CS 285 Set 5

Erich Liang

Due: 11/17/21

1 Question 1

1.1 Subpart 1

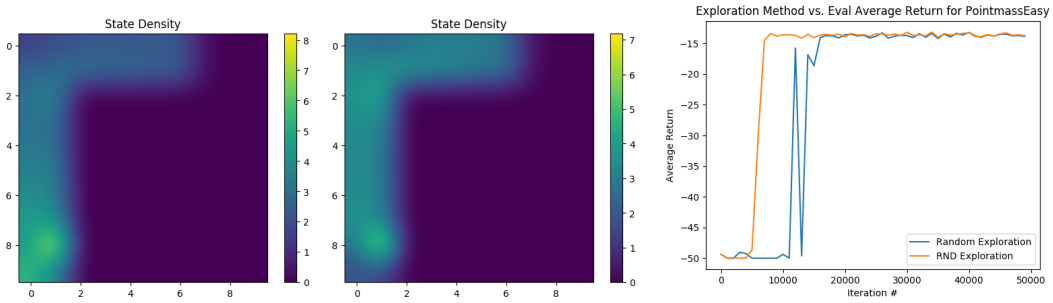


Figure 1: **Left**: State density plot for random exploration method and PointmassEasy. **Center**: State density plot for RND exploration method and PointmassEasy. **Right**: Learning curves for random exploration and RND exploration on PointmassEasy.

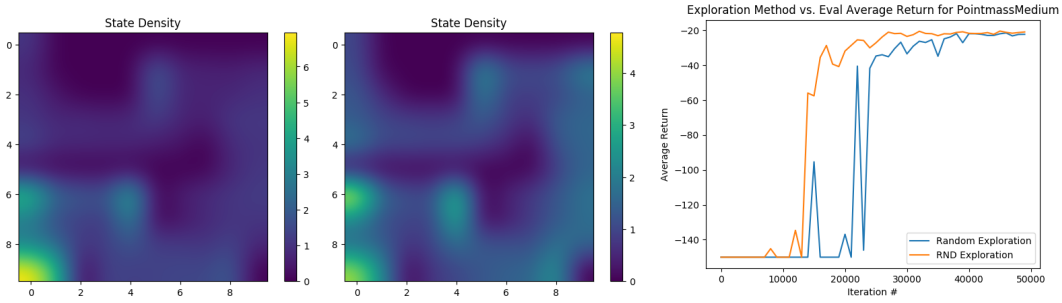


Figure 2: **Left**: State density plot for random exploration method and PointmassMedium. **Center**: State density plot for RND exploration method and PointmassMedium. **Right**: Learning curves for random exploration and RND exploration on PointmassMedium.

Overall, we can see that using RND exploration results in more uniform exploration of traversable space compared to random exploration. Additionally, the learning curve result-

ing from RND converges to higher return values faster than those resulting from random exploration.

1.2 Subpart 2

The new exploration I came up with is a variant of the RND exploration method; let us call this method "RND2". Instead of training only one neural net $\hat{f}_\phi(s')$ to learn the target random neural net $f^*_\theta(s')$, this new method utilizes two neural nets that are both attempting to mimic the target neural net. The exploration reward produced by this variant of RND is the softmax (log sum exponential) of the predicted errors of the individual neural networks.

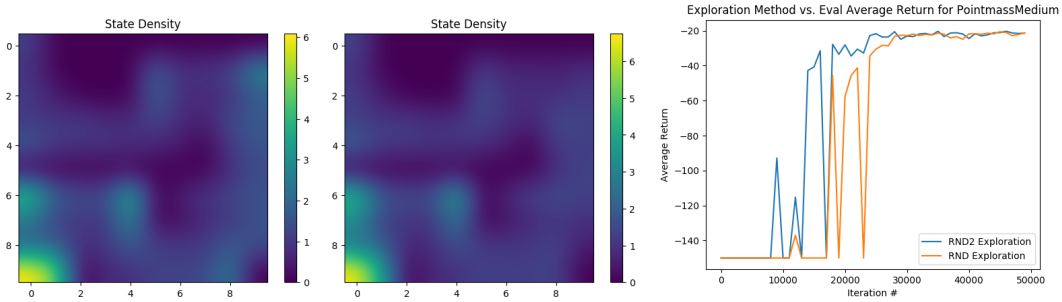


Figure 3: **Left**: State density plot for RND exploration method and PointmassMedium. **Center**: State density plot for RND2 exploration method and PointmassMedium. **Right**: Learning curves for RND and RND2 exploration on PointmassMedium.

