Homework 5

Author:
Reece D. Huff

CS 285: Deep Reinforcement Learning

Dr. Sergey Levine

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

Part 1

Subpart 1

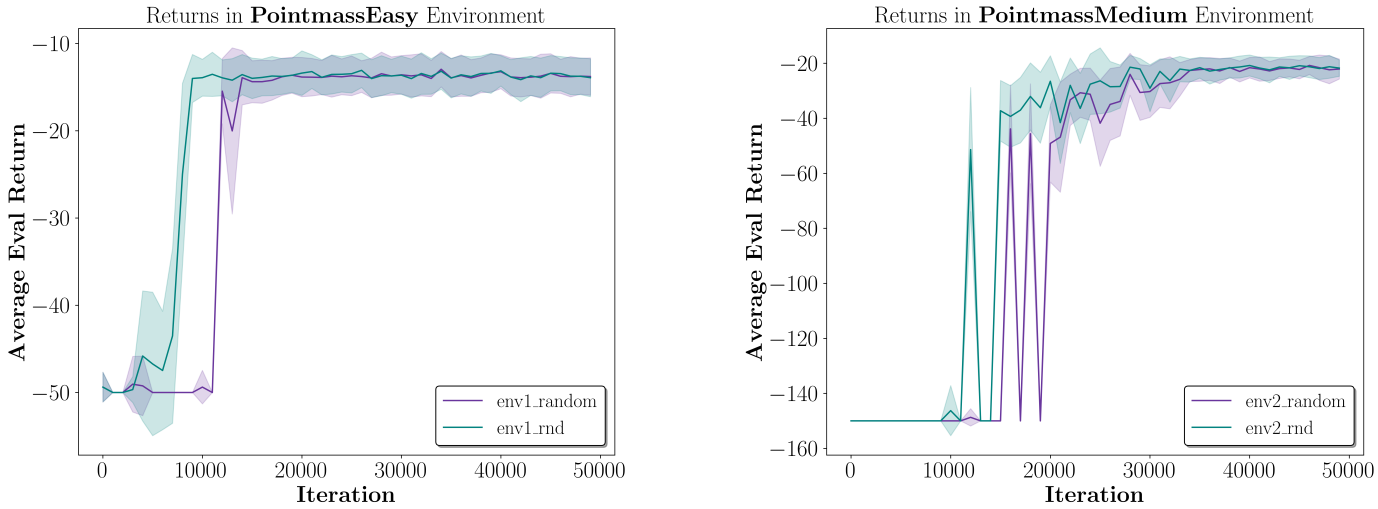
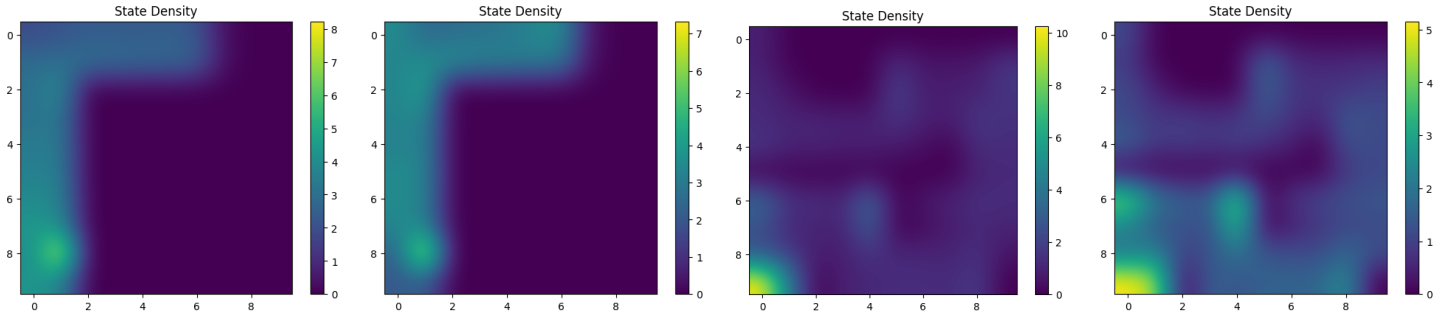


Figure 1: Comparing the performance of RND and random in the Easy and Medium point mass environments.



(a) Random in easy environment.

(b) RND in easy environment.

(c) Random in medium environment.

