. [10.0s]
Iteration 18, loss = 16.24684 [18.8s], (0.6323) on test, (0.9765, 0.9768, 0.9745) on target. [10.2s]
Iteration 19, loss = 15.43365 [18.4s], (0.6435) on test, (0.9808, 0.9811, 0.9792) on target. [9.9s]
Iteration 20, loss = 14.74504 [19.3s], (0.6523) on test, (0.9803, 0.9819, 0.9765) on target. [10.0s]
Iteration 21, loss = 14.17446 [18.2s], (0.6656) on test, (0.9856, 0.9856, 0.9841) on target. [9.4s]
```

管理员: C:\Windows\system32\cmd.exe

```
Iteration 173, loss = 5.32399 [25.4s], (0.7583) on test, (0.9861, 0.9872, 0.9853) on target. [11.8s]
Iteration 174, loss = 5.31357 [24.9s], (0.7554) on test, (0.9859, 0.9872, 0.9848) on target. [11.8s]
Iteration 175, loss = 5.29823 [25.5s], (0.7570) on test, (0.9859, 0.9872, 0.9849) on target. [12.3s]
Iteration 176, loss = 5.28605 [24.9s], (0.7594) on test, (0.9859, 0.9869, 0.9847) on target. [12.4s]
Iteration 177, loss = 5.27458 [25.5s], (0.7626) on test, (0.9859, 0.9867, 0.9848) on target. [12.2s]
Iteration 178, loss = 5.26224 [24.7s], (0.7586) on test, (0.9848, 0.9859, 0.9841) on target. [12.1s]
Iteration 179, loss = 5.25094 [24.7s], (0.7562) on test, (0.9848, 0.9859, 0.9841) on target. [11.3s]
Iteration 180, loss = 5.23834 [25.1s], (0.7575) on test, (0.9848, 0.9853, 0.9839) on target. [12.2s]
Iteration 181, loss = 5.22655 [24.8s], (0.7549) on test, (0.9843, 0.9853, 0.9838) on target. [12.3s]
Iteration 182, loss = 5.21549 [25.2s], (0.7597) on test, (0.9856, 0.9869, 0.9846) on target. [12.3s]
Iteration 183, loss = 5.20377 [25.9s], (0.7597) on test, (0.9851, 0.9856, 0.9840) on target. [12.5s]
Iteration 184, loss = 5.19128 [25.3s], (0.7591) on test, (0.9843, 0.9848, 0.9836) on target. [11.4s]
Iteration 185, loss = 5.18225 [25.8s], (0.7586) on test, (0.9851, 0.9851, 0.9837) on target. [13.2s]
Iteration 186, loss = 5.17111 [26.4s], (0.7615) on test, (0.9856, 0.9861, 0.9848) on target. [12.5s]
Iteration 187, loss = 5.16018 [27.1s], (0.7573) on test, (0.9848, 0.9856, 0.9838) on target. [13.1s]
Iteration 188, loss = 5.14906 [28.3s], (0.7565) on test, (0.9851, 0.9856, 0.9840) on target. [13.2s]
Iteration 189, loss = 5.13792 [28.2s], (0.7586) on test, (0.9851, 0.9853, 0.9840) on target. [14.0s]
Iteration 190, loss = 5.12702 [32.5s], (0.7599) on test, (0.9843, 0.9856, 0.9840) on target. [18.7s]
Iteration 191, loss = 5.11719 [37.5s], (0.7589) on test, (0.9843, 0.9856, 0.9840) on target. [16.4s]
Iteration 192, loss = 5.10648 [35.5s], (0.7583) on test, (0.9853, 0.9861, 0.9845) on target. [17.5s]
Iteration 193, loss = 5.09608 [37.5s], (0.7583) on test, (0.9843, 0.9851, 0.9837) on target. [16.9s]
Iteration 194, loss = 5.08690 [37.1s], (0.7618) on test, (0.9843, 0.9851, 0.9838) on target. [18.5s]
Iteration 195, loss = 5.07555 [35.1s], (0.7575) on test, (0.9848, 0.9851, 0.9837) on target. [17.3s]
Iteration 196, loss = 5.06592 [36.2s], (0.7626) on test, (0.9840, 0.9845, 0.9832) on target. [17.7s]
Iteration 197, loss = 5.05566 [34.0s], (0.7642) on test, (0.9845, 0.9851, 0.9836) on target. [16.1s]
Iteration 198, loss = 5.04633 [36.3s], (0.7618) on test, (0.9843, 0.9848, 0.9833) on target. [17.1s]
Iteration 199, loss = 5.03751 [35.6s], (0.7634) on test, (0.9840, 0.9853, 0.9835) on target. [16.1s]
Iteration 200, loss = 5.02674 [35.7s], (0.7602) on test, (0.9840, 0.9853, 0.9835) on target. [17.3s]
```

D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=steam/ --attack=FedRecAttack --clients_limit=0.05 --items_limit=60 --part_percent=1

可以看到在论文的三个数据集上此攻击方法都运行成功

之后我下载了yelp数据集（Yelp Open Dataset是Yelp业务、评论和用户数据的子集），希望测试一下此方法在论文未涉及的数据集的表现。yelp数据集以json格式提供，需要转换成此方法需要的格式

此数据集很大所以运行得非常慢：

