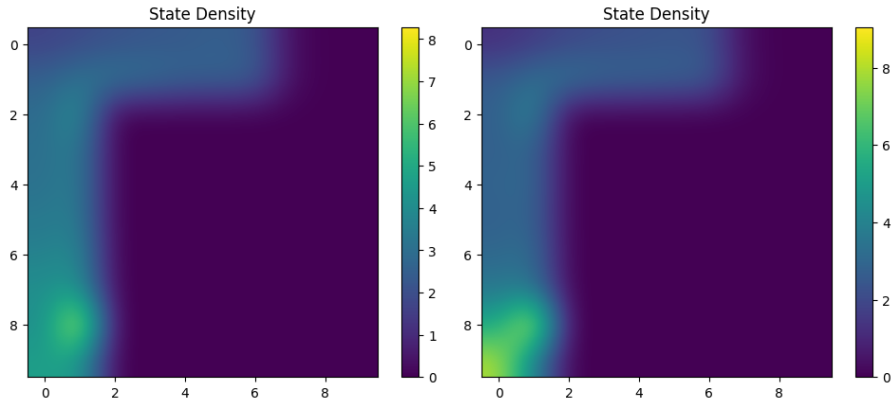
Homework 5

Juanwu Lu (3037432593)

(M.Sc. Civil Engineering, UC Berkeley)

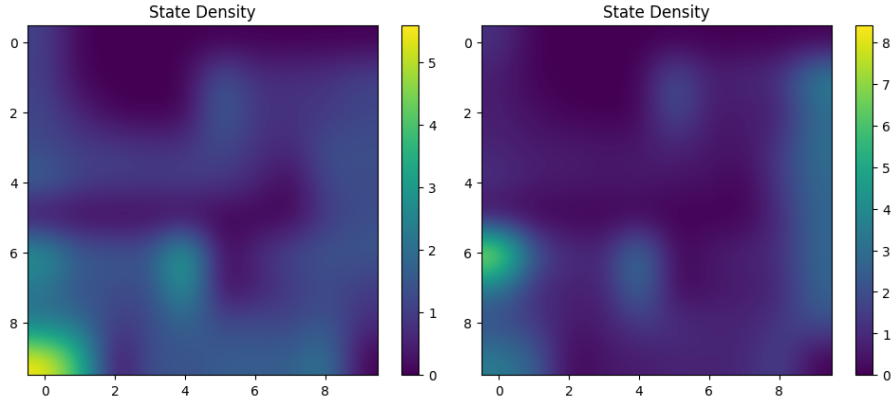
Problem 1 "Unsupervised" RND and exploration performance

Part 1 Results



(a) (b)

Figure 1. Results from PointmassEasy Environment: (a) Random exploration with epsilon-greedy; (b) Exploration with RND. State density of from the two algorithms are similar but RND unexpectedly has a denser density around the lower left corner (origin).



(a) (b)

Figure 2. Results from PointmassMedium Environment: (a) Random exploration with epsilon-greedy; (b) Exploration with RND. State density of from the two algorithms are similar but RND has a more uniformly distributed density than the random exploration one.