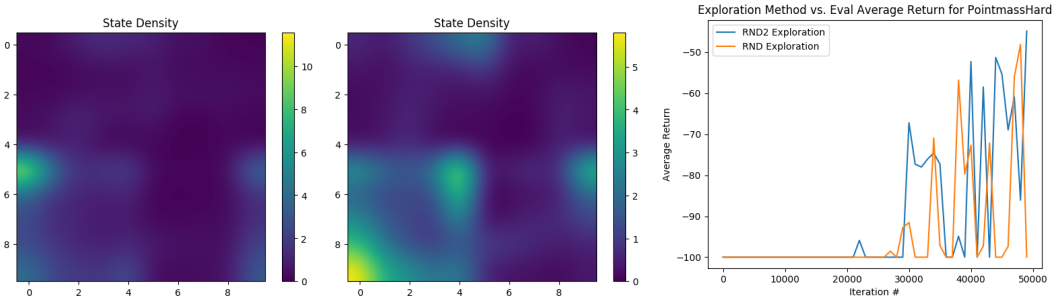


Figure 4: **Left**: State density plot for RND exploration method and PointmassHard. **Center**: State density plot for RND2 exploration method and PointmassHard. **Right**: Learning curves for RND and RND2 exploration on PointmassHard.

From the plots above, we can see that RND2 exploration generally results in a more uniform exploration of states rather than favoring states closer to the start. We also see that in general, RND2 nets higher returns at earlier iterations compared to RND. One reason for the differences between RND2 and RND is that because RND2 is implicitly using two randomly initialized neural nets to help keep track of states that have or haven't been visited before, it is harder for not highly visited states to be erroneously labeled as "visited" (since both trained networks need to have low error for the state to have an overall low error).

2 Question 2

2.1 Subpart 1

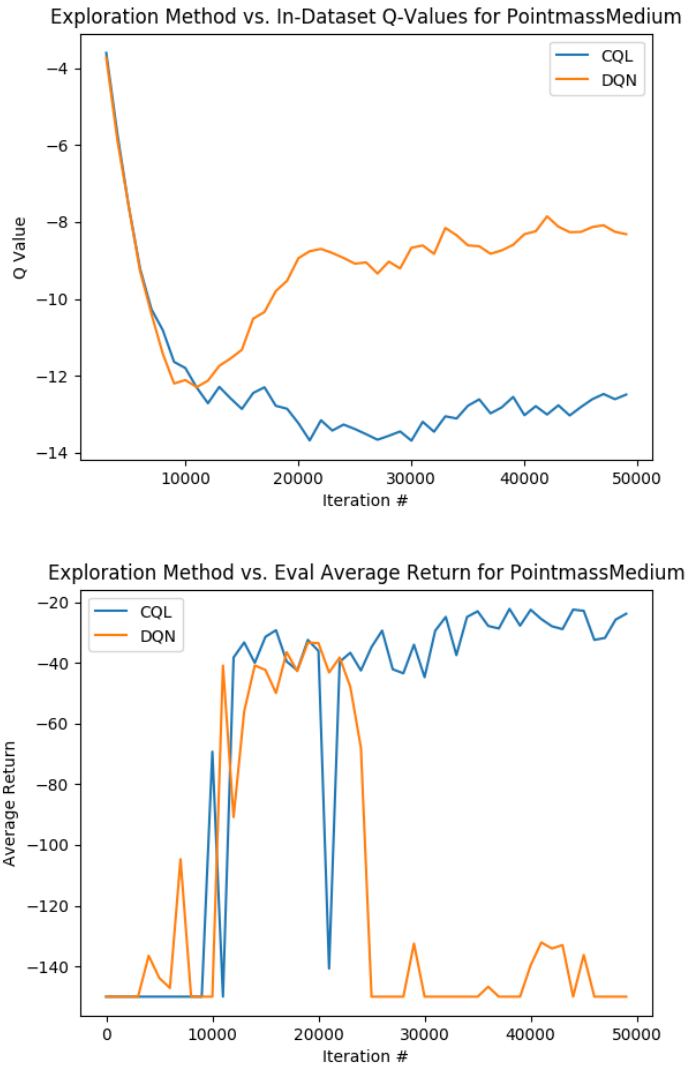
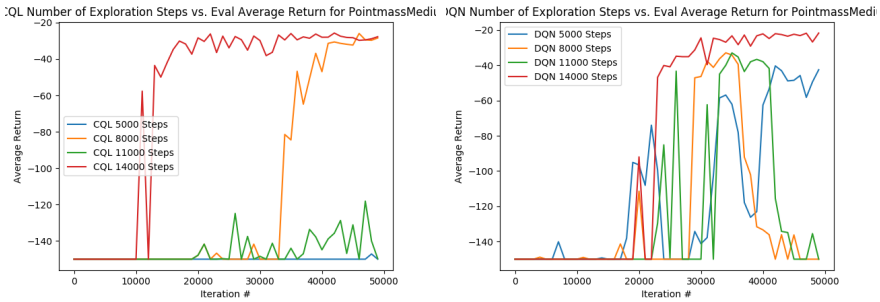