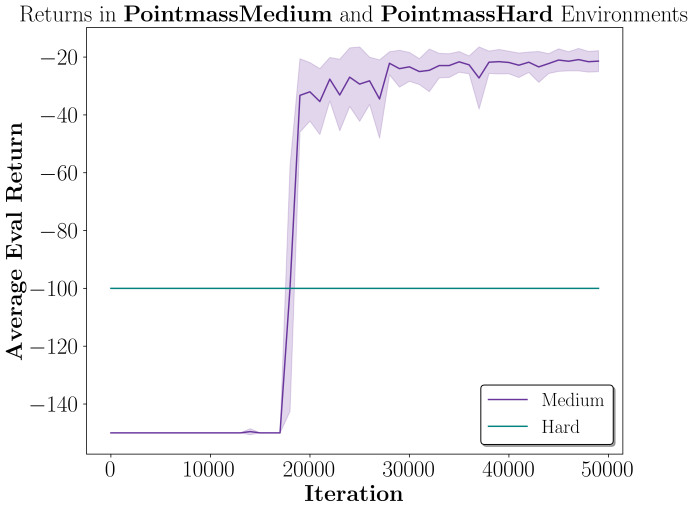
(d) RND in medium environment.

Figure 2: State density plots of RND and random in the Easy and Medium point mass environments.

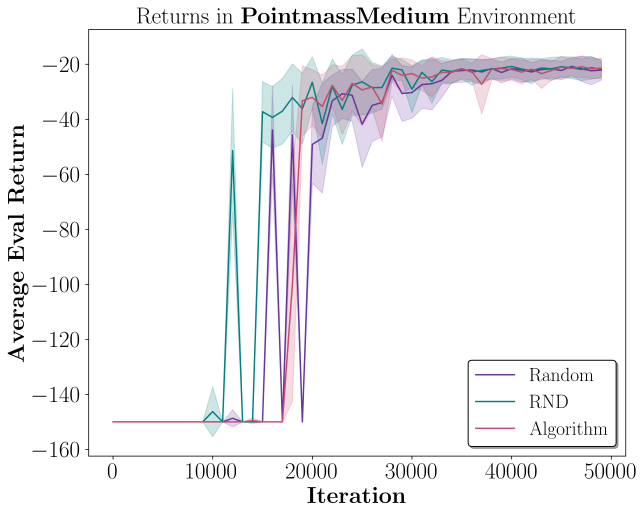
part1_part1.sh

```
python cs285/scripts/run_hw5_expl.py --env_name PointmassEasy-v0 --use_rnd --
    unsupervised_exploration --exp_name q1_env1_rnd

python cs285/scripts/run_hw5_expl.py --env_name PointmassEasy-v0 --
    unsupervised_exploration --exp_name q1_env1_random

python cs285/scripts/run_hw5_expl.py --env_name PointmassMedium-v0 --use_rnd --
    unsupervised_exploration --exp_name q1_env2_rnd

python cs285/scripts/run_hw5_expl.py --env_name PointmassMedium-v0 --
    unsupervised_exploration --exp_name q1_env2_random
```

Subpart 2



(a) Performance of my custom algorithm in the medium and hard environment.



(b) Performance of RND, random, and my algorithm in the medium environment. $\alpha = 0$ corresponds to DQN.

Figure 3: I implemented a new algorithm that defines the error similar to RND, but the error is the ℓ_2 distance between the current state, s_t and the new state, s_{t+1}. In theory this would provide an exploration bonus to transition that make large new steps, thus avoiding small steps that get stuck near the wall or the starting point. My algorithm outperforms the random (epsilon-greedy) exploration policy in the medium environment, but fails in the hard environment.

part1_part2.sh

```
python cs285/scripts/run_hw5_expl.py --env_name PointmassMedium-v0 --
   unsupervised_exploration --use_min_dist --exp_name q1_alg_med

python cs285/scripts/run_hw5_expl.py --env_name PointmassHard-v0 --
   unsupervised_exploration --use_min_dist --exp_name q1_alg_hard
```

