CS 285 HW4

Cuiqianhe Du

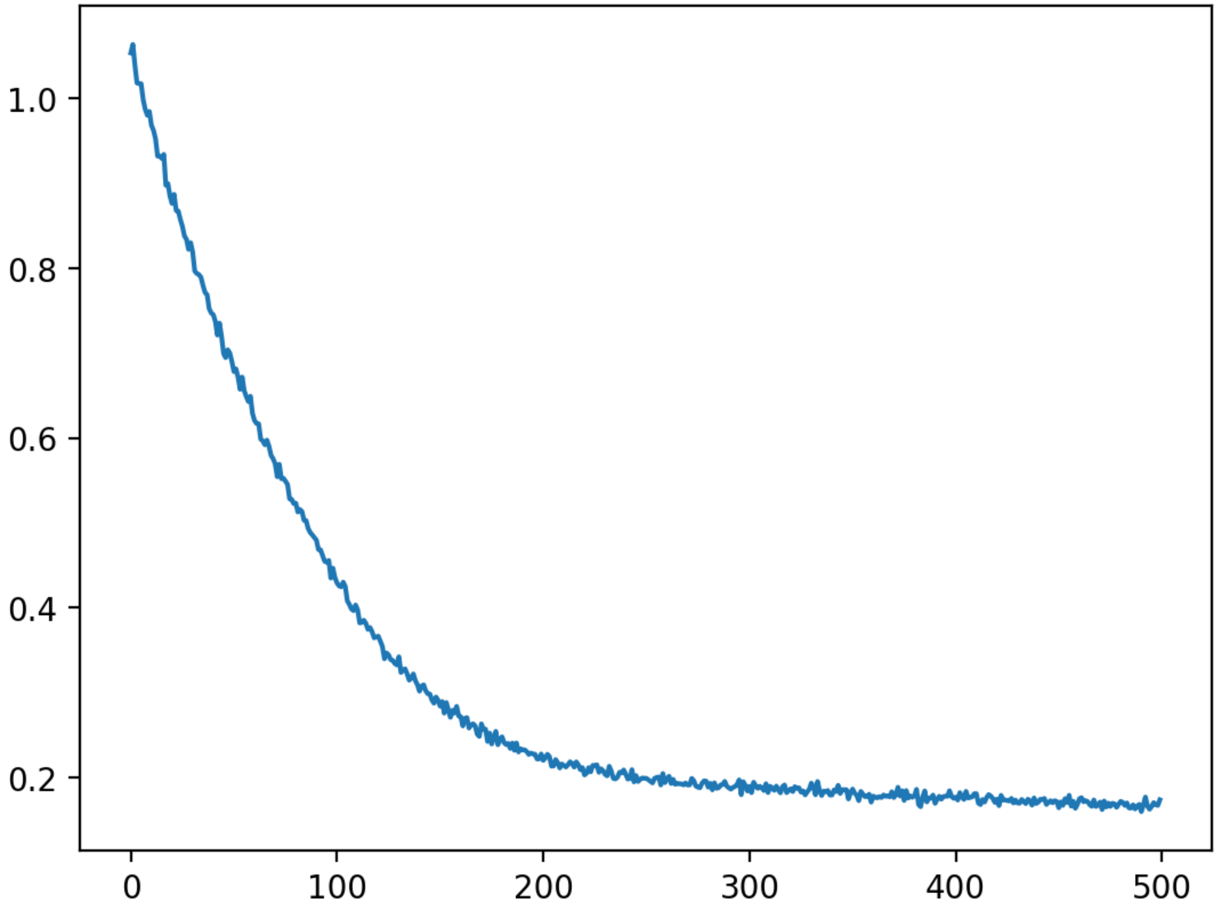
Question 1

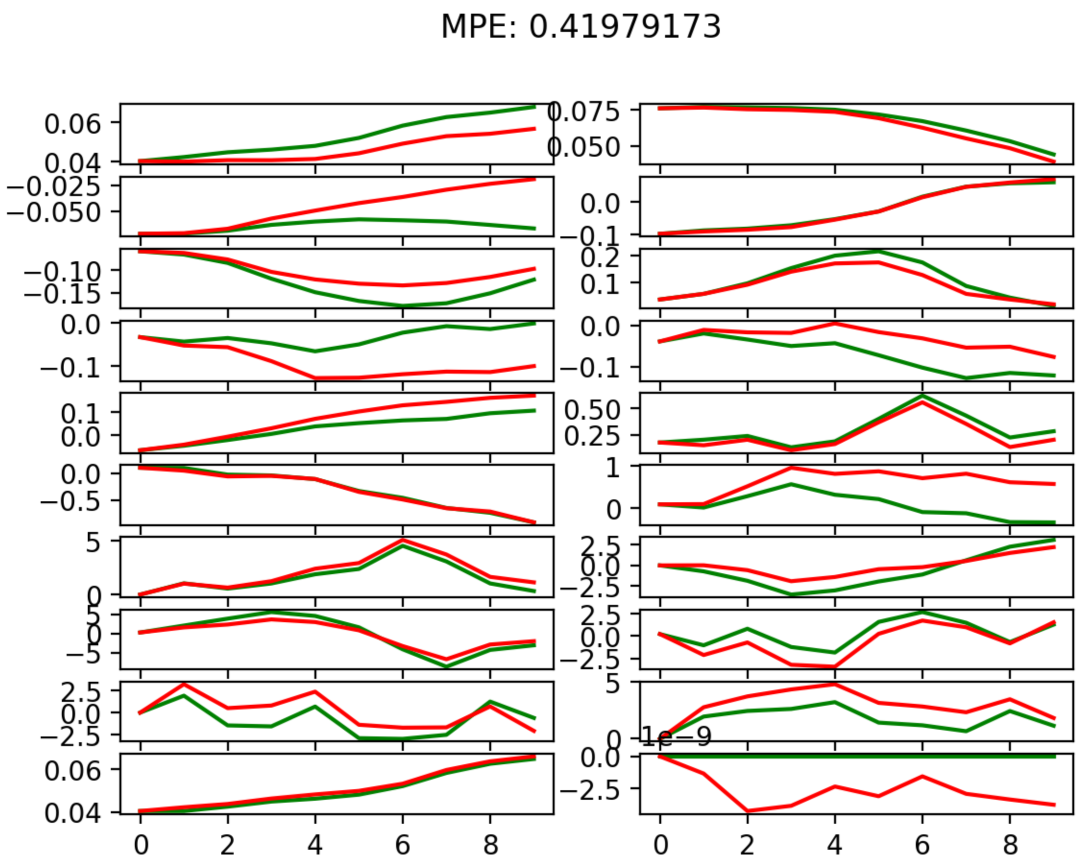
```
python cs285/scripts/run_hw4_mb.py --exp_name q1_cheetah_n500_arch1x32 --env_name cheetah-cs285-v0 --add_sl_noise --n_iter 1 --batch_size_initial 20000 --num_agent_train_steps_per_iter 500 --n_layers 1 --size 32 --scalar_log_freq -1 --video_log_freq -1 --mpc_action_sampling_strategy random
```

```
python cs285/scripts/run_hw4_mb.py --exp_name q1_cheetah_n5_arch2x250 --env_name cheetah-cs285-v0 --add_sl_noise --n_iter 1 --batch_size_initial 20000 --num_agent_train_steps_per_iter 5 --n_layers 2 --size 250 --scalar_log_freq -1 --video_log_freq -1 --mpc_action_sampling_strategy random
```