```
python main.py --dataset=yelp/ --attack=FedRecAttack --clients_limit=0.05 --
items_limit=60 --part_percent=1
```

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=yelp/ --attack=FedRecAttack --clients_limit
=0.05 --items_limit=60 --part_percent=1
Arguments: attack=FedRecAttack, dim=32, path=Data/, dataset=yelp/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=
1.0, clients_limit=0.05, items_limit=60, part_percent=1, attack_lr=0.01, attack_batch_size=256
load data done [5.7 s]. #user=15303, #item=25602, #train=555374, #test=14575
target items: [8798].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0 (init), (0.0991) on test, (0.0001, 0.0001, 0.0001) on target. [22.0s]
Iteration 1, loss = 26.42195 [98.7s], (0.0913) on test, (0.2767, 0.2859, 0.2667) on target. [22.7s]
Iteration 2, loss = 26.41800 [59.7s], (0.0878) on test, (0.1205, 0.1341, 0.1113) on target. [23.3s]
Iteration 3, loss = 26.41606 [57.6s], (0.0830) on test, (0.0069, 0.0107, 0.0058) on target. [23.3s]
Iteration 4, loss = 26.41460 [57.9s], (0.0800) on test, (0.2319, 0.2457, 0.2187) on target. [22.8s]
Iteration 5, loss = 26.41341 [57.0s], (0.0802) on test, (0.0469, 0.0591, 0.0398) on target. [22.5s]
Iteration 6, loss = 26.41233 [56.5s], (0.0813) on test, (0.1011, 0.1183, 0.0906) on target. [22.7s]
Iteration 7, loss = 26.41125 [57.1s], (0.0882) on test, (0.1755, 0.1942, 0.1607) on target. [23.3s]
Iteration 8, loss = 26.41005 [64.0s], (0.1024) on test, (0.3419, 0.3570, 0.3298) on target. [26.8s]
Iteration 9, loss = 26.40856 [63.2s], (0.1242) on test, (0.3317, 0.3491, 0.3184) on target. [30.2s]
Iteration 10, loss = 26.40651 [65.8s], (0.1662) on test, (0.1144, 0.1357, 0.0999) on target. [26.0s]
Iteration 11, loss = 26.40338 [68.4s], (0.2252) on test, (0.4582, 0.4695, 0.4456) on target. [25.3s]
Iteration 12, loss = 26.39820 [67.2s], (0.2998) on test, (0.4085, 0.4316, 0.3904) on target. [25.9s]
Iteration 13, loss = 26.38910 [67.8s], (0.3821) on test, (0.4717, 0.4975, 0.4568) on target. [25.3s]
Iteration 14, loss = 26.37274 [67.1s], (0.4549) on test, (0.5130, 0.5419, 0.4947) on target. [25.6s]
Iteration 15, loss = 26.34273 [66.3s], (0.5232) on test, (0.5294, 0.5634, 0.5063) on target. [26.9s]
Iteration 16, loss = 26.28766 [67.2s], (0.5714) on test, (0.2627, 0.3193, 0.2434) on target. [24.5s]
Iteration 17, loss = 26.18786 [65.5s], (0.6057) on test, (0.4204, 0.4840, 0.3753) on target. [24.9s]
Iteration 18, loss = 26.01456 [63.1s], (0.6248) on test, (0.3594, 0.4438, 0.3208) on target. [26.6s]
```

一开始运行良好，直到第38轮可能是攻击太强且数据集过大，模型损坏

与论文中所述数据集越密集攻击越困难这一论述相符合


```
Iteration 31, loss = 16.83787 [65.7s], (0.6912) on test, (0.1626, 0.1863, 0.1564) on target. [25.9s]
Iteration 32, loss = 16.28805 [62.2s], (0.6912) on test, (0.2473, 0.2701, 0.2454) on target. [24.6s]
Iteration 33, loss = 15.78929 [63.3s], (0.6930) on test, (0.2840, 0.3050, 0.2783) on target. [24.2s]
Iteration 34, loss = 15.33929 [62.5s], (0.6950) on test, (0.2508, 0.2679, 0.2398) on target. [24.7s]
Iteration 35, loss = 14.93358 [63.4s], (0.6991) on test, (0.2795, 0.2924, 0.2690) on target. [27.4s]
Iteration 36, loss = 14.56517 [60.2s], (0.6986) on test, (0.3050, 0.3212, 0.2922) on target. [22.3s]
Iteration 37, loss = 14.23215 [57.4s], (0.6989) on test, (0.2301, 0.2430, 0.2180) on target. [24.6s]
Iteration 38, loss = nan [204.7s], (1.0000) on test, (0.0000, 0.0000, 0.0000) on target. [24.6s]
Iteration 39, loss = nan [233.3s], (1.0000) on test, (0.0000, 0.0000, 0.0000) on target. [23.8s]
Iteration 40, loss = nan [234.8s], (1.0000) on test, (0.0000, 0.0000, 0.0000) on target. [24.1s]
Iteration 41, loss = nan [231.1s], (1.0000) on test, (0.0000, 0.0000, 0.0000) on target. [25.4s]
Iteration 42, loss = nan [225.1s], (1.0000) on test, (0.0000, 0.0000, 0.0000) on target. [22.5s]
Iteration 43, loss = nan [222.4s], (1.0000) on test, (0.0000, 0.0000, 0.0000) on target. [22.5s]
Iteration 44, loss = nan [224.4s], (1.0000) on test, (0.0000, 0.0000, 0.0000) on target. [23.0s]
Iteration 45, loss = nan [230.2s], (1.0000) on test, (0.0000, 0.0000, 0.0000) on target. [23.6s]
```

2.对照试验部分

尝试更换参数重新训练,按照作者思路对于ml-1k数据集进行对照试验

(1) 设置part percent为不变，改变client_limit（恶意用户比例）分别为0.01, 0.02, 0.03, 0.05, 0.10，结果如表所示：

client limit	1%	2%	3%	5%	10%
ER@5	0.0011	0.0043	0.6602	0.9443	0.9475
ER@10	0.0011	0.0075	0.717	0.9539	0.9539
NDCG@10	0.0011	0.001	0.6380	0.948	0.9424

与论文中提供结果基本一致，选取5%为最佳比例

实验截图如下：

0.01:

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=FedRecAttack --clients_limit=0.01 --items_limit=60 --part_percent=1
Arguments: attack=FedRecAttack, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.01, items_limit=60, part_percent=1, attack_lr=0.01, attack_batch_size=256
Load data done [0.9 s]. #user=952, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 196, loss = 21.68017 [10.5s], (0.5832) on test, (0.0011, 0.0011, 0.0011) on target.
Iteration 197, loss = 21.64107 [11.6s], (0.5843) on test, (0.0011, 0.0011, 0.0011) on target.
Iteration 198, loss = 21.62718 [10.2s], (0.5737) on test, (0.0011, 0.0011, 0.0011) on target.
Iteration 199, loss = 21.55910 [10.8s], (0.5726) on test, (0.0011, 0.0011, 0.0011) on target.
Iteration 200, loss = 21.53680 [10.8s], (0.5960) on test, (0.0011, 0.0011, 0.0011) on target.
```

0.02