Part 2

Subpart 1

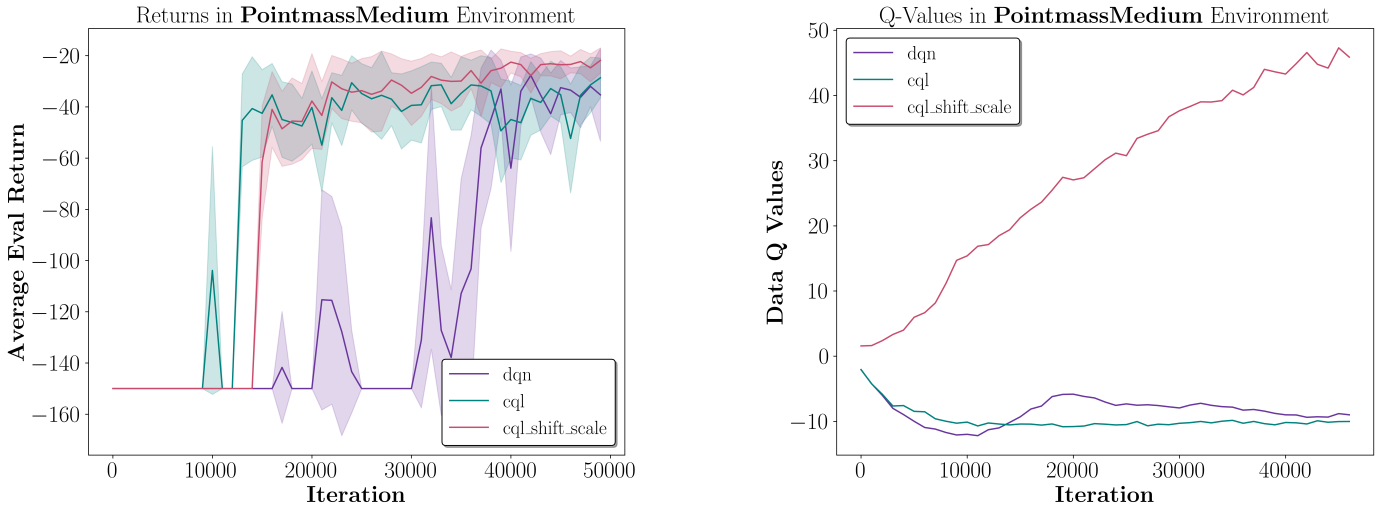


Figure 4: The average return (left) and learned Q values (right) in the medium environment for DQN, CQL, and CQL with shifted and scaled reward.

Comments: We note a couple things in the learning curves above. On the left, we see that the CQL outperforms DQN. The CQL with the shifted and scaled reward is in fact the best performer. In the curve on the right, we note the learned Q-values are slightly higher for DQN but smaller for standard CQL. When we shift and scale the reward, the learned Q-values are much larger, perhaps helping the algorithm in its actual performance. Shifting the rewards could essentially lead to penalizing poor performing trajectories with negative rewards while incentivizing high performing trajectories with positive rewards.

part2_part1.sh

```
# cql_alpha = 0 => DQN, cql_alpha = 0.1 => CQL

python cs285/scripts/run_hw5_expl.py --env_name PointmassMedium-v0 --exp_name q2_dqn
    --use_rnd --unsupervised_exploration --offline_exploitation --cql_alpha=0

python cs285/scripts/run_hw5_expl.py --env_name PointmassMedium-v0 --exp_name q2_cql
    --use_rnd --unsupervised_exploration --offline_exploitation --cql_alpha=0.1

python cs285/scripts/run_hw5_expl.py --env_name PointmassMedium-v0 --exp_name
    q2_cql_shift_scale --use_rnd --unsupervised_exploration --offline_exploitation --
    cql_alpha=0.1 --exploit_rew_shift=1 --exploit_rew_scale=100
```

Subpart 2

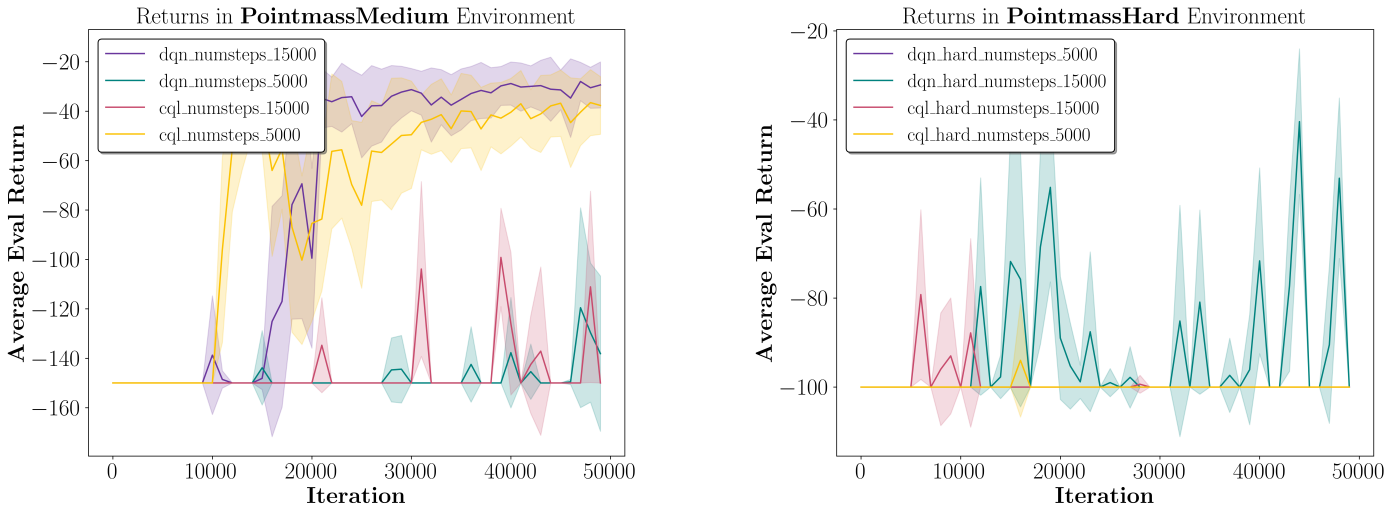


Figure 5: Performance of DQN and CQL in the medium (left) and hard (right) environment as the number of environment steps is varied. **Note:** The assignment calls for a table when comparing the performances as a function of the number of steps but I find a plot of reward over time more insightful.

