

# Implicit Q Learning: Improvements on Antmaze

111652017 廖修誼、111652040 吳泓諺、111550066 王裕昕

## Outline

Part I. Introduction .....	1
Part II. Interesting findings .....	3
Part III. Tech. 1 - Distribution model.....	5
Part IV. Tech. 2 - D2RL .....	6
Part V. Tech. 3 - Bonus Reward .....	8
Part VI. Conclusion.....	10

## Part I. Introduction

- Concept of IQL

Offline reinforcement learning involves learning a policy that improves over the behavior policy that collected the dataset, while at the same time minimizing the deviation from the behavior policy so as to avoid errors due to distributional shift. Traditional methods need to query the value of unseen actions to improve the policy, which can lead to inaccuracies due to out-of-distribution actions.

The key insight in Implicit Q-Learning (IQL) is to treat the state value function as a random variable with randomness determined by the action and take a state conditional upper expectile of this random variable to estimate the value of the best actions. In this way, IQL will never query the values of any unseen actions.

- Algorithm Steps

Starting with a dataset  $D$  of state-action-reward-next state tuples  $(s, a, r, s')$ . The algorithm consists of two stages. First, we fit the value function and  $Q$ , performing a number of gradient updates alternating between fitting the upper expectile value function and backing it up into a  $Q$ -function. Second, we extract the policy via advantage-weighted behavioral cloning, which also avoids querying out-of-sample actions. For both steps, we take a minimum of two  $Q$ -functions for  $V$ -function and policy updates. The algorithm is shown in the figure below, where

three neural networks are implemented: value network, Q-network and policy network.