```
管理员: C:\Windows\system32\cmd.exe
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=FedRecAttack --clients_limit=0.02 --items_limit=60 --part_percent=1
Arguments: attack=FedRecAttack, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.02, items_limit=60, part_percent=1, attack_lr=0.01, attack_batch_size=256
Load data done [0.9 s]. #user=961, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0 (init), (0.0848) on test, (0.0021, 0.0054, 0.0025) on target. [2.2s]
Iteration 1, loss = 72.84055 [4.3s], (0.0742) on test, (0.2369, 0.2615, 0.2276) on target. [1.7s]
Iteration 2, loss = 72.82975 [3.4s], (0.0679) on test, (0.4234, 0.4352, 0.4112) on target. [1.5s]
Iteration 3, loss = 72.82205 [3.4s], (0.0710) on test, (0.5466, 0.5573, 0.5395) on target. [1.5s]
Iteration 4, loss = 72.81394 [3.0s], (0.1082) on test, (0.6710, 0.6742, 0.6673) on target. [1.6s]
Iteration 5, loss = 72.80054 [3.4s], (0.2131) on test, (0.7803, 0.7814, 0.7748) on target. [1.6s]
Iteration 6, loss = 72.76665 [3.4s], (0.2842) on test, (0.8478, 0.8478, 0.8441) on target. [1.6s]
Iteration 7, loss = 72.66980 [2.9s], (0.3192) on test, (0.9046, 0.9057, 0.9039) on target. [1.6s]
Iteration 8, loss = 72.38023 [3.7s], (0.3340) on test, (0.9528, 0.9528, 0.9515) on target. [1.6s]
Iteration 9, loss = 71.52165 [3.6s], (0.3436) on test, (0.9775, 0.9775, 0.9770) on target. [1.7s]
Iteration 10, loss = 69.19540 [3.8s], (0.3309) on test, (0.9861, 0.9861, 0.9857) on target. [1.6s]
Iteration 11, loss = 64.16395 [3.8s], (0.3309) on test, (0.4051, 0.7063, 0.3713) on target. [1.7s]
Iteration 12, loss = 57.24733 [3.7s], (0.3372) on test, (0.0129, 0.0171, 0.0125) on target. [1.8s]
Iteration 13, loss = 51.65778 [4.5s], (0.3531) on test, (0.0032, 0.0032, 0.0027) on target. [2.4s]
Iteration 14, loss = 48.29337 [5.3s], (0.3425) on test, (0.0032, 0.0032, 0.0026) on target. [2.6s]
Iteration 15, loss = 46.23841 [5.8s], (0.3542) on test, (0.0021, 0.0032, 0.0025) on target. [2.8s]
Iteration 16, loss = 44.90632 [6.1s], (0.3574) on test, (0.0000, 0.0000, 0.0000) on target. [2.8s]
Iteration 17, loss = 43.92907 [6.0s], (0.3924) on test, (0.0011, 0.0011, 0.0011) on target. [2.9s]
Iteration 18, loss = 43.21267 [5.9s], (0.3849) on test, (0.0021, 0.0021, 0.0021) on target. [2.4s]
Iteration 19, loss = 42.61553 [4.7s], (0.3733) on test, (0.0021, 0.0021, 0.0021) on target. [2.3s]
Iteration 20, loss = 42.42358 [4.4s], (0.3623) on test, (0.0022, 0.0022, 0.0022) on target. [2.3s]
Iteration 178, loss = 22.38912 [5.0s], (0.5790) on test, (0.0021, 0.0043, 0.0028) on target. [2.5s]
Iteration 179, loss = 22.36089 [5.2s], (0.5758) on test, (0.0021, 0.0043, 0.0028) on target. [2.5s]
Iteration 180, loss = 22.32574 [4.8s], (0.5875) on test, (0.0021, 0.0043, 0.0028) on target. [2.4s]
Iteration 181, loss = 22.29192 [4.8s], (0.5769) on test, (0.0021, 0.0043, 0.0028) on target. [2.5s]
Iteration 182, loss = 22.24504 [5.3s], (0.5663) on test, (0.0021, 0.0054, 0.0031) on target. [2.5s]
Iteration 183, loss = 22.21801 [4.9s], (0.5748) on test, (0.0021, 0.0054, 0.0032) on target. [2.5s]
Iteration 184, loss = 22.16005 [4.9s], (0.5779) on test, (0.0021, 0.0054, 0.0031) on target. [2.5s]
Iteration 185, loss = 22.13547 [4.8s], (0.5875) on test, (0.0021, 0.0054, 0.0031) on target. [2.4s]
Iteration 186, loss = 22.09582 [5.2s], (0.5801) on test, (0.0021, 0.0054, 0.0032) on target. [2.5s]
Iteration 187, loss = 22.04043 [5.4s], (0.5822) on test, (0.0021, 0.0054, 0.0032) on target. [2.5s]
Iteration 188, loss = 22.02274 [5.0s], (0.5695) on test, (0.0032, 0.0054, 0.0033) on target. [2.5s]
Iteration 189, loss = 21.98285 [4.6s], (0.5822) on test, (0.0032, 0.0054, 0.0033) on target. [2.4s]
Iteration 190, loss = 21.95585 [5.5s], (0.5748) on test, (0.0032, 0.0064, 0.0036) on target. [2.5s]
Iteration 191, loss = 21.89178 [4.5s], (0.5779) on test, (0.0032, 0.0054, 0.0033) on target. [2.5s]
Iteration 192, loss = 21.86089 [4.7s], (0.5801) on test, (0.0032, 0.0054, 0.0033) on target. [2.5s]
Iteration 193, loss = 21.82058 [4.9s], (0.5811) on test, (0.0032, 0.0064, 0.0036) on target. [2.5s]
Iteration 194, loss = 21.79967 [5.6s], (0.5790) on test, (0.0032, 0.0064, 0.0037) on target. [2.5s]
Iteration 195, loss = 21.75558 [5.1s], (0.5716) on test, (0.0032, 0.0064, 0.0036) on target. [2.4s]
Iteration 196, loss = 21.72111 [4.8s], (0.5589) on test, (0.0043, 0.0075, 0.0041) on target. [2.5s]
Iteration 197, loss = 21.67985 [4.9s], (0.5790) on test, (0.0043, 0.0075, 0.0041) on target. [2.4s]
Iteration 198, loss = 21.64073 [5.1s], (0.5822) on test, (0.0043, 0.0075, 0.0041) on target. [2.6s]
Iteration 199, loss = 21.62256 [4.7s], (0.5779) on test, (0.0032, 0.0075, 0.0040) on target. [2.4s]
Iteration 200, loss = 21.57138 [4.4s], (0.5695) on test, (0.0043, 0.0075, 0.0041) on target. [2.5s]
```

0.03:

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=FedRecAttack --clients_limit=0.03 --items_limit=60 --part_percent=1
Arguments: attack=FedRecAttack, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.03, items_limit=60, part_percent=1, attack_lr=0.01, attack_batch_size=256
Load data done [1.4 s]. #user=971, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0 (init), (0.0848) on test, (0.0021, 0.0054, 0.0025) on target. [3.5s]
Iteration 1, loss = 72.84042 [5.3s], (0.0774) on test, (0.3237, 0.3387, 0.3119) on target. [1.9s]
Iteration 196, loss = 21.73717 [9.6s], (0.5716) on test, (0.6484, 0.6999, 0.6256) on target. [3.8s]
Iteration 197, loss = 21.69068 [8.5s], (0.5864) on test, (0.6517, 0.7010, 0.6256) on target. [4.0s]
Iteration 198, loss = 21.64971 [8.8s], (0.5790) on test, (0.6602, 0.7095, 0.6335) on target. [3.9s]
Iteration 199, loss = 21.63022 [8.8s], (0.5726) on test, (0.6581, 0.7149, 0.6385) on target. [3.9s]
Iteration 200, loss = 21.58924 [9.5s], (0.5822) on test, (0.6602, 0.7170, 0.6380) on target. [3.9s]
```