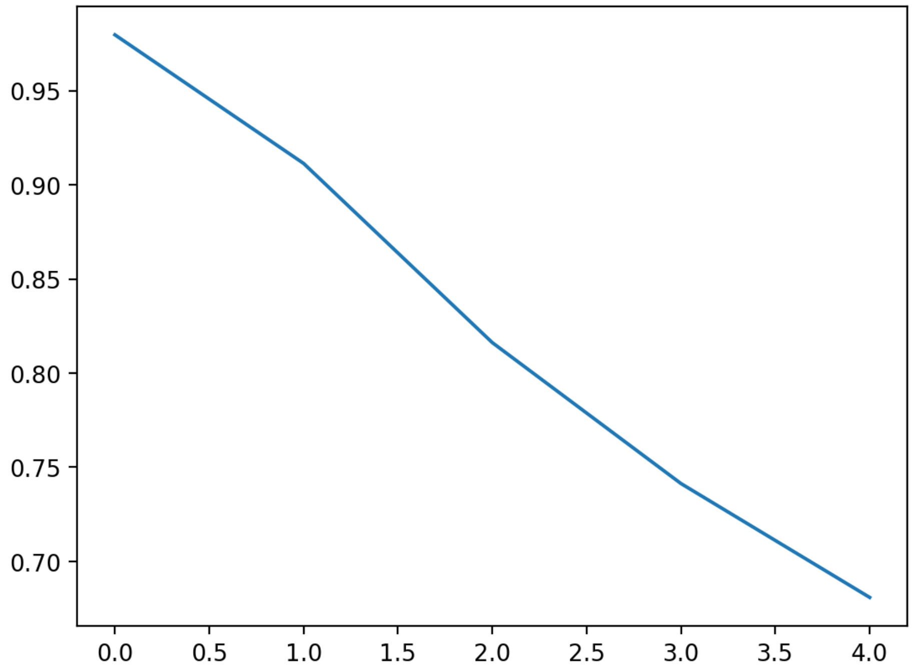
```
python cs285/scripts/run_hw4_mb.py --exp_name q1_cheetah_n500_arch2x250 --env_name cheetah-cs285-v0 --add_sl_noise --n_iter 1 --batch_size_initial 20000 --num_agent_train_steps_per_iter 500 --n_layers 2 --size 250 --scalar_log_freq -1 --video_log_freq -1 --mpc_action_sampling_strategy random
```