---

**Algorithm 1** Implicit Q-learning

---

```

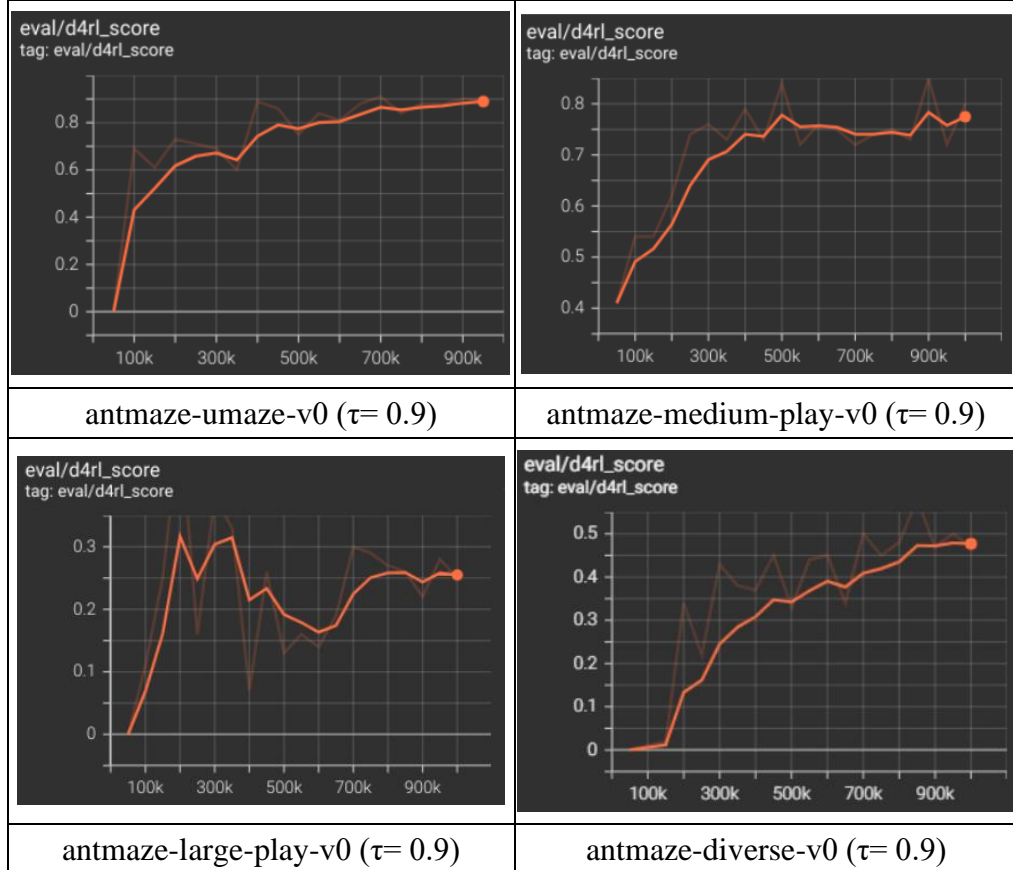
Initialize parameters  $\psi, \theta, \hat{\theta}, \phi$ .
TD learning (IQL):
for each gradient step do
   $\psi \leftarrow \psi - \lambda_V \nabla_{\psi} L_V(\psi)$ 
   $\theta \leftarrow \theta - \lambda_Q \nabla_{\theta} L_Q(\theta)$ 
   $\hat{\theta} \leftarrow (1 - \alpha)\hat{\theta} + \alpha\theta$ 
end for
Policy extraction (AWR):
for each gradient step do
   $\phi \leftarrow \phi - \lambda_{\pi} \nabla_{\phi} L_{\pi}(\phi)$ 
end for

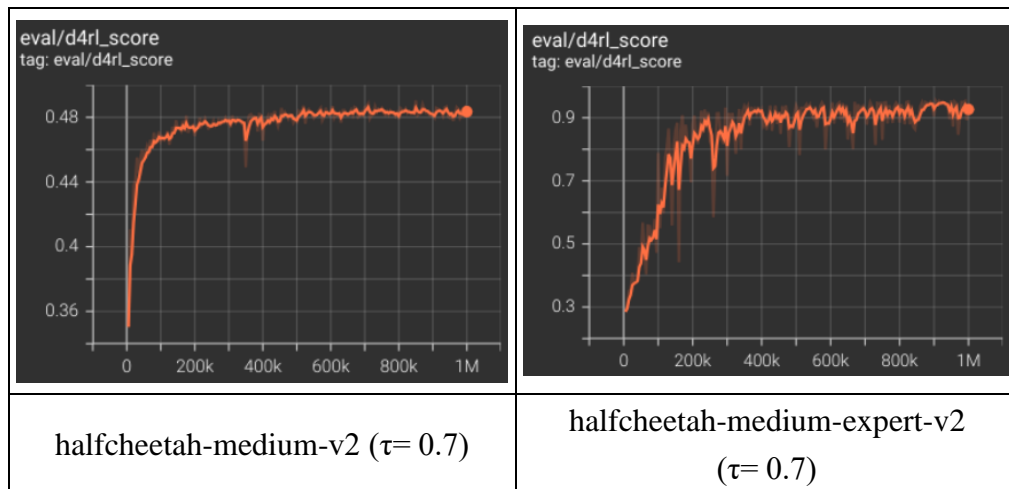
```

---

- **Reproduction**

We reproduce IQL in six different environments: antmaze-umaze-v0, antmaze-medium-v0, antmaze-large-v0, antmaze-large-diverse-v0, halfcheetah-medium-v2, halfcheetah-medium-expert-v2. The accumulated return scores are similar to the results in the original paper. However, the convergence in antmaze-large was not as expected, we assume that it's because we did not finetune the hyperparameters.





## Part II. Interesting findings

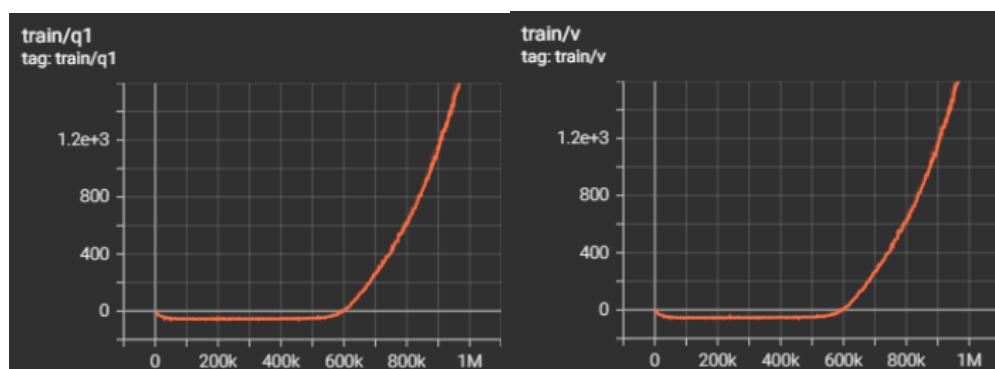
- Why can IQL converge

There are some interesting details on the equation of IQL.