0.1

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=FedRecAttack --clients_limit=0.1 --items_limit=60 --part_percent=1
Arguments: attack=FedRecAttack, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.1, items_limit=60, part_percent=1, attack_lr=0.01, attack_batch_size=256
Load data done [1.4 s]. #user=1037, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0(init), (0.0848) on test, (0.0021, 0.0054, 0.0025) on target. [4.5s]
Iteration 1, loss = 72.84017 [8.3s], (0.0742) on test, (0.3934, 0.4062, 0.3828) on target. [2.4s]
Iteration 193, loss = 22.10274 [7.2s], (0.5843) on test, (0.9443, 0.9528, 0.9399) on target. [3.5s]
Iteration 194, loss = 22.05673 [8.0s], (0.5673) on test, (0.9486, 0.9539, 0.9421) on target. [3.5s]
Iteration 195, loss = 22.00807 [7.1s], (0.5864) on test, (0.9443, 0.9518, 0.9401) on target. [3.4s]
Iteration 196, loss = 21.98966 [7.6s], (0.5832) on test, (0.9389, 0.9507, 0.9389) on target. [3.5s]
Iteration 197, loss = 21.94604 [8.0s], (0.5748) on test, (0.9432, 0.9518, 0.9380) on target. [3.5s]
Iteration 198, loss = 21.92409 [7.7s], (0.5779) on test, (0.9518, 0.9582, 0.9481) on target. [3.5s]
Iteration 199, loss = 21.88342 [7.4s], (0.5896) on test, (0.9486, 0.9582, 0.9463) on target. [3.5s]
Iteration 200, loss = 21.84087 [8.2s], (0.5843) on test, (0.9475, 0.9539, 0.9424) on target. [3.5s]
```

(2) 设置client_limit为不变, 改变part_percent (公共互动比例) 分别为0.01, 0.02, 0.03, 0.05, 0.10, 结果如表所示:

part percent	1%	2%	3%	5%	10%
ER@5	0.9443	0.9786	0.9882	0.9914	0.9807
ER@10	0.9539	0.9861	0.9893	0.9946	0.9861
NDCG@10	0.9448	0.9772	0.9846	0.9887	0.9809

与论文结果基本一致, 取最佳比例为5%

实验截图如下:

2%:

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=FedRecAttack --clients_limit=0.05 --items_limit=60 --part_percent=2
Arguments: attack=FedRecAttack, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.05, items_limit=60, part_percent=2, attack_lr=0.01, attack_batch_size=256
Load data done [1.7 s]. #user=990, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0(init), (0.0848) on test, (0.0021, 0.0054, 0.0025) on target. [5.4s]
Iteration 1, loss = 72.84027 [9.6s], (0.0742) on test, (0.3912, 0.3976, 0.3788) on target. [3.1s]
Iteration 197, loss = 22.03341 [6.1s], (0.5811) on test, (0.9807, 0.9850, 0.9785) on target. [2.9s]
Iteration 198, loss = 21.98507 [6.4s], (0.5790) on test, (0.9839, 0.9882, 0.9802) on target. [2.9s]
Iteration 199, loss = 21.94500 [5.9s], (0.5949) on test, (0.9796, 0.9861, 0.9772) on target. [3.0s]
Iteration 200, loss = 21.90386 [6.2s], (0.5875) on test, (0.9786, 0.9861, 0.9772) on target. [3.0s]
```

3%

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=FedRecAttack --clients_limit=0.05 --items_limit=60 --part_percent=3
Arguments: attack=FedRecAttack, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.05, items_limit=60, part_percent=3, attack_lr=0.01, attack_batch_size=256
Load data done [2.0 s]. #user=990, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0(init), (0.0848) on test, (0.0021, 0.0054, 0.0025) on target. [7.2s]
Iteration 1, loss = 72.84026 [12.9s], (0.0732) on test, (0.3773, 0.3891, 0.3711) on target. [5.0s]
Iteration 196, loss = 22.19873 [5.0s], (0.5748) on test, (0.9871, 0.9893, 0.9835) on target. [2.7s]
Iteration 197, loss = 22.17423 [5.1s], (0.5769) on test, (0.9893, 0.9904, 0.9853) on target. [2.4s]
Iteration 198, loss = 22.13286 [4.9s], (0.5673) on test, (0.9861, 0.9882, 0.9839) on target. [2.4s]
Iteration 199, loss = 22.11574 [5.2s], (0.5875) on test, (0.9882, 0.9893, 0.9851) on target. [2.4s]
Iteration 200, loss = 22.07910 [4.9s], (0.5726) on test, (0.9882, 0.9893, 0.9846) on target. [2.6s]
```

5%

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=FedRecAttack --clients_limit=0.05 --items_limit=60 --part_percent=5
Arguments: attack=FedRecAttack, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.05, items_limit=60, part_percent=5, attack_lr=0.01, attack_batch_size=256
Load data done [2.8 s]. #user=990, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0(init), (0.0848) on test, (0.0021, 0.0054, 0.0025) on target. [11.4s]
Iteration 1, loss = 72.84029 [20.5s], (0.0817) on test, (0.4019, 0.4202, 0.3973) on target. [5.2s]
```



```
Iteration 196, loss = 22.30896 [4.0s], (0.5801) on test, (0.9882, 0.9946, 0.9884) on target. [1.6s]
Iteration 197, loss = 22.29452 [4.5s], (0.5854) on test, (0.9882, 0.9936, 0.9876) on target. [1.8s]
Iteration 198, loss = 22.25793 [4.0s], (0.5822) on test, (0.9904, 0.9936, 0.9888) on target. [1.6s]
Iteration 199, loss = 22.22527 [3.8s], (0.5737) on test, (0.9904, 0.9936, 0.9878) on target. [1.7s]
Iteration 200, loss = 22.18094 [2.8s], (0.5726) on test, (0.9914, 0.9946, 0.9887) on target. [1.3s]
```