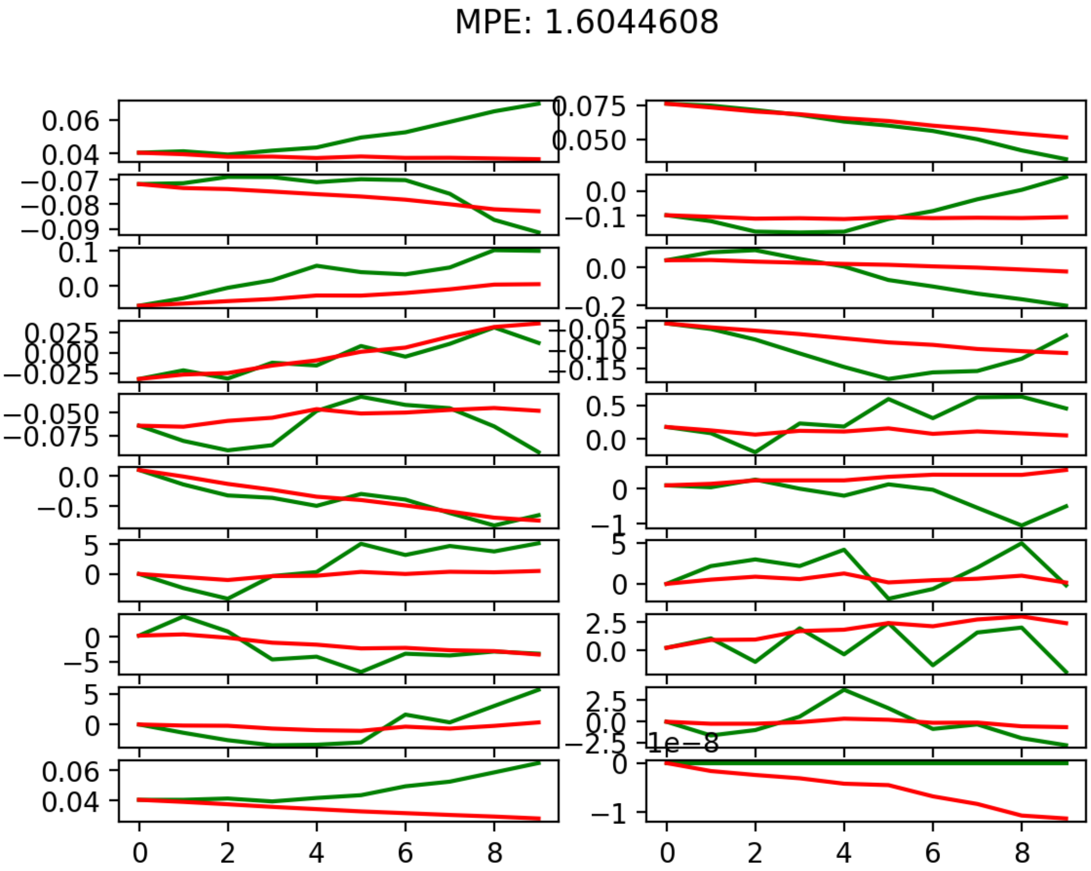
first run: q1_cheetah_n500_arch1x32



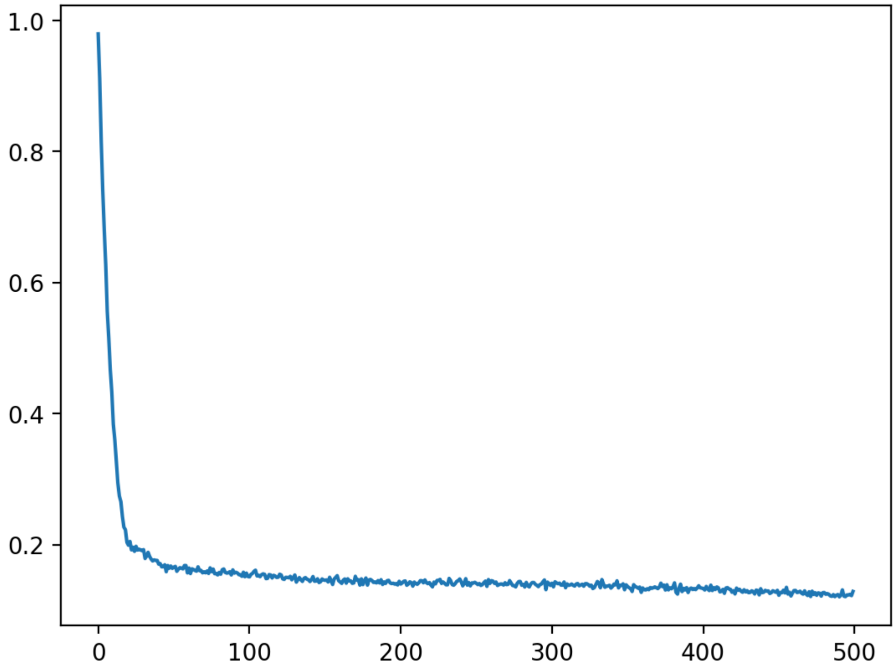


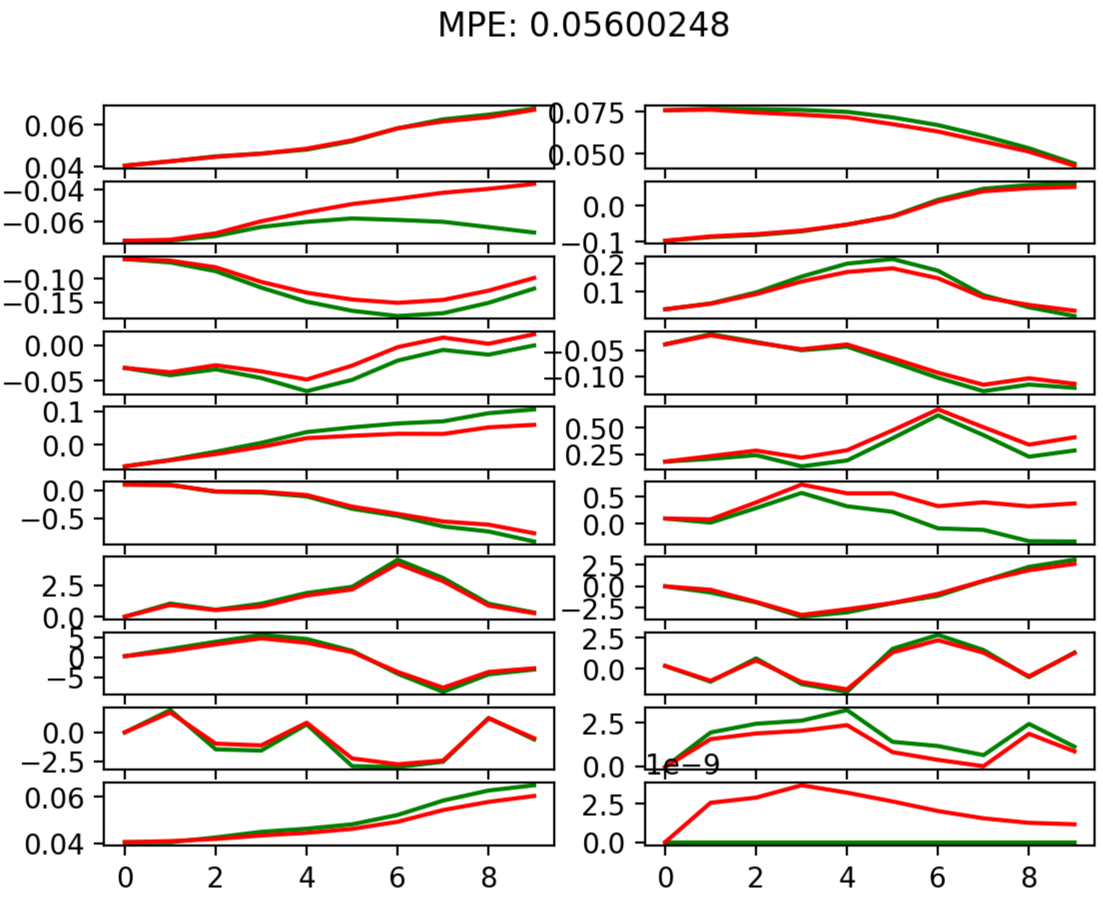
Second run: q1_cheetah_n5_arch2x250





Third run: q1_cheetah_n500_arch2x250





From previous three experiments, we can tell that:

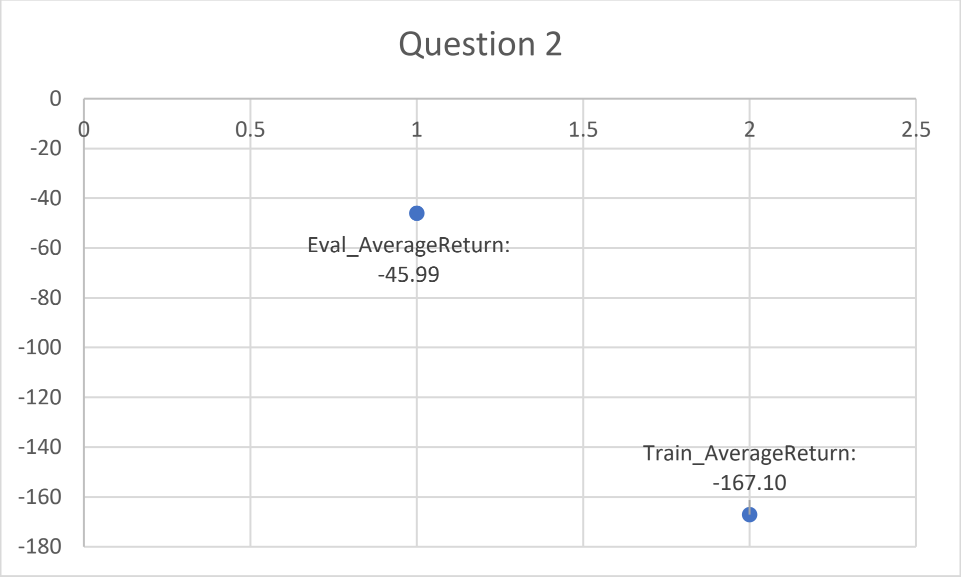
- The bigger the layer and the size, the better the performance (from experiment 1 and 3)
- The more the training step per iteration, the better the performance (from experiment 2 and 3)

Question 2

```
python cs285/scripts/run_hw4_mb.py --exp_name q2_obstacles_singleiteration --env_name obstacles-cs285-v0 --add_sl_noise --num_agent_train_steps_per_iter 20 --n_iter 1 --batch_size_initial 5000 --batch_size 1000 --mpc_horizon 10 --mpc_action_sampling_strategy random
```