part2_part2.sh

```
python cs285/scripts/run_hw5_expl.py --env_name PointmassMedium-v0 --use_rnd --
    num_exploration_steps=5000 --offline_exploitation --cql_alpha=0.1 --
    unsupervised_exploration --exp_name q2_cql_numsteps_5000
python cs285/scripts/run_hw5_expl.py --env_name PointmassMedium-v0 --use_rnd --
    num_exploration_steps=15000 --offline_exploitation --cql_alpha=0.1 --
    unsupervised_exploration --exp_name q2_cql_numsteps_15000

python cs285/scripts/run_hw5_expl.py --env_name PointmassMedium-v0 --use_rnd --
    num_exploration_steps=5000 --offline_exploitation --cql_alpha=0.0 --
    unsupervised_exploration --exp_name q2_dqn_numsteps_5000
python cs285/scripts/run_hw5_expl.py --env_name PointmassMedium-v0 --use_rnd --
    num_exploration_steps=15000 --offline_exploitation --cql_alpha=0.0 --
    unsupervised_exploration --exp_name q2_dqn_numsteps_15000

python cs285/scripts/run_hw5_expl.py --env_name PointmassHard-v0 --use_rnd --
    num_exploration_steps=5000 --offline_exploitation --cql_alpha=0.1 --
    unsupervised_exploration --exp_name q2_cql_hard_numsteps_5000
python cs285/scripts/run_hw5_expl.py --env_name PointmassHard-v0 --use_rnd --
    num_exploration_steps=15000 --offline_exploitation --cql_alpha=0.1 --
    unsupervised_exploration --exp_name q2_cql_hard_numsteps_15000

python cs285/scripts/run_hw5_expl.py --env_name PointmassHard-v0 --use_rnd --
    num_exploration_steps=5000 --offline_exploitation --cql_alpha=0.0 --
    unsupervised_exploration --exp_name q2_dqn_hard_numsteps_5000
python cs285/scripts/run_hw5_expl.py --env_name PointmassHard-v0 --use_rnd --
    num_exploration_steps=15000 --offline_exploitation --cql_alpha=0.0 --
    unsupervised_exploration --exp_name q2_dqn_hard_numsteps_15000
```

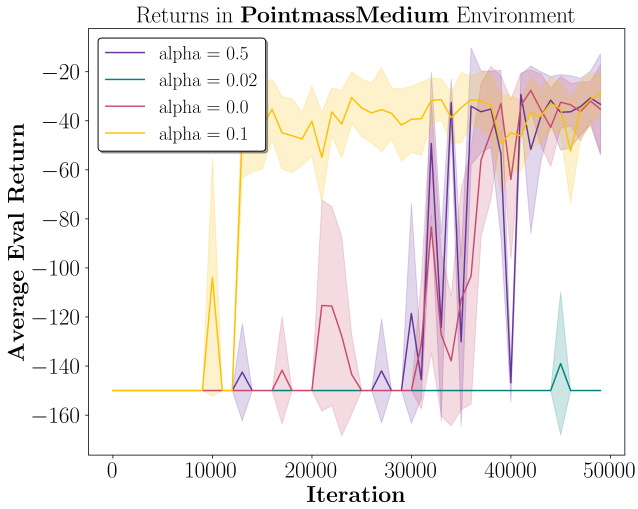


Figure 6: Performance of CQL as the CQL α hyperparameter is varied.

Subpart 3

Comments: There is a large variation in the performance as the α is varied. The highest performer is α = 0.1. DQN and CQL when $\alpha = 0.5$ are similar performers. Interestingly, a small α such as 0.02 greatly decrease the performance. I am not sure why the small α decreases performance so greatly; I would expect it to perform similarly to DQN.

part2_part3.sh

```
for a in 0.02 0.5
do
    python cs285/scripts/run_hw5_expl.py --env_name PointmassMedium-v0 --use_rnd --
   unsupervised_exploration --offline_exploitation --cql_alpha=$a --exp_name
   q2_alpha$a
done
```