10%

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=FedRecAttack --clients_limit=0.05 --items_limit=60 --part_percent=10
Arguments: attack=FedRecAttack, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.05, items_limit=60, part_percent=10, attack_lr=0.01, attack_batch_size=256
Load data done [2.9 s]. #user=990, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0(init), (0.0848) on test, (0.0021, 0.0054, 0.0025) on target. [15.4s]
Iteration 1, loss = 72.84032 [24.9s], (0.0732) on test, (0.4062, 0.4191, 0.3963) on target. [5.4s]
Iteration 197, loss = 22.34182 [4.2s], (0.5769) on test, (0.9786, 0.9839, 0.9788) on target. [1.6s]
Iteration 198, loss = 22.29743 [3.9s], (0.5801) on test, (0.9818, 0.9861, 0.9803) on target. [1.7s]
Iteration 199, loss = 22.24665 [4.5s], (0.5832) on test, (0.9807, 0.9850, 0.9794) on target. [1.6s]
Iteration 200, loss = 22.22613 [4.1s], (0.5695) on test, (0.9807, 0.9861, 0.9809) on target. [1.7s]
```

3.与其他方法作对比

参数规定:clients_limit = 0.05 part_percent = 1,测试论文提供的攻击方法Random,Bandwagon和Popular与FedRecAttack的性能差异:

method(clients_limit = 0.05)	FedRecAttack	Random	Bandwagon	Popular
ER@5	0.9443	0.000	0.000	0.0011
ER@10	0.9539	0.000	0.000	0.0011
NDCG@10	0.9448	0.000	0.000	0.0011

FedRecAttack性能远强于其余三个方法

因为性能指标太小,适当调大恶意用户比例再进行测试,这里调整参数clients_limit = 0.1 part_percent = 1;

method(clients_limit = 0.1)	FedRecAttack	Random	Bandwagon	Popular
ER@5	0.9475	0.0011	0.000	0.0032
ER@10	0.9539	0.0011	0.000	0.0075
NDCG@10	0.9424	0.005	0.000	0.0033

实验截图如下

Random:0.05

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=Random --clients_limit=0.05 --items_limit=60 --part_percent=1
Arguments: attack=Random, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.05, items_limit=60, part_percent=1, attack_lr=0.01, attack_batch_size=256
Load data done [1.1 s]. #user=990, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0(init), (0.0891) on test, (0.0021, 0.0054, 0.0025) on target. [2.1s]
Iteration 1, loss = 72.84071 [2.5s], (0.0689) on test, (0.0043, 0.0054, 0.0029) on target. [1.0s]
Iteration 196, loss = 21.64610 [4.8s], (0.5801) on test, (0.0000, 0.0000, 0.0000) on target. [2.5s]
Iteration 197, loss = 21.60145 [5.4s], (0.5758) on test, (0.0000, 0.0000, 0.0000) on target. [2.6s]
Iteration 198, loss = 21.56985 [5.3s], (0.5885) on test, (0.0000, 0.0000, 0.0000) on target. [2.5s]
Iteration 199, loss = 21.53751 [4.6s], (0.5779) on test, (0.0000, 0.0000, 0.0000) on target. [2.6s]
Iteration 200, loss = 21.51632 [4.8s], (0.5811) on test, (0.0000, 0.0000, 0.0000) on target. [2.7s]
```

Random:0.1

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=Random --clients_limit=0.1 --items_limit=60 --part_percent=1
Arguments: attack=Random, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.1, items_limit=60, part_percent=1, attack_lr=0.01, attack_batch_size=256
Load data done [0.9 s]. #user=1037, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0(init), (0.0870) on test, (0.0021, 0.0054, 0.0025) on target. [2.0s]
Iteration 1, loss = 72.84068 [2.4s], (0.0700) on test, (0.0043, 0.0054, 0.0027) on target. [1.0s]

Iteration 196, loss = 21.67303 [8.5s], (0.5970) on test, (0.0011, 0.0011, 0.0005) on target. [4.1s]
Iteration 197, loss = 21.61880 [8.0s], (0.5885) on test, (0.0011, 0.0011, 0.0004) on target. [4.0s]
Iteration 198, loss = 21.58230 [7.6s], (0.5748) on test, (0.0011, 0.0011, 0.0005) on target. [4.1s]
Iteration 199, loss = 21.53650 [8.5s], (0.5875) on test, (0.0011, 0.0011, 0.0005) on target. [4.2s]
Iteration 200, loss = 21.50799 [8.3s], (0.5854) on test, (0.0011, 0.0011, 0.0005) on target. [4.1s]
```

Bandwagon:0.05

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=Bandwagon --clients_limit=0.05 --items_limit=60 --part_percent=1
Arguments: attack=Bandwagon, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.05, items_limit=60, part_percent=1, attack_lr=0.01, attack_batch_size=256
Load data done [1.2 s]. #user=990, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0(init), (0.0933) on test, (0.0021, 0.0054, 0.0025) on target. [3.8s]
Iteration 1, loss = 72.84073 [3.5s], (0.0700) on test, (0.0043, 0.0054, 0.0029) on target. [1.8s]

Iteration 196, loss = 21.64073 [3.4s], (0.5190) on test, (0.0000, 0.0000, 0.0000) on target. [2.1s]
Iteration 197, loss = 21.61573 [4.1s], (0.5705) on test, (0.0000, 0.0000, 0.0000) on target. [1.9s]
Iteration 198, loss = 21.56709 [3.7s], (0.5769) on test, (0.0000, 0.0000, 0.0000) on target. [1.9s]
Iteration 199, loss = 21.54826 [4.0s], (0.5790) on test, (0.0000, 0.0000, 0.0000) on target. [2.0s]
Iteration 200, loss = 21.50386 [3.7s], (0.5970) on test, (0.0000, 0.0000, 0.0000) on target. [1.9s]
```

Bandwagon:0.1

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=Bandwagon --clients_limit=0.1 --items_limit=60 --part_percent=1
Arguments: attack=Bandwagon, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.1, items_limit=60, part_percent=1, attack_lr=0.01, attack_batch_size=256
Load data done [1.2 s]. #user=1037, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0(init), (0.0870) on test, (0.0021, 0.0054, 0.0025) on target. [3.3s]
Iteration 1, loss = 72.84069 [3.8s], (0.0817) on test, (0.0043, 0.0054, 0.0028) on target. [2.0s]
Iteration 196, loss = 21.68057 [8.3s], (0.5801) on test, (0.0000, 0.0000, 0.0000) on target. [4.4s]
Iteration 197, loss = 21.62651 [8.1s], (0.5769) on test, (0.0000, 0.0000, 0.0000) on target. [4.6s]
Iteration 198, loss = 21.61202 [7.9s], (0.5811) on test, (0.0000, 0.0011, 0.0003) on target. [3.2s]
Iteration 199, loss = 21.56540 [6.0s], (0.5822) on test, (0.0000, 0.0011, 0.0003) on target. [3.0s]
Iteration 200, loss = 21.53406 [6.3s], (0.5864) on test, (0.0000, 0.0000, 0.0000) on target. [3.0s]
```

Popular:0.05

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=Popular --clients_limit=0.05 --items_limit=60 --part_percent=1
Arguments: attack=Popular, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.05, items_limit=60, part_percent=1, attack_lr=0.01, attack_batch_size=256
Load data done [1.5 s]. #user=990, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0(init), (0.0923) on test, (0.0021, 0.0054, 0.0025) on target. [3.9s]
Iteration 1, loss = 72.84071 [4.3s], (0.0679) on test, (0.0043, 0.0054, 0.0029) on target. [2.3s]

Iteration 197, loss = 21.62055 [2.7s], (0.5811) on test, (0.0011, 0.0011, 0.0011) on target. [1.5s]
Iteration 198, loss = 21.59166 [2.8s], (0.5748) on test, (0.0011, 0.0011, 0.0011) on target. [1.5s]
Iteration 199, loss = 21.54119 [3.0s], (0.5769) on test, (0.0011, 0.0011, 0.0011) on target. [1.6s]
Iteration 200, loss = 21.52357 [3.2s], (0.5864) on test, (0.0011, 0.0011, 0.0011) on target. [1.5s]
```