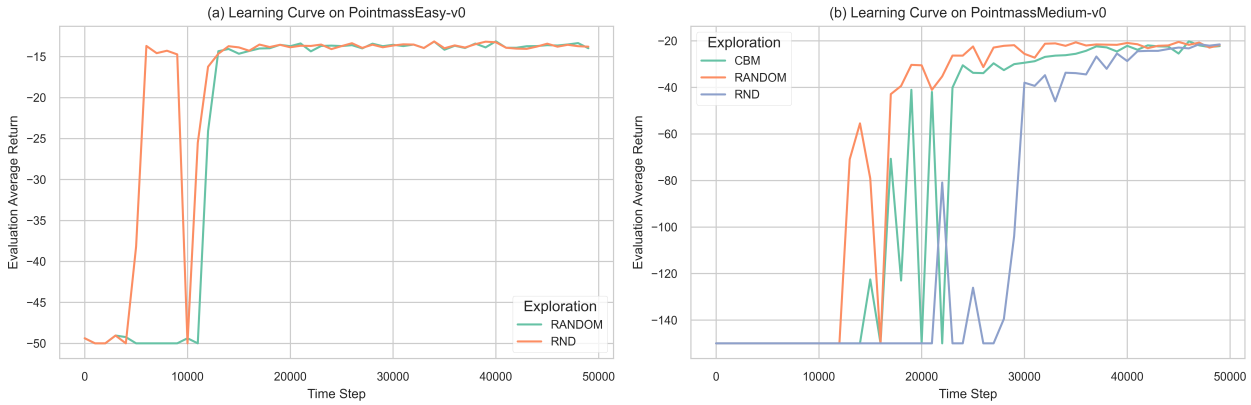


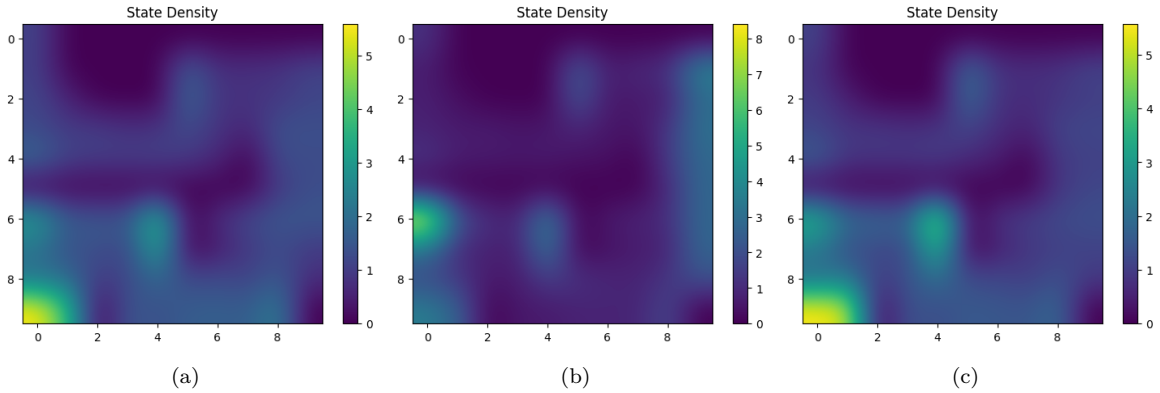
Figure 3. Learning curve from the two environments. RND reaches higher score faster than epsilon-greedy on the PointmassEasy environment, while slower on the PointmassMedium environment.

Part 2 Results

Since the state space is a grid world. I implemented the UCB-like count-based exploration (CBM), where

$$r_{\text{explore}}(s) = N(s)^{-\frac{1}{2}} \tag{1}$$

As shown in Figure 3, DQN with CBM learns faster than the RND, but slower than the epsilon-greedy. From the figures below, I think this is partly because it's exploring more states than the epsilon-greedy algorithm.



(a) (b) (c)

Figure 4. Results from PointmassMedium Environment: (a) Random exploration with epsilon-greedy; (b) Exploration with RND; (c) Exploration with count-based Model. State density from epsilon-greed and count-based exploration strategies are quite similar, while the one from RND-based exploration is more uniformly distributed.