Eval_AverageReturn : -45.98809814453125

Train_AverageReturn : -167.09857177734375

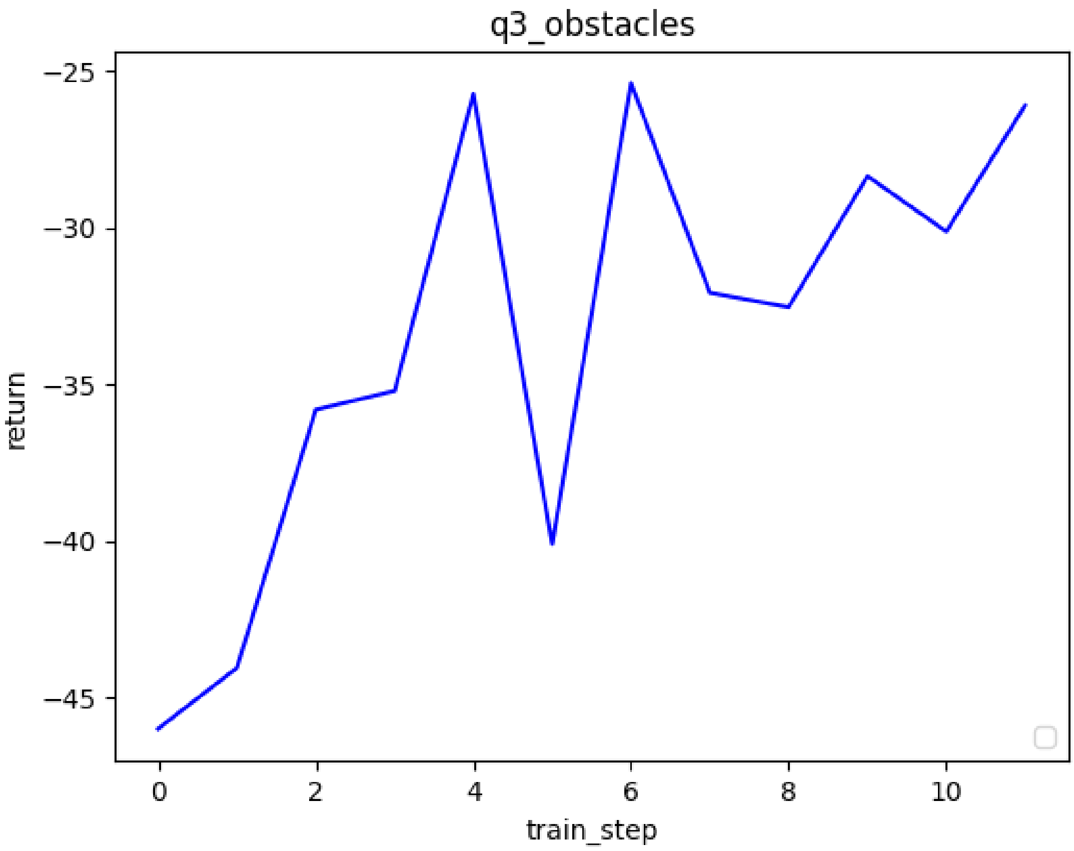


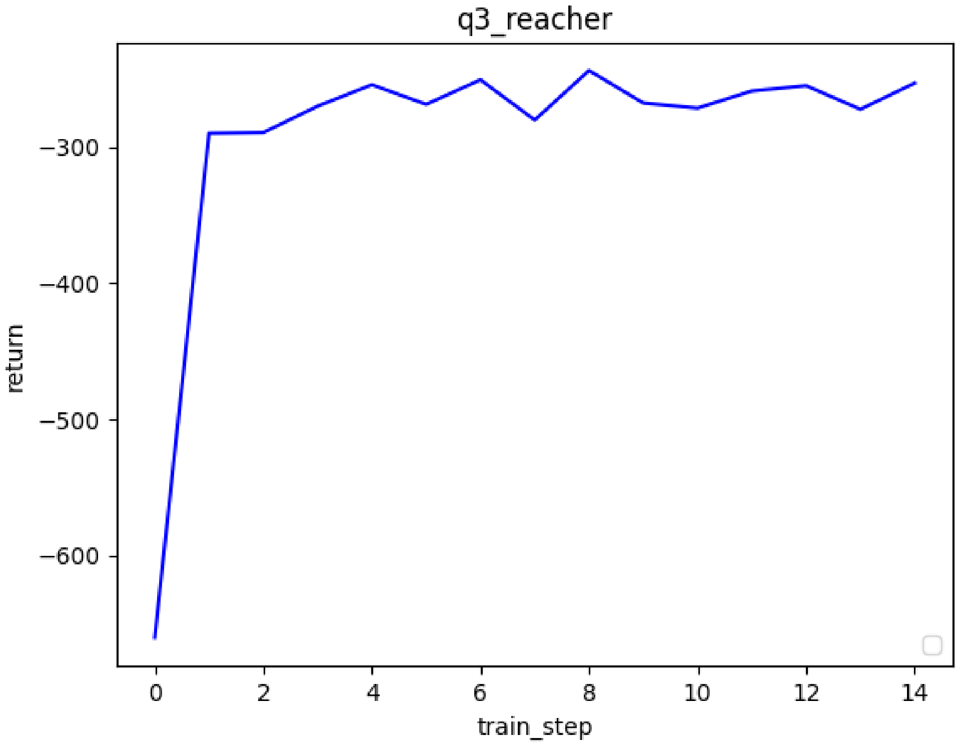
Question 3

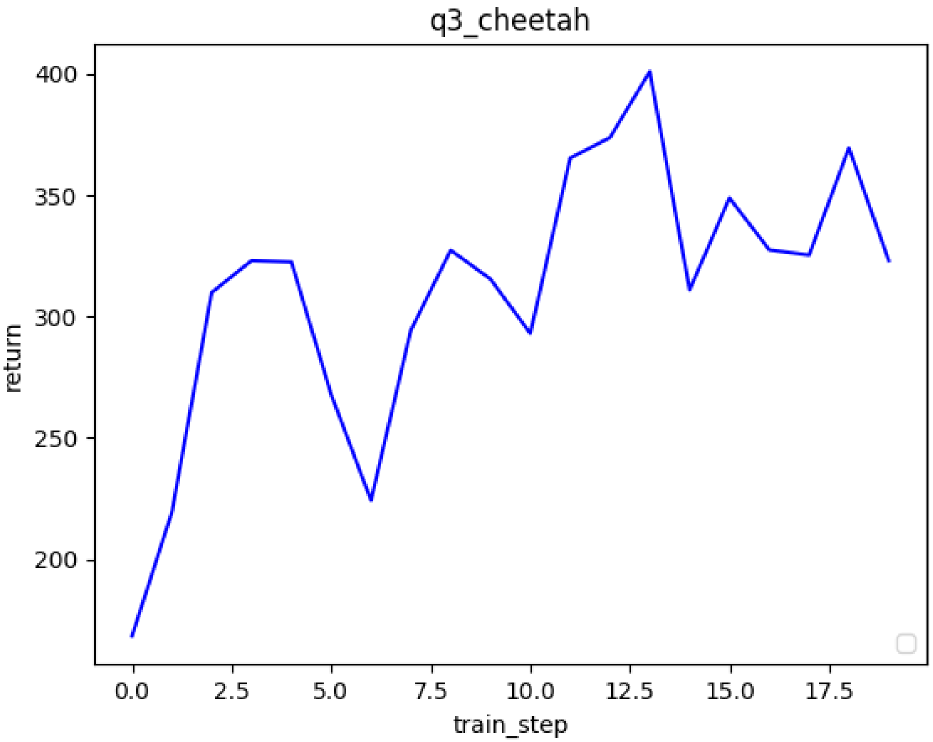
```
python cs285/scripts/run_hw4_mb.py --exp_name q3_obstacles --env_name obstacles-cs285-v0 --add_sl_noise --num_agent_train_steps_per_iter 20 --batch_size_initial 5000 --batch_size 1000 --mpc_horizon 10 --n_iter 12 --mpc_action_sampling_strategy random
```

```
python cs285/scripts/run_hw4_mb.py --exp_name q3_reacher --env_name reacher-cs285-v0 --add_sl_noise --mpc_horizon 10 --num_agent_train_steps_per_iter 1000 --batch_size_initial 5000 --batch_size 5000 --n_iter 15 --mpc_action_sampling_strategy random
```

```
python cs285/scripts/run_hw4_mb.py --exp_name q3_cheetah --env_name cheetah-cs285-v0 --mpc_horizon 15 --add_sl_noise --num_agent_train_steps_per_iter 1500 --batch_size_initial 5000 --batch_size 5000 --n_iter 20 --mpc_action_sampling_strategy random
```