Popular:0.1

```
D:\Users\Administrator\Desktop\大作业3\FedRecAttack>python main.py --dataset=ml-100k/ --attack=Bandwagon --clients_limit=0.1 --items_limit=60 --part_percent=1
Arguments: attack=Bandwagon, dim=32, path=Data/, dataset=ml-100k/, device=cuda, lr=0.01, epochs=200, batch_size=256, grad_limit=1.0, clients_limit=0.1, items_limit=60, part_percent=1, attack_lr=0.01, attack_batch_size=256
Load data done [1.2 s]. #user=1037, #item=1682, #train=99056, #test=943
Target items: [894].
output format: ({Sampled HR@10}), ({ER@5}, {ER@10}, {NDCG@10})
Iteration 0(init), (0.0870) on test, (0.0021, 0.0054, 0.0025) on target. [3.3s]
Iteration 1, loss = 72.84069 [3.8s], (0.0817) on test, (0.0043, 0.0054, 0.0028) on target. [2.0s]

Iteration 196, loss = 21.68057 [8.3s], (0.5801) on test, (0.0000, 0.0000, 0.0000) on target. [4.4s]
Iteration 197, loss = 21.62651 [8.1s], (0.5769) on test, (0.0000, 0.0000, 0.0000) on target. [4.6s]
Iteration 198, loss = 21.61202 [7.9s], (0.5811) on test, (0.0000, 0.0011, 0.0003) on target. [3.2s]
Iteration 199, loss = 21.56540 [6.0s], (0.5822) on test, (0.0000, 0.0011, 0.0003) on target. [3.0s]
Iteration 200, loss = 21.53406 [6.3s], (0.5864) on test, (0.0000, 0.0000, 0.0000) on target. [3.0s]
```

4.个人方法改进

新方法基本思想:

首先假设攻击者知道每个item的popularity,那么它可以在攻击之前首先构造一个模型,使得target item的embedding为最受欢迎的几个item embedding的平均.为了更好地promote,可以将这个embedding加倍.

然而实际场景下攻击者不知道最受欢迎的item,这可以通过item embedding的l2 norm估计:一般来说,norm越大的item对应的item越受欢迎.

核心代码讲解:

```
import torch
import numpy as np
from parse import args
import torch.nn as nn
import math

class OurAttackClient(nn.Module):
    def __init__(self, target_items):
        super().__init__()
        ...

    def eval_(self, _items_emb):
        # 评估函数, 返回 None, None (可以进一步根据需要实现特定的评估功能)

    def compute_k_popularities(self, k, items_emb):
        # 计算基于物品嵌入向量范数的前 k 个最受欢迎的物品
        norms = torch.norm(items_emb, dim=1)
        self.k_popularities = torch.argsort(norms, descending=True)[:k]

    def train_(self, items_emb):
        # 训练攻击模型的函数
        with torch.no_grad():
            if self.global_rounds == args.attack_round:
                # 如果处于攻击轮次, 计算 k_popularities 以生成 target_model
                self.compute_k_popularities(args.k, items_emb)
                top_k_embedding = items_emb[self.k_popularities]
                average_top_k_embedding = torch.mean(top_k_embedding, axis=0)
                self.target_model = items_emb.clone()
                # 更新 target_model, 将目标物品的嵌入向量更新为
                average_top_k_embedding * 10
                self.target_model[self.target_items] = average_top_k_embedding *
                10

            if self.global_rounds < args.attack_round:
                # 如果不是攻击轮次, 返回 None, None, None (可以根据具体条件进一步实现特定的
                逻辑)

                self.global_rounds += 1
                return None, None, None

            self.global_rounds += 1
            # 计算 items_emb_model_update, 通过从 target_model 中减去 items_emb, 然
            后乘以 alpha 进行缩放
            items_emb_model_update = (self.target_model - items_emb) *
            args.alpha
            # 选择要进行模型更新的物品 (选择差异范数最大的 args.items_limit -
            len(self.target_items) 个物品)
            chosen_items = torch.argsort(torch.norm(items_emb_model_update,
            dim=1), descending=True)[:args.items_limit - len(self.target_items)]
```

```

# 从 chosen_items 中排除目标物品，以避免冗余更新
chosen_items = torch.tensor(list(set(chosen_items.tolist()) -
set(self.target_items))).to(args.device)
# 将目标物品添加到 chosen_items 以进行更新
chosen_items = torch.cat((chosen_items,
torch.tensor(self.target_items).to(args.device)), dim=0)
return chosen_items, items_emb_model_update[chosen_items], None

```

结果展示:

和论文中的方法进行效果比较,采用的参数是clients_limit=0.03,part_percent=0.1,最终效果如下,新方法优于论文方法.上面是论文方法的结果下面是新方法

```

Iteration 276, loss = 13.58421 [3.2s], (0.6045) on test, (0.0804, 0.1318, 0.0694) on target. [1.0s]
Iteration 277, loss = 13.58352 [3.1s], (0.5917) on test, (0.0825, 0.1318, 0.0699) on target. [1.0s]
Iteration 278, loss = 13.58283 [2.9s], (0.5822) on test, (0.0815, 0.1329, 0.0706) on target. [0.9s]
Iteration 279, loss = 13.57993 [3.1s], (0.5737) on test, (0.0815, 0.1308, 0.0694) on target. [1.0s]
Iteration 280, loss = 13.57855 [3.4s], (0.5854) on test, (0.0793, 0.1297, 0.0684) on target. [1.0s]
Iteration 281, loss = 13.57725 [3.2s], (0.5832) on test, (0.0772, 0.1275, 0.0668) on target. [1.0s]
Iteration 282, loss = 13.57383 [3.2s], (0.5928) on test, (0.0804, 0.1286, 0.0687) on target. [1.0s]
Iteration 283, loss = 13.57536 [3.1s], (0.5960) on test, (0.0793, 0.1275, 0.0676) on target. [0.9s]
Iteration 284, loss = 13.57221 [3.2s], (0.5875) on test, (0.0793, 0.1286, 0.0687) on target. [0.9s]
Iteration 285, loss = 13.57268 [3.1s], (0.5970) on test, (0.0804, 0.1297, 0.0692) on target. [0.9s]
Iteration 286, loss = 13.56916 [2.8s], (0.5875) on test, (0.0782, 0.1297, 0.0689) on target. [1.0s]
Iteration 287, loss = 13.57073 [2.8s], (0.5907) on test, (0.0782, 0.1318, 0.0697) on target. [0.9s]
Iteration 288, loss = 13.56915 [2.7s], (0.5907) on test, (0.0804, 0.1361, 0.0723) on target. [0.9s]
Iteration 289, loss = 13.56913 [3.1s], (0.5928) on test, (0.0836, 0.1383, 0.0734) on target. [0.9s]
Iteration 290, loss = 13.56523 [3.4s], (0.5949) on test, (0.0836, 0.1393, 0.0736) on target. [1.0s]
Iteration 291, loss = 13.56404 [3.3s], (0.5970) on test, (0.0879, 0.1468, 0.0777) on target. [1.0s]
Iteration 292, loss = 13.56219 [3.2s], (0.5928) on test, (0.0868, 0.1426, 0.0752) on target. [1.0s]
Iteration 293, loss = 13.56148 [3.3s], (0.5875) on test, (0.0857, 0.1426, 0.0752) on target. [0.9s]
Iteration 294, loss = 13.56031 [3.2s], (0.5896) on test, (0.0879, 0.1447, 0.0778) on target. [1.0s]
Iteration 295, loss = 13.55905 [3.2s], (0.5907) on test, (0.0900, 0.1490, 0.0783) on target. [1.0s]
Iteration 296, loss = 13.55483 [3.2s], (0.6034) on test, (0.0911, 0.1597, 0.0833) on target. [0.9s]
Iteration 297, loss = 13.55668 [2.7s], (0.5885) on test, (0.0932, 0.1651, 0.0853) on target. [0.9s]
Iteration 298, loss = 13.55769 [3.0s], (0.5864) on test, (0.0922, 0.1640, 0.0850) on target. [0.9s]
Iteration 299, loss = 13.55188 [3.0s], (0.5970) on test, (0.0932, 0.1672, 0.0862) on target. [0.9s]
Iteration 300, loss = 13.55419 [2.7s], (0.5864) on test, (0.0922, 0.1693, 0.0868) on target. [0.9s]

```

```

Iteration 276, loss = 14.61185 [3.0s], (0.5938) on test, (0.9753, 0.9795, 0.9739) on target. [1.0s]
Iteration 277, loss = 14.63897 [3.0s], (0.5716) on test, (0.9753, 0.9775, 0.9704) on target. [1.0s]
Iteration 278, loss = 14.60908 [2.9s], (0.5854) on test, (0.9753, 0.9786, 0.9721) on target. [1.0s]
Iteration 279, loss = 14.61881 [3.0s], (0.5928) on test, (0.9753, 0.9786, 0.9722) on target. [0.9s]
Iteration 280, loss = 14.63068 [3.0s], (0.5843) on test, (0.9764, 0.9786, 0.9713) on target. [0.9s]
Iteration 281, loss = 14.62653 [2.9s], (0.5832) on test, (0.9732, 0.9786, 0.9707) on target. [0.9s]
Iteration 282, loss = 14.64172 [3.0s], (0.5981) on test, (0.9743, 0.9775, 0.9703) on target. [0.9s]
Iteration 283, loss = 14.61922 [3.2s], (0.5938) on test, (0.9721, 0.9775, 0.9696) on target. [0.9s]
Iteration 284, loss = 14.60825 [3.0s], (0.5864) on test, (0.9732, 0.9775, 0.9701) on target. [0.9s]
Iteration 285, loss = 14.61918 [2.9s], (0.5769) on test, (0.9743, 0.9753, 0.9683) on target. [1.0s]
Iteration 286, loss = 14.61683 [3.0s], (0.5864) on test, (0.9743, 0.9775, 0.9715) on target. [0.9s]
Iteration 287, loss = 14.61739 [2.9s], (0.5779) on test, (0.9732, 0.9775, 0.9700) on target. [0.9s]
Iteration 288, loss = 14.60065 [3.1s], (0.5949) on test, (0.9732, 0.9775, 0.9693) on target. [0.9s]
Iteration 289, loss = 14.60834 [3.3s], (0.5907) on test, (0.9743, 0.9775, 0.9700) on target. [0.9s]
Iteration 290, loss = 14.59413 [2.8s], (0.5843) on test, (0.9732, 0.9764, 0.9684) on target. [0.9s]
Iteration 291, loss = 14.59046 [3.0s], (0.5875) on test, (0.9764, 0.9775, 0.9718) on target. [1.0s]
Iteration 292, loss = 14.60469 [3.1s], (0.5906) on test, (0.9753, 0.9786, 0.9703) on target. [1.0s]
Iteration 293, loss = 14.59492 [3.2s], (0.5790) on test, (0.9732, 0.9775, 0.9702) on target. [1.0s]
Iteration 294, loss = 14.59323 [3.1s], (0.5864) on test, (0.9732, 0.9764, 0.9684) on target. [1.0s]
Iteration 295, loss = 14.57084 [3.0s], (0.5864) on test, (0.9732, 0.9775, 0.9698) on target. [0.9s]
Iteration 296, loss = 14.59494 [2.9s], (0.5854) on test, (0.9743, 0.9775, 0.9695) on target. [0.9s]
Iteration 297, loss = 14.57455 [3.0s], (0.5854) on test, (0.9732, 0.9764, 0.9689) on target. [0.9s]
Iteration 298, loss = 14.57047 [2.8s], (0.6002) on test, (0.9732, 0.9775, 0.9701) on target. [0.9s]
Iteration 299, loss = 14.58484 [2.7s], (0.5875) on test, (0.9743, 0.9764, 0.9704) on target. [0.9s]
Iteration 300, loss = 14.58700 [2.8s], (0.5864) on test, (0.9732, 0.9786, 0.9703) on target. [0.9s]