Problem 2 Offline learning on exploration data

Part 1 Compare CQL to DQN on the `PointmassMedium` environment

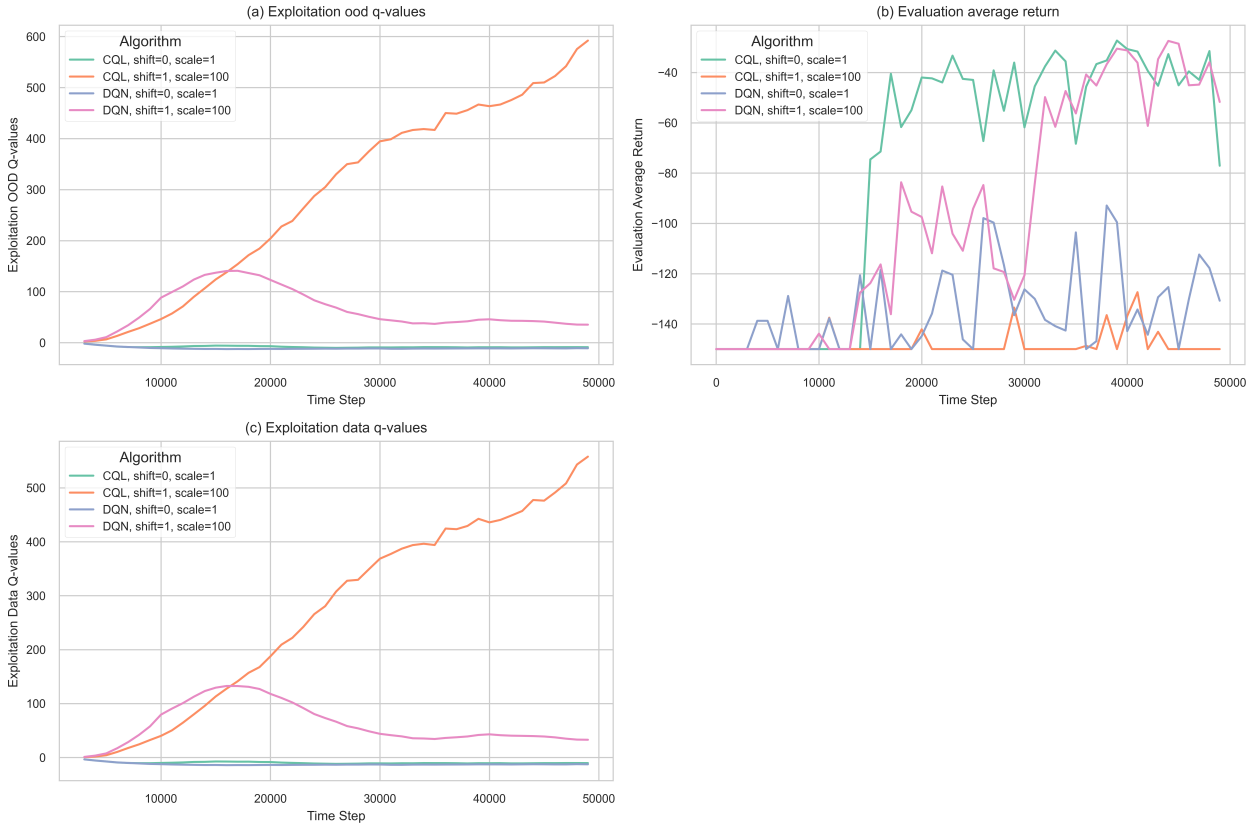


Figure 5. (a) OOD Q-value estimation; (b) Evaluation Average Returns; and (c) Buffer Q-value estimation on DQN- and CQL-based algorithm with/without reward scale and shifts. My experiment results shows that CQL works better with the original reward with only penalties of -1, while DQN works better with scale and shifted rewards. With the original reward settings, CQL helps to prevent overestimation of OOD Q values, while it fails with scaled and shifted rewards. My explanation to this is that shifting and scaling with 1 and 100 causes rewards to be more sparse (i.e., only getting 100 when success while getting 0 at other places). This mitigate overestimation of OOD Q values for DQN, which helps it perform better.

Part 2 Ablation study on amount of exploration data

TABLE 1. Final average evaluation return of DQN- and CQL-based exploration with different exploration steps. Surprisingly, my experiment results indicate that increasing the number of unsupervised exploration steps in return deteriorate the learning performance. And DQN-based algorithm worsen much more significantly than the CQL-based one. My explanation for this is that unsupervised exploration can actually generate a series of out-of-distribution data. Given sparse rewards, it actually makes it more difficult to learn from the exploration data. And DQN can overestimate the Q values of some OOD data, which cause a even worse performance.