Part 3

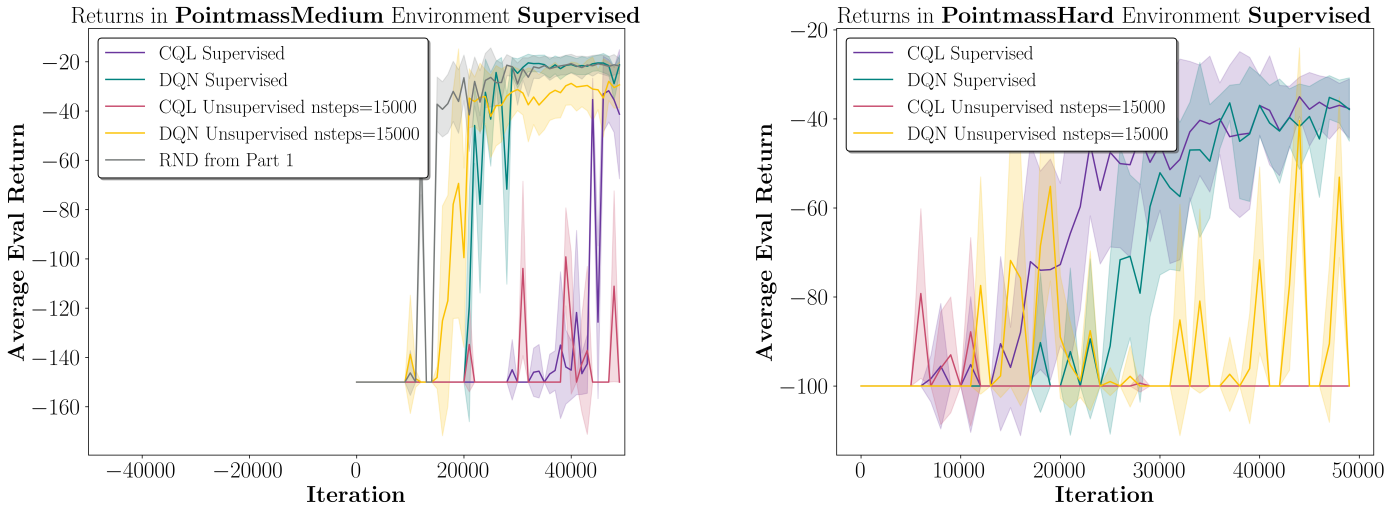


Figure 7: Performance in the medium (left) and hard (right) environment across all of our algorithms thus far. **Note:** I did not shift and scale the reward for CQL in the plot on the left, and I did not run the commands from part 1 in the hard environment so we cannot make a comparison.

Comments:

In the medium environment, we see that DQN and run from part 1 have similar performance. Additionally, the supervise DQN does quite well. CQL struggles, but it is important to note that I did not shift and scale the rewards for these runs. If I did, we would expect CQL to perform much better (see Part 2 Subpart 1).

In the hard environment, we start to see the importance of supervision. Supervised CQL (now with shifted and scaled rewards) is the best performer follow by supervised DQN. When supervision is not present, the algorithm fails to achieve high returns. While I did not run RND from part 1 in the hard environment, I would expect that it would similar returns as the unsupervised DQN and CQL. It makes sense that supervision will help because we finetune our algorithm with a mix of environment and exploration rewards.

`part3.sh`

```
python cs285/scripts/run_hw5_expl.py --env_name PointmassMedium-v0 --use_rnd --
    num_exploration_steps=20000 --cql_alpha=0.0 --exp_name q3_medium_dqn

python cs285/scripts/run_hw5_expl.py --env_name PointmassMedium-v0 --use_rnd --
    num_exploration_steps=20000 --cql_alpha=1.0 --exp_name q3_medium_cql


python cs285/scripts/run_hw5_expl.py --env_name PointmassHard-v0 --use_rnd --
    num_exploration_steps=20000 --cql_alpha=0.0 --exp_name q3_hard_dqn --
    exploit_rew_shift=1 --exploit_rew_scale=100

python cs285/scripts/run_hw5_expl.py --env_name PointmassHard-v0 --use_rnd --
    num_exploration_steps=20000 --cql_alpha=1.0 --exp_name q3_hard_cql --
    exploit_rew_shift=1 --exploit_rew_scale=100
```