q3_reacher
return
−300
−400
−500
−600
0
2
4
6
8
10
12
14
train_step


q3_cheetah
return
400
350
300
250
200
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
train_step

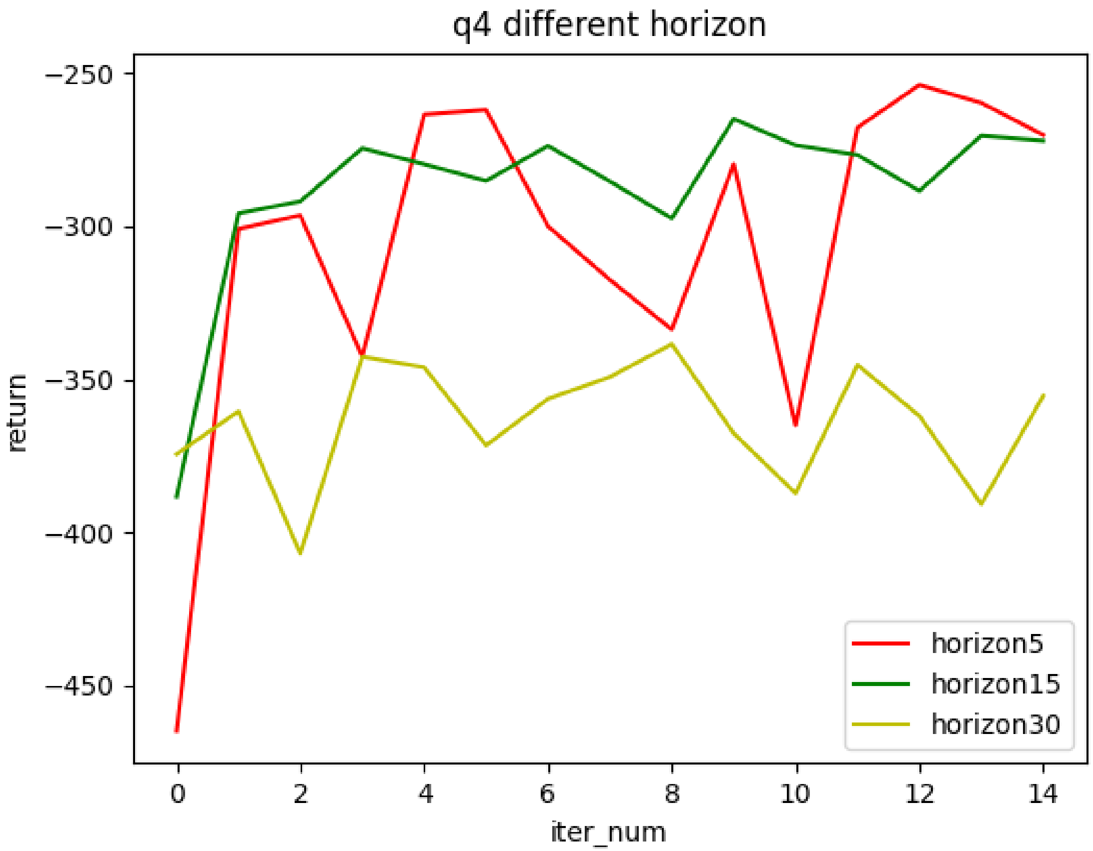
Question 4

```
python cs285/scripts/run_hw4_mb.py --exp_name q4_reacher_horizon5 --env_name reacher-cs285-v0 --add_sl_noise --mpc_horizon 5 --num_agent_train_steps_per_iter 1000 --batch_size 800 --n_iter 15 --mpc_action_sampling_strategy random
```

```
python cs285/scripts/run_hw4_mb.py --exp_name q4_reacher_horizon15 --env_name reacher-cs285-v0 --add_sl_noise --mpc_horizon 15 --num_agent_train_steps_per_iter 1000 --batch_size 800 --n_iter 15 --mpc_action_sampling_strategy random
```

```
python cs285/scripts/run_hw4_mb.py --exp_name q4_reacher_horizon30 --env_name reacher-cs285-v0 --add_sl_noise --mpc_horizon 30 --num_agent_train_steps_per_iter 1000 --batch_size 800 --n_iter 15 --mpc_action_sampling_strategy random
```