Number of Exploration Steps	DQN	CQL
5000	-36.75	-30.73
15000	-127.4	-75.33

Part 3 Ablation study on the regularizer weight α

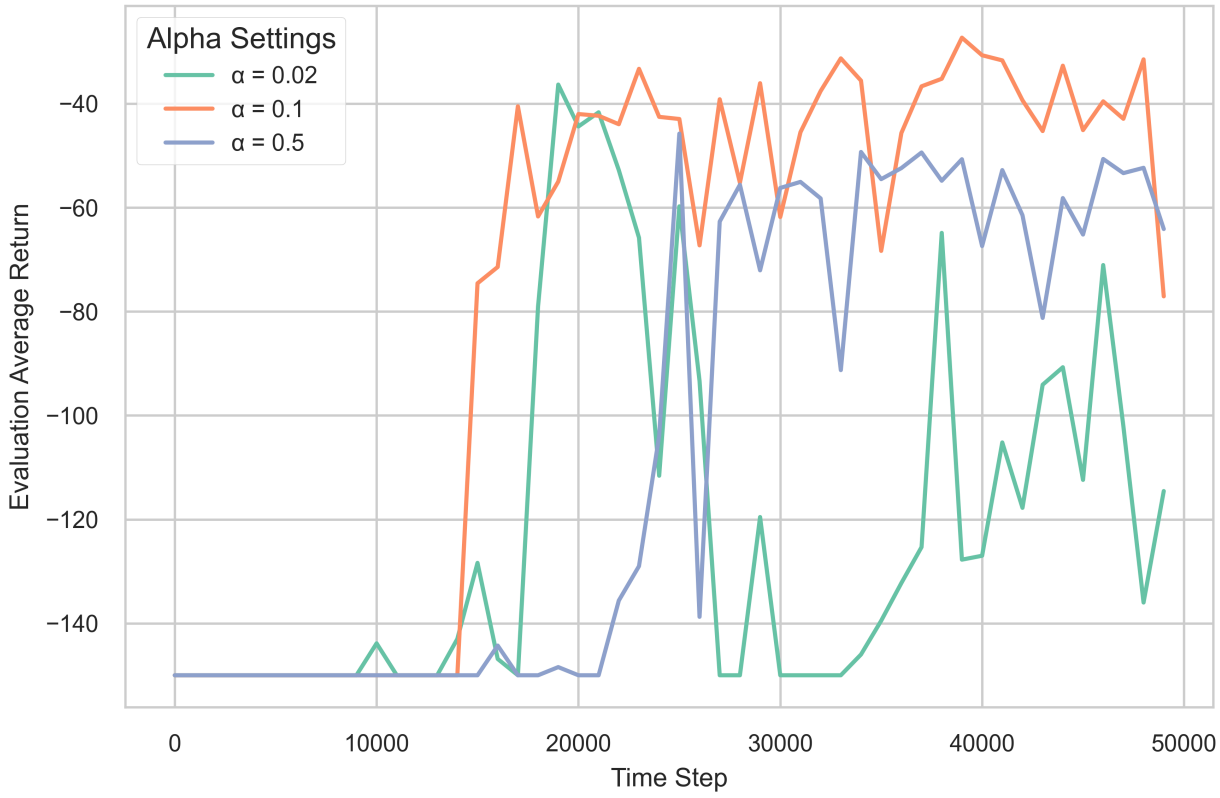


Figure 6. Learning curve of CQL-based algorithm on `PointmassMedium` with different α settings. From its formulation, higher value of α promotes stronger constraints on overestimation of the OOD q-value, which results in a more conservative. Therefore, the figure shows that with smaller α value, the algorithm tends to reach higher score earlier, but finds it hard to maintain the achievement since it tends to overestimate some OOD states. On the contrary, a higher α value would cause a longer time to train, but the learning curve is more stable at the end.

Problem 3 ”Supervised” exploration with mixed reward bonuses

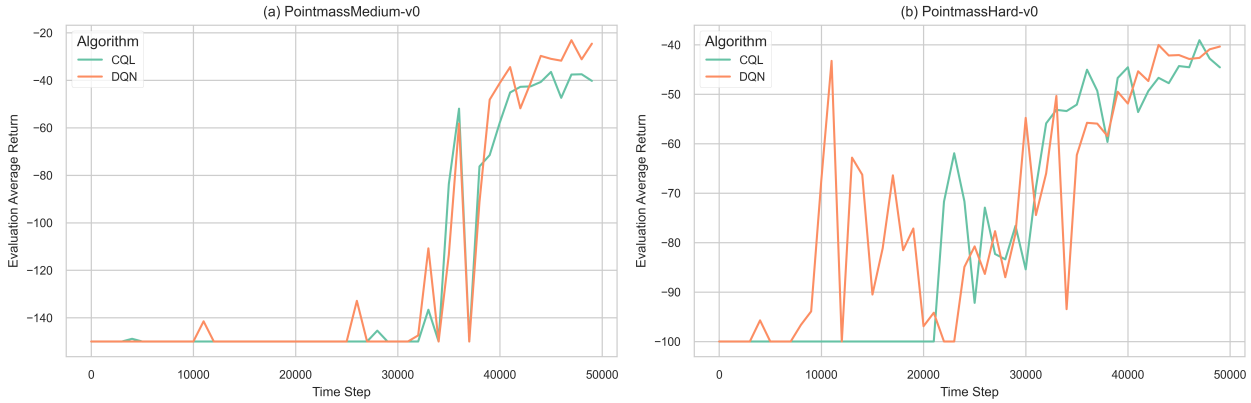


Figure 7. Learning curve of DQN- and CQL-based exploitation algorithms on (a) `PointmassMedium` and (b) `PointmassHard` environments. Compared to the results in Part 1, supervised exploration helps stablize the learning curve and is more effective regarding the convergence. The reason I can think of is that supervised exploration has a changing weighting of exploration vs. exploitation, which helps the algorithms to adjust its strategy across different learning period to explore more effectively.