Part 4

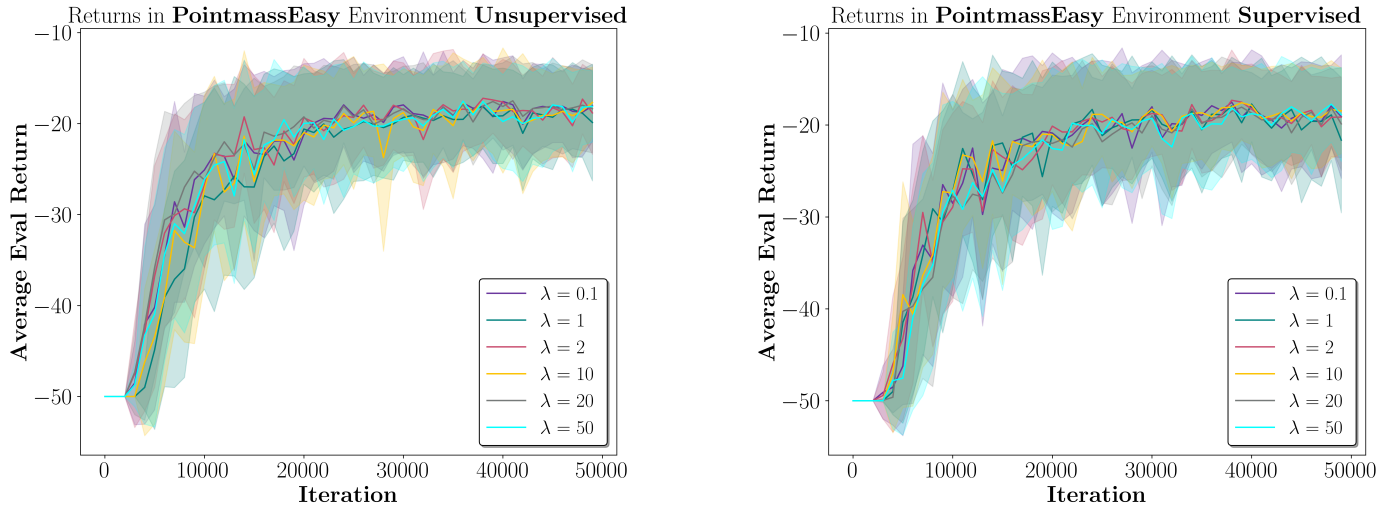


Figure 8: Performance in the **easy** environment without supervision (left) and with supervision (right).

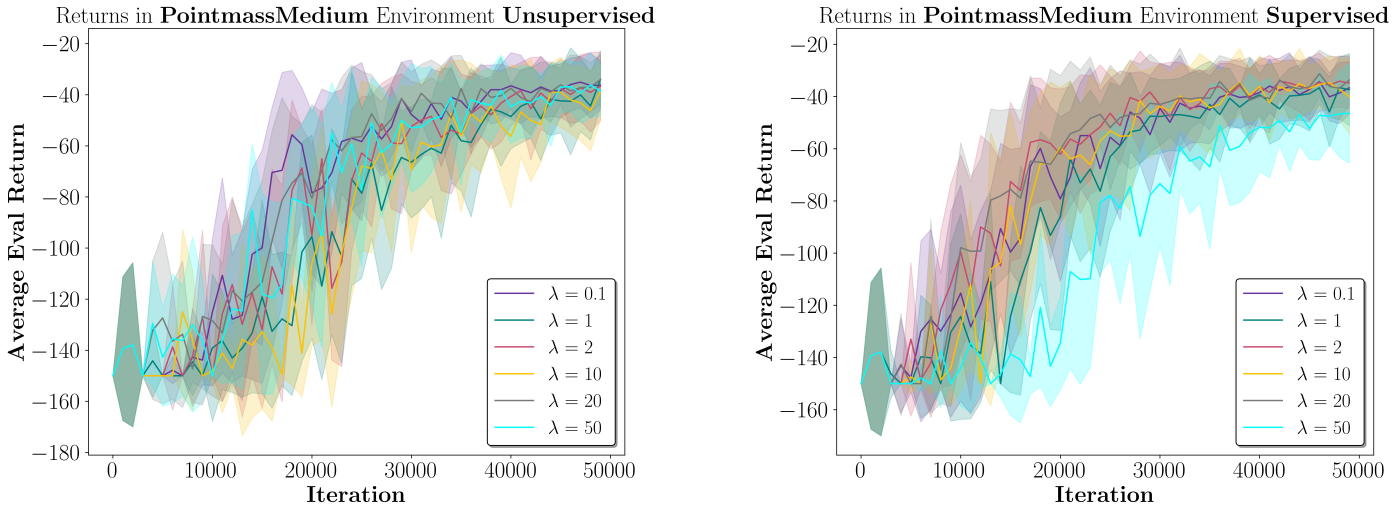


Figure 9: Performance in the **medium** environment without supervision (left) and with supervision (right).

Comments:

In the easy environment, λ had little to no effect on the performance regardless of the presence of supervision or not, perhaps because the easy environment is a rather easy environment in the first place. In part 5, I used an optimal λ of 10 in the easy environment.

In the medium environment, λ had a greater impact on performance. Many of the λ's achieve high performance, but some of them (e.g., $\lambda = 50$) result in much lower performance. In part 5, I used an optimal λ of 20 in the medium environment.

`part4.sh`

```
for lambda in 0.1 1 2 10 20 50
do
```

```
    python cs285/scripts/run_hw5_awac.py --env_name PointmassEasy-v0 --exp_name
   q4_awac_easy_unsupervised_lam$lambda --use_rnd --num_exploration_steps=20000 --
   unsupervised_exploration --awac_lambda=$lambda

    python cs285/scripts/run_hw5_awac.py --env_name PointmassEasy-v0 --use_rnd --
   num_exploration_steps=20000 --awac_lambda=$lambda --exp_name
   q4_awac_easy_supervised_lam$lambda

done

for lambda in 0.1 1 2 10 20 50
do

    python cs285/scripts/run_hw5_awac.py --env_name PointmassMedium-v0 --exp_name
   q4_awac_medium_unsupervised_lam$lambda --use_rnd --num_exploration_steps=20000 --
   unsupervised_exploration --awac_lambda=$lambda

    python cs285/scripts/run_hw5_awac.py --env_name PointmassMedium-v0 --use_rnd --
   num_exploration_steps=20000 --awac_lambda=$lambda --exp_name
   q4_awac_medium_supervised_lam$lambda

done
```