Figure 5: **Top**: Q-values for CQL and DQN for in-dataset state action pairs. **Bottom**: Learning curves for CQL and DQN on PointmassMedium.

From these plots, we can see that the Q-values for CQL are underestimates of the Q-values learned via DQN. Because CQL tends to underestimate Q-values (hence conservative Q learning), it is less perceptible towards taking out of distribution actions that may actually be very bad (due to lower bounding the Q values learned), which leads to overall higher and more consistent returns compared to DQN.

2.2 Subpart 2



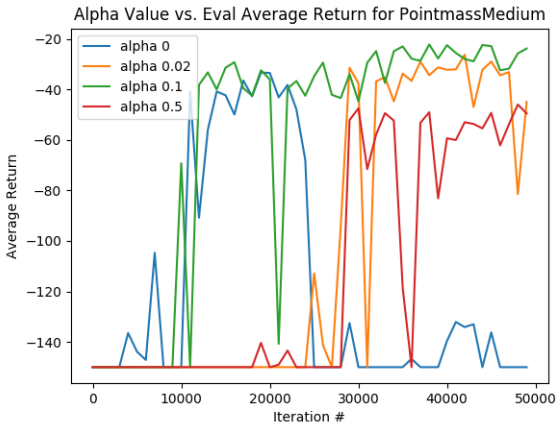
Number of Exploration Steps vs. Max Eval Average Return

	5000 Steps	8000 Steps	11000 Steps	14000 Steps
CQL	-147.14285278320312	-25.973684310913086	-118.11111450195312	-25.736841201782227
DQN	-40.36000061035156	-32.93333435058594	-32.96666717529297	-21.727272033691406

Figure 6: **Top Left**: Exploration Amount vs. Eval Average Return for CQL on PointmassMedium. **Top Right**: Exploration Amount vs. Eval Average Return for DQN on PointmassMedium. **Bottom**: Table of results.

From this ablation study, we can see that in general, as the number of exploration steps increases, the maximum return achieved as well as the consistency of the return increases. This trend makes sense; by providing both CQL and DQN more exploration steps before performing offline exploitation, both algorithms have the potential to learn a more accurate model for Q values, which in turn may the algorithms chose beneficial actions that increase return.

2.3 Subpart 3



Alpha Value vs. Eval Average Return for PointmassMedium

0	0.02	0.1	0.5
-33.33333206176758	-26.243244171142578	-22.204545974731445	-46.0

Figure 7: **Top**: Alpha vs. Eval Average Return for CQL on PointmassMedium. **Bottom**: Table of results.