Problem 4 Offline Learning with AWAC

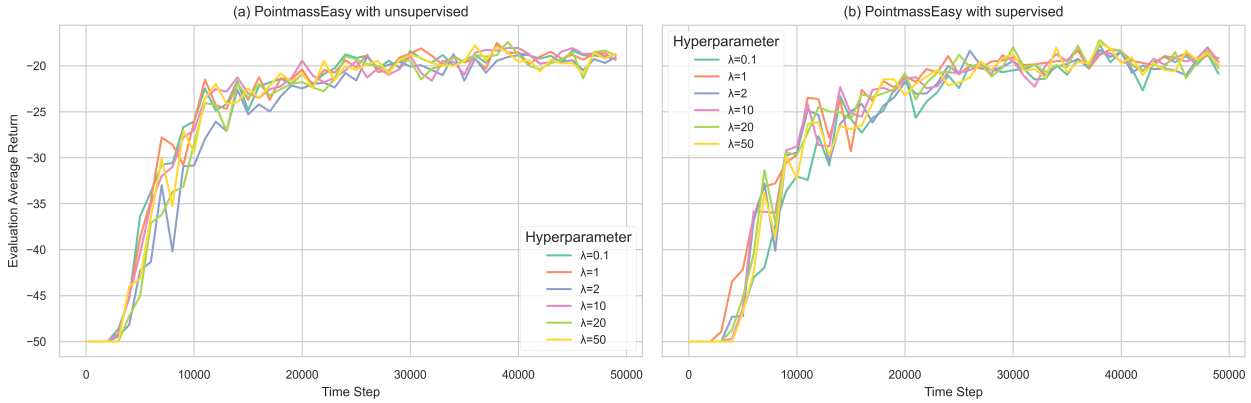


Figure 8. Learning curve from `PointmassEasy` environment: (a) supervised algorithm with different λ settings; (b) unsupervised algorithm with different λ settings. Explorations with supervised and unsupervised perform quite similar in this environment, which I think is partly due to the simplicity of the task. The best λ settings for unsupervised and supervised explorations are both $\lambda = 1$.

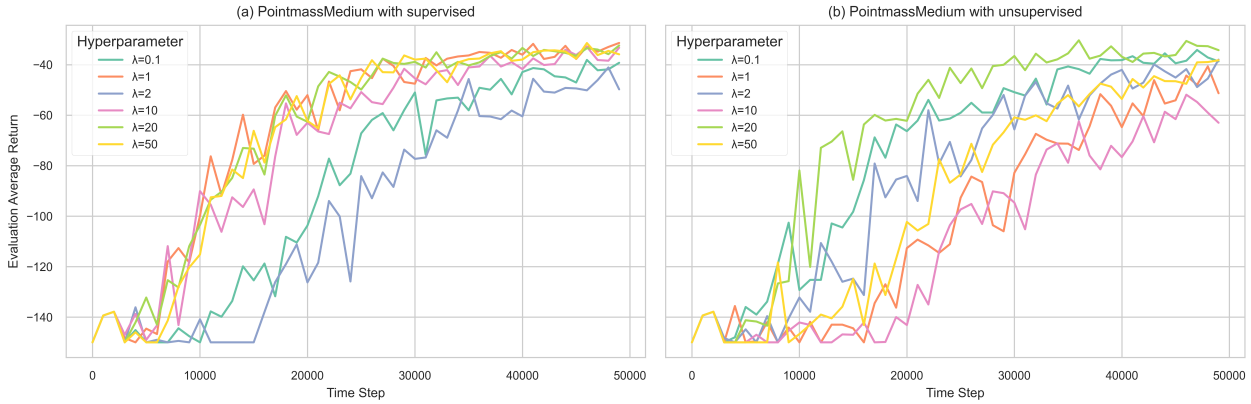


Figure 9. Learning curve from `PointmassMedium` environment: (a) supervised algorithm with different λ settings; (b) unsupervised algorithm with different λ settings. Training with supervised exploration is slightly better regarding the convergence speed. The best λ setting for supervised and unsupervised RND are $\lambda = 1$ and $\lambda = 20$, respectively.