Part 5

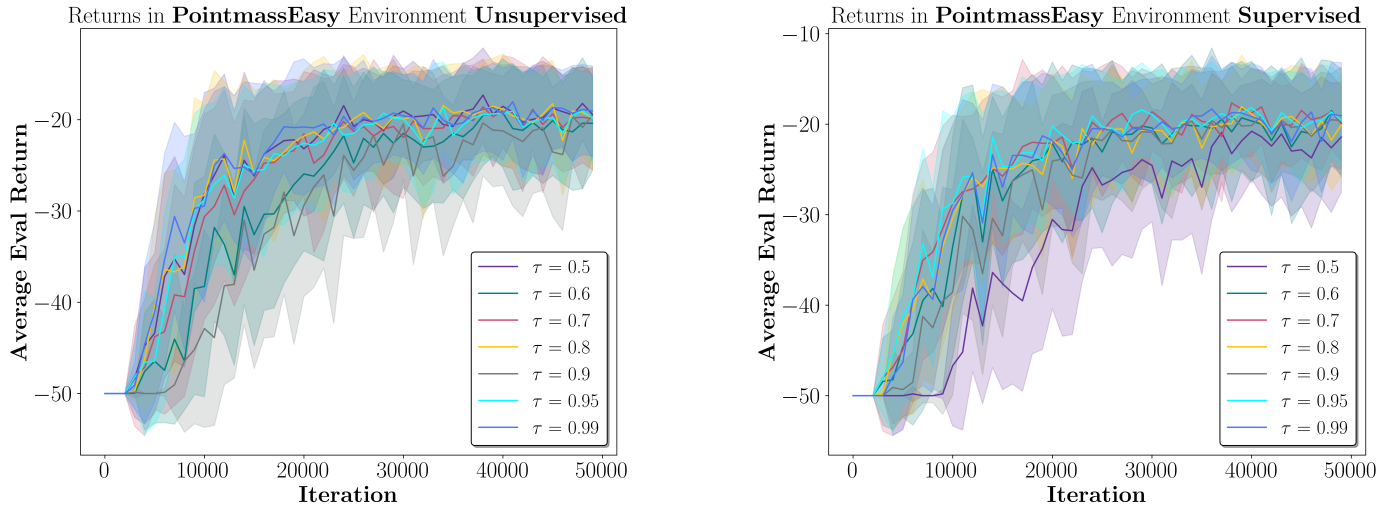


Figure 10: Performance in the **easy** environment without supervision (left) and with supervision (right).

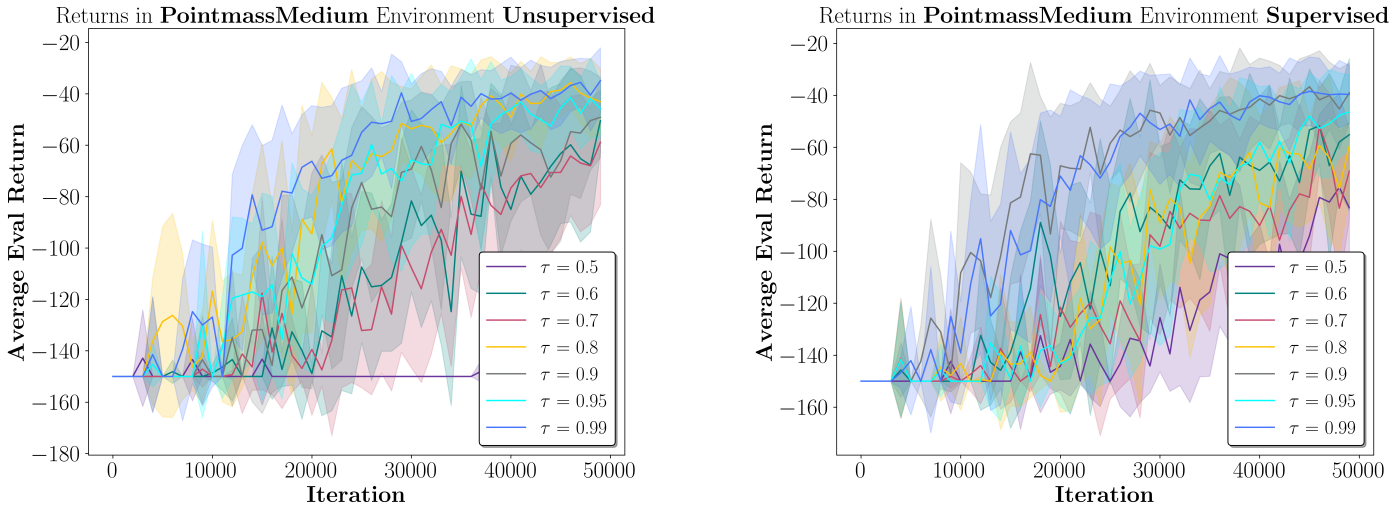


Figure 11: Performance in the **medium** environment without supervision (left) and with supervision (right).