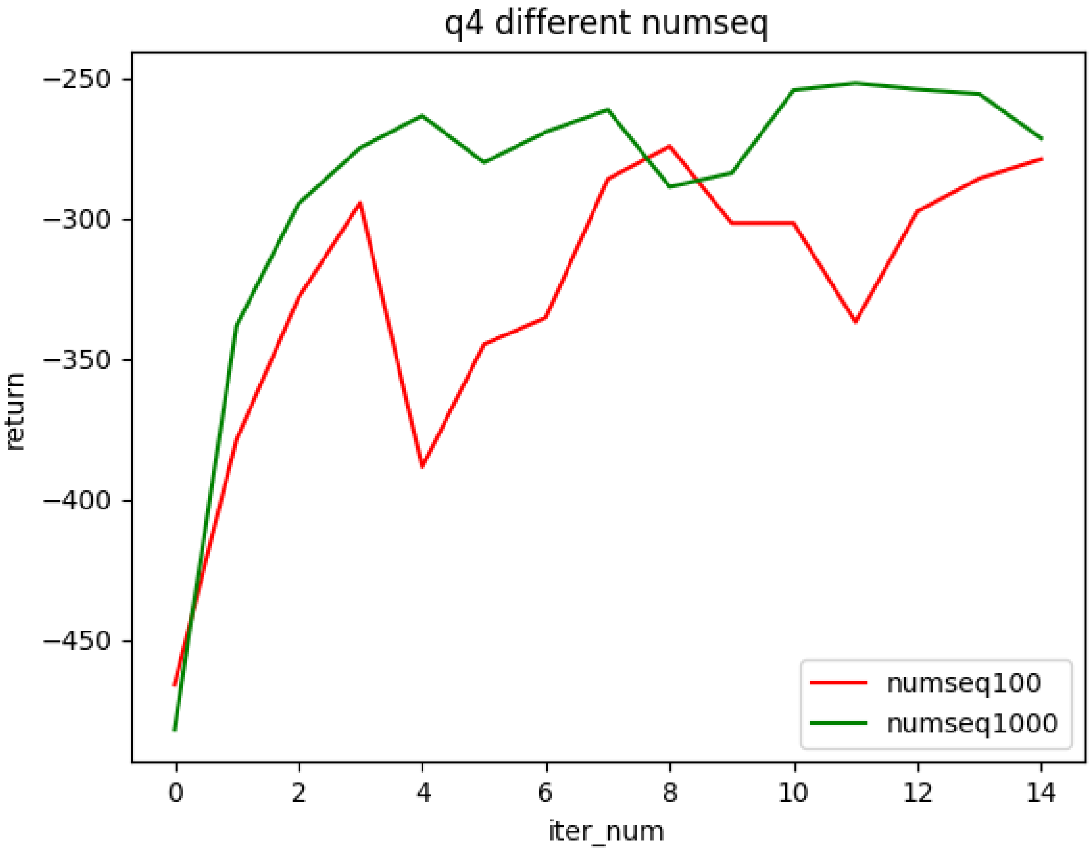
performance is the worst when planning horizon is 30, and even though 15 and 5 have similar output, performance is more stable when horizon is 15.



```
python cs285/scripts/run_hw4_mb.py --exp_name q4_reacher_numseq100 --env_name reacher-cs285-v0 --add_sl_noise --mpc_horizon 10 --num_agent_train_steps_per_iter 1000 --batch_size 800 --n_iter 15 --mpc_num_action_sequences 100 --mpc_action_sampling_strategy random
```

```
python cs285/scripts/run_hw4_mb.py --exp_name q4_reacher_numseq1000 --env_name reacher-cs285-v0 --add_sl_noise --mpc_horizon 10 --num_agent_train_steps_per_iter 1000 --batch_size 800 --n_iter 15 --mpc_num_action_sequences 1000 --mpc_action_sampling_strategy random
```

Larger number of candidate action sequences have better performance

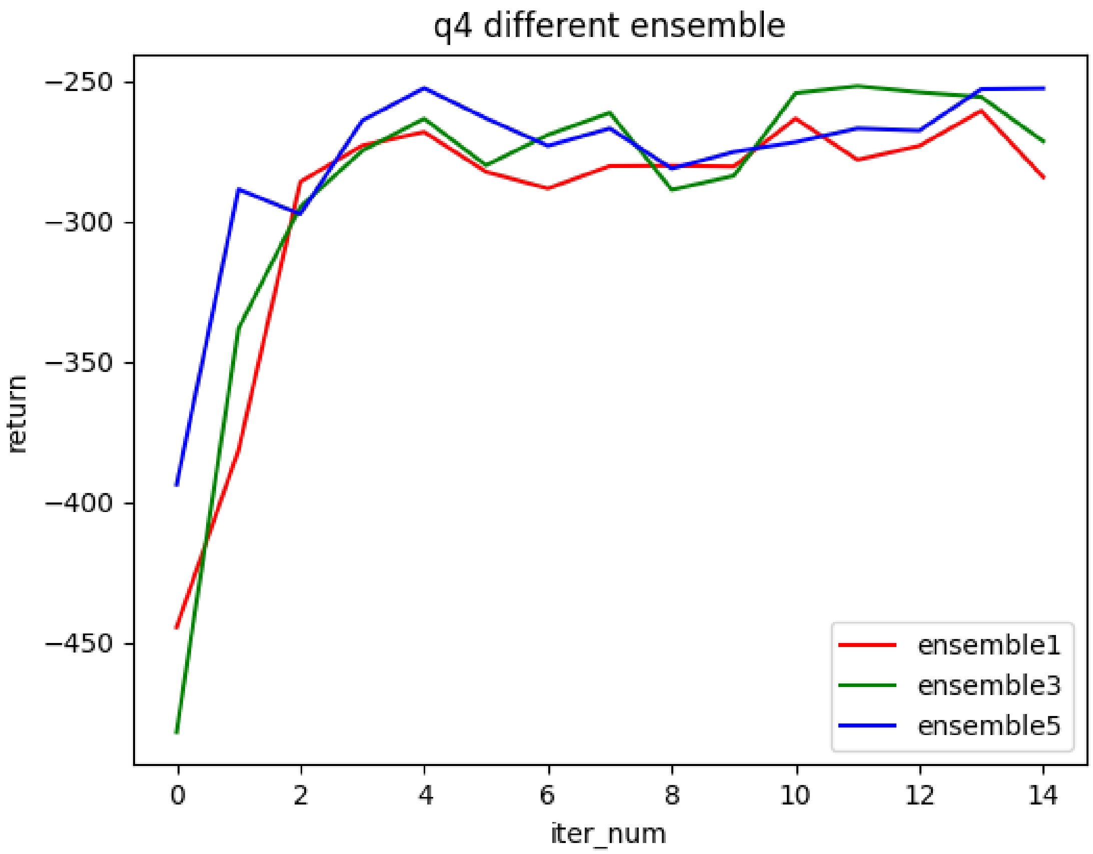


```
python cs285/scripts/run_hw4_mb.py --exp_name q4_reacher_ensemble1 --env_name reacher-cs285-v0 --ensemble_size 1 --add_sl_noise --mpc_horizon 10 --num_agent_train_steps_per_iter 1000 --batch_size 800 --n_iter 15 --mpc_action_sampling_strategy random
```

```
python cs285/scripts/run_hw4_mb.py --exp_name q4_reacher_ensemble3 --env_name reacher-cs285-v0 --ensemble_size 3 --add_sl_noise --mpc_horizon 10 --num_agent_train_steps_per_iter 1000 --batch_size 800 --n_iter 15 --mpc_action_sampling_strategy random
```

```
python cs285/scripts/run_hw4_mb.py --exp_name q4_reacher_ensemble5 --env_name reacher-cs285-v0 --ensemble_size 5 --add_sl_noise --mpc_horizon 10 --num_agent_train_steps_per_iter 1000 --batch_size 800 --n_iter 15 --mpc_action_sampling_strategy random
```