From these results, we can see that in general, the optimal value of alpha is between 0.02 and 0.5 (maybe around 0.1, based on our tests). It makes sense that the optimal value of alpha is not extremely small or extremely large; this is because the alpha parameter affects the importance of the CQL regularizer and the standard TD error. If alpha is too small, the algorithm essentially becomes DQN and becomes prone to estimating too high of a Q value for out of distribution actions. However if alpha is too big, then the original objective of choosing actions with good Q values gets lost, since the actual Q value of the action gets overpowered by the large alpha.

3 Question 3

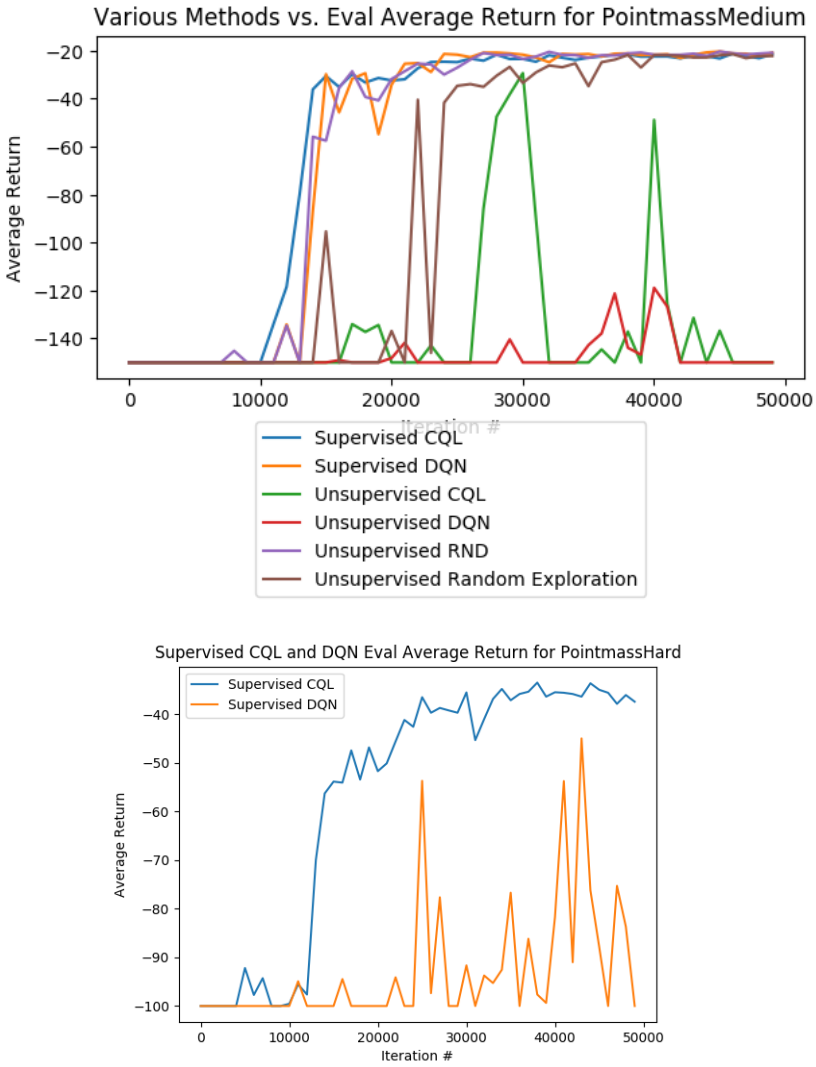


Figure 8: **Top**: Learning curves for PointmassMedium. **Bottom**: Learning curves for PointmassHard.

From the results for PointmassMedium, we can see that supervised CQL and DQN achieve higher returns earlier compared to unsupervised CQL and DQN (question 2 subpart 2) and the random exploration and RND exploration methods from question 1. One reason why supervised exploration outperforms pure RND is that the model is choosing to explore new places that have higher chances of giving high rewards; as a result, higher reward areas tend to be searched first, leading to faster gains.