```

同时设置前50轮只收集信息,第51轮开始攻击,可以看到结果中的数据在第51轮从零突变到0.98+,并维持在较高水平.(上面是论文方法的结果下面是新方法)

```

Arguments: attack=FcdAccAttack,dim=32,path=Data/,k=50,Lambda=1,alpha=1,attack_round=50,dataset=ml_100k/,device=cuda,lr=0.05,cpoes=300,batch_size=
256,grad_limit=0.0,clients_limit=0.02,items_limit=60,per_client_max_attack_lr=0.05,attack_batch_size=256,aggregation=mean,cpu=1
Load data done [6.4 s]. Nuser=971, #item=1682, #train=99856, #test=913
Target items: [894]
Output format: ({Sampled HR@10}), ({ER@5},{ER@10},{NDC@10})
Iteration 0 (init), (0.6844) on test, (0.8871, 0.8954, 0.8874) on target. [0.4s]
Iteration 1, loss = 72.84350 [2.6s], (0.1039) on test, (0.0815, 0.1115, 0.0722) on target. [0.9s]
Iteration 2, loss = 72.85262 [2.6s], (0.1184) on test, (0.1451, 0.1348, 0.1219) on target. [0.9s]
Iteration 3, loss = 72.82125 [2.6s], (0.1326) on test, (0.6034, 0.6313, 0.5834) on target. [0.9s]
Iteration 4, loss = 72.80532 [2.5s], (0.1453) on test, (0.7610, 0.7786, 0.7483) on target. [0.9s]
Iteration 5, loss = 72.77523 [2.8s], (0.1972) on test, (0.8789, 0.8896, 0.8703) on target. [0.9s]
Iteration 6, loss = 72.79622 [3.3s], (0.2503) on test, (0.9411, 0.9443, 0.9374) on target. [1.0s]
Iteration 7, loss = 72.57948 [3.0s], (0.2768) on test, (0.9668, 0.9668, 0.9658) on target. [1.0s]
Iteration 8, loss = 72.05746 [3.1s], (0.3001) on test, (0.9754, 0.9775, 0.9764) on target. [0.9s]
Iteration 9, loss = 70.82756 [3.2s], (0.3043) on test, (0.9861, 0.9861, 0.9861) on target. [1.0s]
Iteration 10, loss = 67.86239 [3.1s], (0.2884) on test, (0.9925, 0.9925, 0.9925) on target. [0.9s]
Iteration 11, loss = 61.99706 [3.2s], (0.2812) on test, (0.9925, 0.9925, 0.9925) on target. [0.9s]
Iteration 12, loss = 54.34340 [3.2s], (0.2948) on test, (0.0450, 0.0536, 0.0409) on target. [0.9s]
Iteration 13, loss = 47.88914 [3.2s], (0.3139) on test, (0.0054, 0.0064, 0.0064) on target. [0.9s]
Iteration 14, loss = 43.76459 [2.6s], (0.3404) on test, (0.0037, 0.0037, 0.0037) on target. [0.9s]
Iteration 15, loss = 41.19083 [3.1s], (0.3421) on test, (0.0021, 0.0021, 0.0021) on target. [0.9s]
Iteration 16, loss = 39.55571 [4.1s], (0.3444) on test, (0.0021, 0.0021, 0.0021) on target. [0.9s]
Iteration 17, loss = 38.39977 [3.1s], (0.3603) on test, (0.0043, 0.0043, 0.0043) on target. [0.9s]
Iteration 18, loss = 37.51890 [3.0s], (0.3775) on test, (0.0032, 0.0032, 0.0032) on target. [0.9s]
Iteration 19, loss = 36.85732 [3.1s], (0.3765) on test, (0.0032, 0.0032, 0.0032) on target. [0.9s]

```

```
Iteration 41, loss = 24.80048 [2.6s], (0.5376) on test, (0.0000, 0.0000, 0.0000) on target. [0.9s]
Iteration 42, loss = 24.46308 [2.3s], (0.5217) on test, (0.0000, 0.0000, 0.0000) on target. [0.9s]
Iteration 43, loss = 24.13988 [2.5s], (0.5387) on test, (0.0000, 0.0000, 0.0000) on target. [0.9s]
Iteration 44, loss = 23.79102 [2.9s], (0.5334) on test, (0.0000, 0.0000, 0.0000) on target. [0.9s]
Iteration 45, loss = 23.48067 [3.0s], (0.5355) on test, (0.0000, 0.0000, 0.0000) on target. [0.9s]
Iteration 46, loss = 23.18168 [3.0s], (0.5376) on test, (0.0000, 0.0000, 0.0000) on target. [0.9s]
Iteration 47, loss = 22.88689 [2.9s], (0.5483) on test, (0.0000, 0.0000, 0.0000) on target. [0.9s]
Iteration 48, loss = 22.60305 [2.9s], (0.5523) on test, (0.0000, 0.0000, 0.0000) on target. [0.9s]
Iteration 49, loss = 22.31694 [2.9s], (0.5493) on test, (0.0000, 0.0000, 0.0000) on target. [0.9s]
Iteration 50, loss = 22.05977 [2.9s], (0.5546) on test, (0.0000, 0.0000, 0.0000) on target. [0.9s]
Iteration 51, loss = 22.00500 [2.8s], (0.5525) on test, (0.9882, 0.9882, 0.9861) on target. [0.9s]
Iteration 52, loss = 22.11385 [2.3s], (0.5567) on test, (0.9893, 0.9893, 0.9889) on target. [0.9s]
Iteration 53, loss = 21.99215 [2.6s], (0.5525) on test, (0.9893, 0.9893, 0.9893) on target. [0.9s]
Iteration 54, loss = 21.75483 [2.7s], (0.5620) on test, (0.9893, 0.9893, 0.9893) on target. [0.9s]
Iteration 55, loss = 21.54424 [2.9s], (0.5578) on test, (0.9893, 0.9893, 0.9889) on target. [0.9s]
Iteration 56, loss = 21.33480 [3.0s], (0.5620) on test, (0.9893, 0.9904, 0.9896) on target. [0.9s]
Iteration 57, loss = 21.10641 [3.0s], (0.5599) on test, (0.9893, 0.9904, 0.9896) on target. [0.9s]
Iteration 58, loss = 20.90411 [3.0s], (0.5790) on test, (0.9893, 0.9893, 0.9893) on target. [1.0s]
Iteration 59, loss = 20.71460 [3.0s], (0.5620) on test, (0.9893, 0.9893, 0.9889) on target. [1.0s]
Iteration 60, loss = 20.53967 [3.0s], (0.5673) on test, (0.9893, 0.9893, 0.9889) on target. [1.0s]
Iteration 61, loss = 20.35701 [3.1s], (0.5684) on test, (0.9893, 0.9893, 0.9885) on target. [0.9s]
Iteration 62, loss = 20.18388 [3.1s], (0.5673) on test, (0.9893, 0.9893, 0.9885) on target. [0.9s]
Iteration 63, loss = 20.02223 [3.1s], (0.5663) on test, (0.9893, 0.9893, 0.9889) on target. [1.0s]
Iteration 64, loss = 19.88250 [3.4s], (0.5642) on test, (0.9893, 0.9893, 0.9889) on target. [0.9s]
Iteration 65, loss = 19.74218 [2.8s], (0.5726) on test, (0.9893, 0.9893, 0.9885) on target. [0.9s]
Iteration 66, loss = 19.57523 [2.6s], (0.5642) on test, (0.9893, 0.9893, 0.9885) on target. [0.9s]
```