Larger ensemble size has better and more stable performance



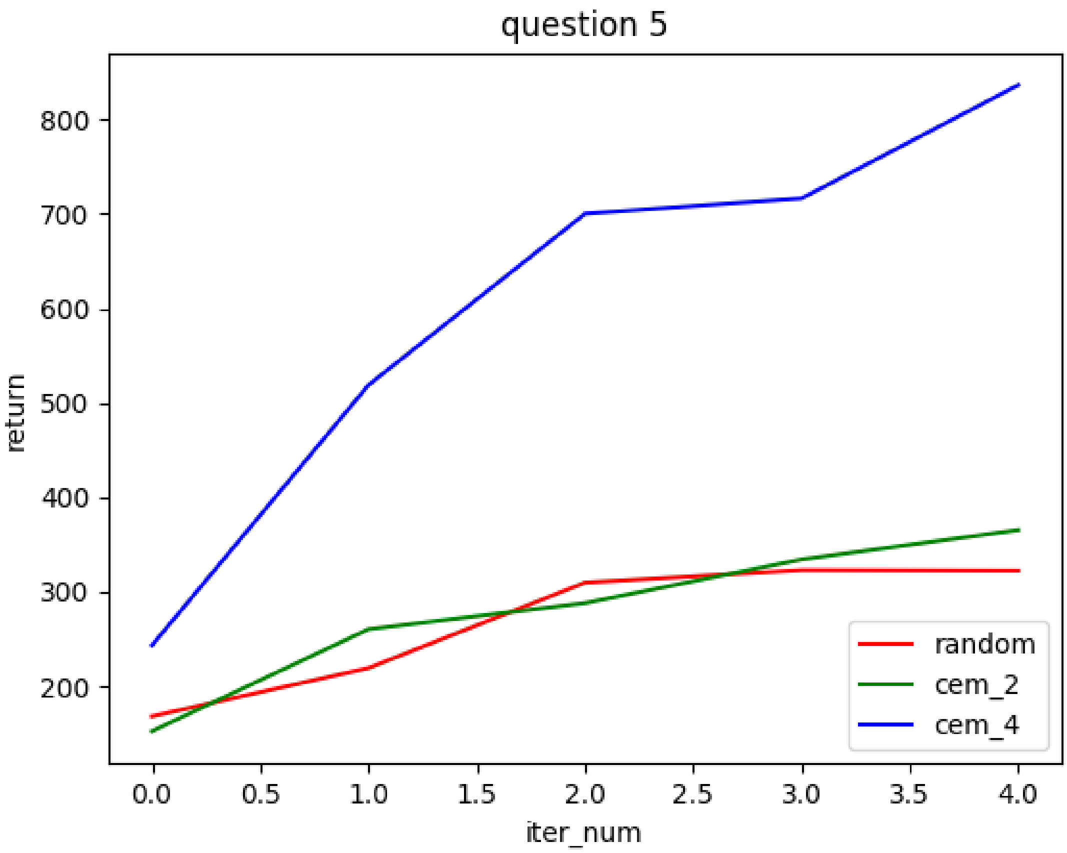
Question 5

```
python cs285/scripts/run_hw4_mb.py --exp_name q5_cheetah_random --env_name cheetah-cs285-v0 --mpc_horizon 15 --add_sl_noise --num_agent_train_steps_per_iter 1500 --batch_size_initial 5000 --batch_size 5000 --n_iter 5 --mpc_action_sampling_strategy random
```

```
python cs285/scripts/run_hw4_mb.py --exp_name q5_cheetah_cem_2 --env_name cheetah-cs285-v0 --mpc_horizon 15 --add_sl_noise --num_agent_train_steps_per_iter 1500 --batch_size_initial 5000 --batch_size 5000 --n_iter 5 --mpc_action_sampling_strategy cem --cem_iterations 2
```

```
python cs285/scripts/run_hw4_mb.py --exp_name q5_cheetah_cem_4 --env_name cheetah-cs285-v0 --mpc_horizon 15 --add_sl_noise --num_agent_train_steps_per_iter 1500 --batch_size_initial 5000 --batch_size 5000 --n_iter 5 --mpc_action_sampling_strategy cem --cem_iterations 4
```

CEM method with 4 iterations have best performance compared to CEM with 2 iterations and random shooting method


question 5
return
iter_num
random
cem_2
cem_4