4 Question 4

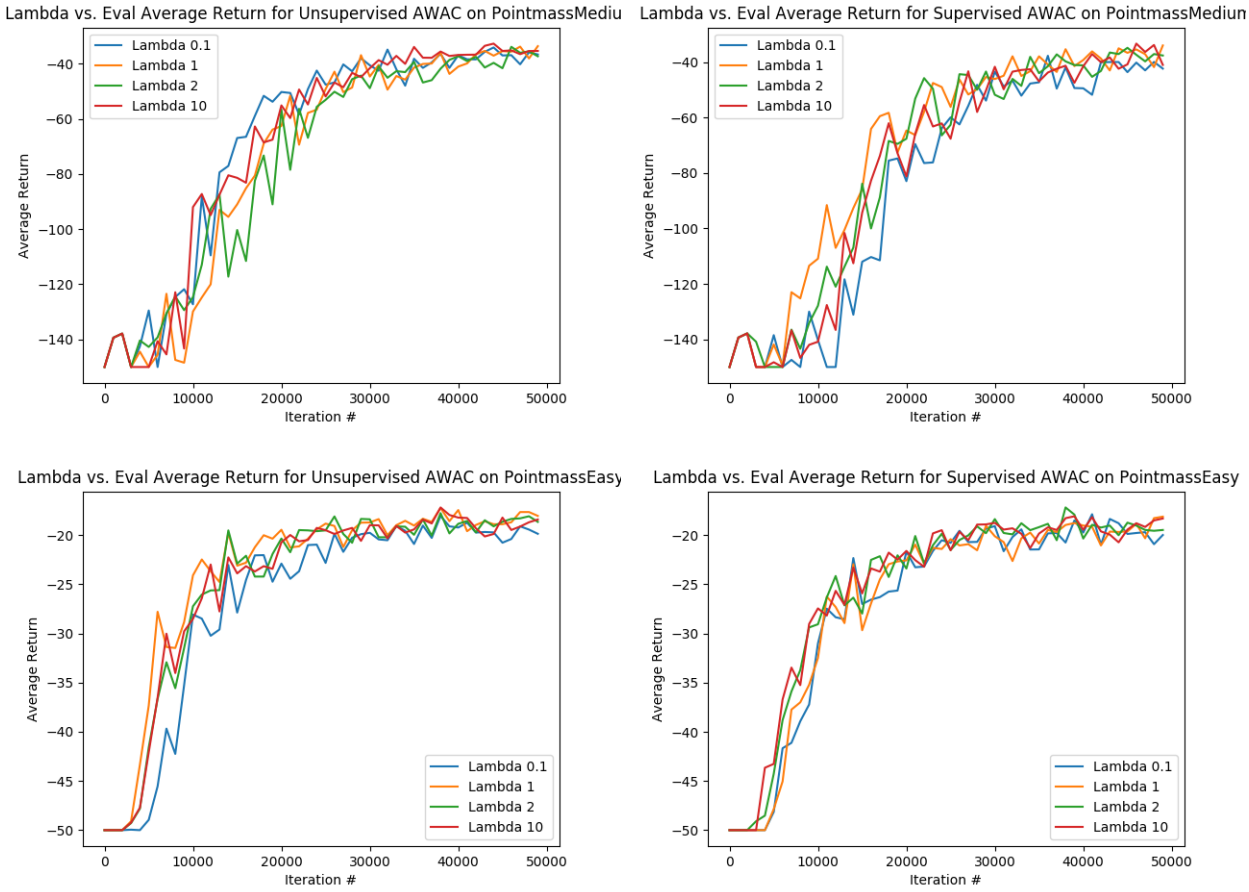


Figure 9: **Top Left**: Learning curves for unsupervised AWAC on PointmassMedium. **Top Right**: Learning curves for supervised AWAC on PointmassMedium. **Bottom Left**: Learning curves for unsupervised AWAC on PointmassEasy. **Bottom Right**: Learning curves for supervised AWAC on PointmassEasy.

For lambda that approach 0, the advantage's effect in AWAC increases dramatically to the point where the action distribution part of AWAC doesn't matter; this could potentially lead to overly-optimistic actor that may be prone to explore out of distribution actions due to small errors in the advantage estimation. For lambda that approach infinity, the advantage will have no effect in AWAC, which would make the actor only depend on its existing action distribution, which would lead to very little exploration overall.

In comparison to CQL, AWAC tends to achieve higher returns in fewer iterations.