Problem 5 Offline Learning with IQL

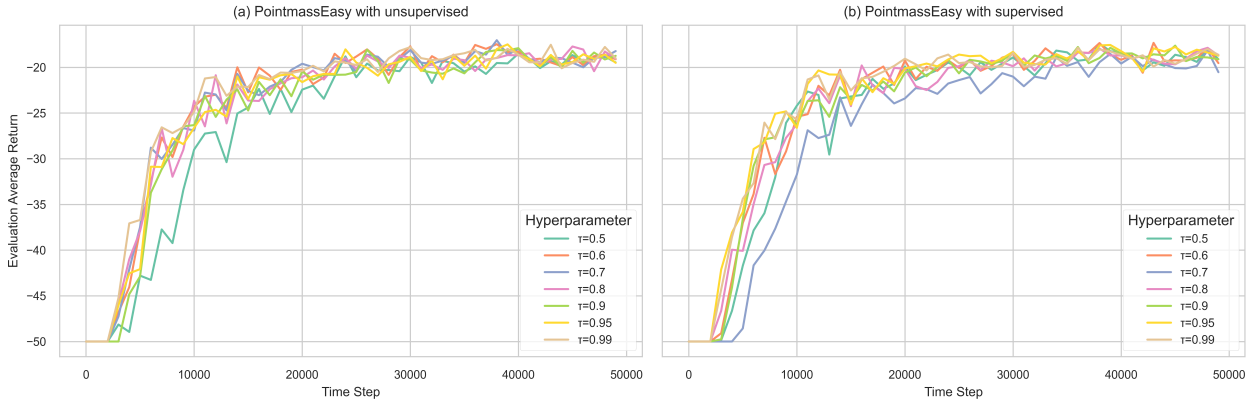


Figure 10. Learning curve from `PointmassEasy` environment: (a) supervised algorithm with different τ settings; (b) unsupervised algorithm with different τ settings. Increasing τ overall benefits the training but the marginal benefits decreases.

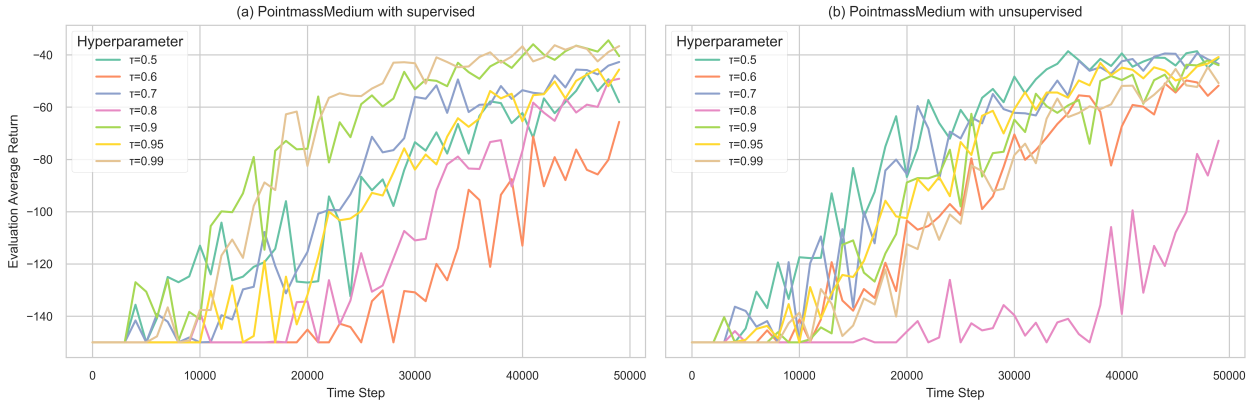


Figure 11. Learning curve from `PointmassMedium` environment: (a) supervised algorithm with different λ settings; (b) unsupervised algorithm with different λ settings. For supervised exploration, increasing τ overall benefits the training. But for the unsupervised exploration, increasing τ does not necessarily bring benefits. The optimal τ from my experiment is around $\tau = 0.7$.

In summary, by comparing the learning curve from Problem 2 - 5, my experiment results show that IQL performs slightly better than the AWAC on the `PointmassMedium` environment, while almost the same on the `PointmassEasy` one. The two algorithm perform significantly better than the CQL algorithm.