Comments:

In the easy environment, τ had a small effect on the performance regardless of the presence of supervision or not. The easy environment is rather easy in the first place, so perhaps this is unsurprising.

In the medium environment, τ had a greater impact on performance. Only a couple of the τ's achieve high performance, and some of them (e.g., $\tau = 0.5$) result in much lower performance. In comparison below, I used an optimal τ of 0.9.

`part5.sh`

```
for tau in 0.5 0.6 0.7 0.8 0.9 0.95 0.99
do
```

```
    python cs285/scripts/run_hw5_iql.py --env_name PointmassEasy-v0 --exp_name
    q5_iql_easy_supervised_lam10_tau${tau} --use_rnd --num_exploration_steps=20000 --
    awac_lambda=10 --iql_expectile=${tau}

    python cs285/scripts/run_hw5_iql.py --env_name PointmassEasy-v0 --exp_name
    q5_iql_easy_unsupervised_lam10_tau${tau} --use_rnd --unsupervised_exploration --
    num_exploration_steps=20000 --awac_lambda=10 --iql_expectile=${tau}

    python cs285/scripts/run_hw5_iql.py --env_name PointmassMedium-v0 --exp_name
    q5_iql_medium_supervised_lam20_tau${tau} --use_rnd --num_exploration_steps=20000
    --awac_lambda=20 --iql_expectile=${tau}

    python cs285/scripts/run_hw5_iql.py --env_name PointmassMedium-v0 --exp_name
    q5_iql_medium_unsupervised_lam20_tau${tau} --use_rnd --unsupervised_exploration --
    num_exploration_steps=20000 --awac_lambda=20 --iql_expectile=${tau}

done
```

Comparing CQL, AWAC, and IQL

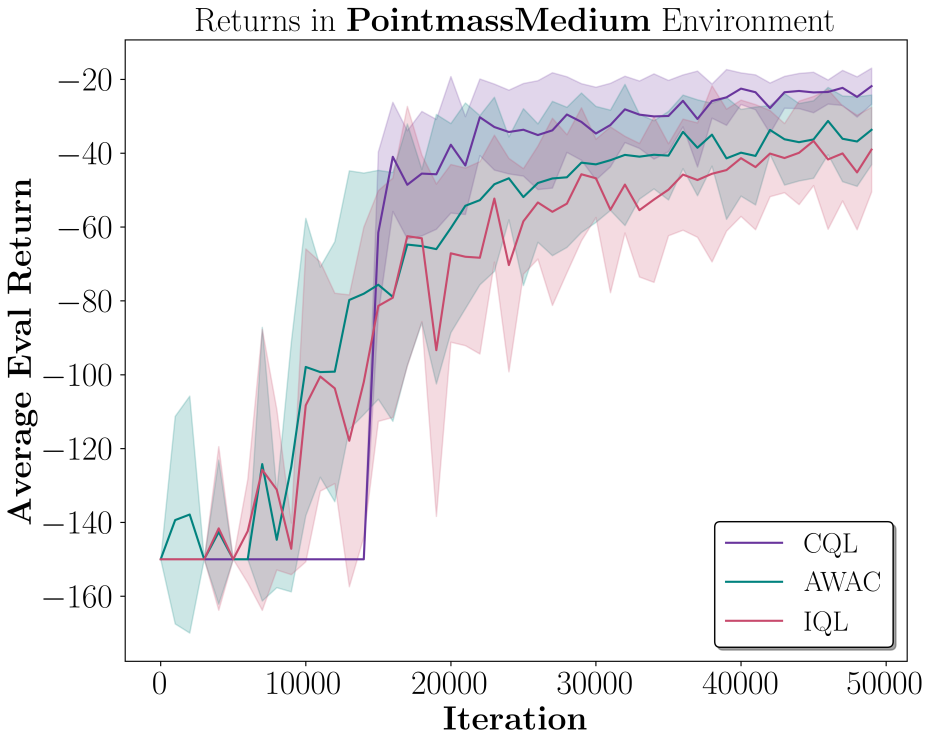


Figure 12: Comparing CQL, AWAC, and IQL performance in the medium environment. It is important to note that this is CQL with the shifted and scaled rewards, whereas the other two did not have shifted and scaled rewards. Additionally, I used the optimal hyperparameters for AWAC and IQL ($\lambda = 20$ and $\tau = 0.9$, respectively).