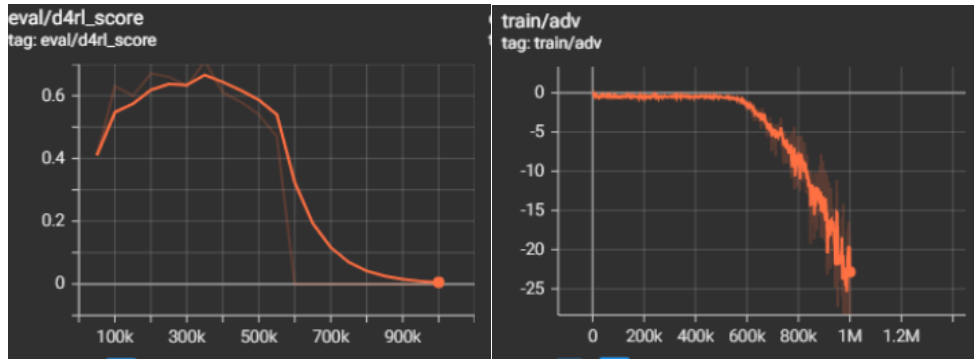
Recall that in the loss function of value function, it uses expectile regression to penalize those point whose  $Q > V$ . That is, it encourages  $V$  to be higher than  $Q$ . On the other hand, in the loss function of  $Q$  function, it uses MSE to encourage  $Q = r + \gamma * V$ .

It means that in each update, the value of  $V$  will transcend  $Q$ , while in the same time,  $Q$  will catch up with  $V$ . The scenario forms a racing competition, causing the value of  $Q$ ,  $V$  to be explode.

The below graph shows the explanation of value  $Q$  and  $V$



But why can IQL converge in official code? It turns out that it is because they add a double  $Q$  to constraint the overestimation of  $Q$ . In our experiment above, we showed that without double  $Q$  implementations, the algorithm may not converge.

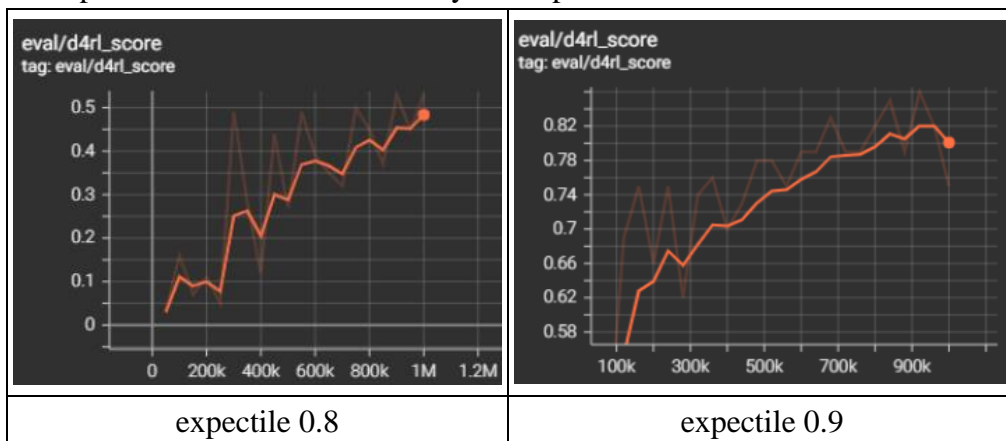


- The importance of expectile

When conducting the experiments, we find that when we adjust expectile to be bigger, the d4rl score improves.

The scenario occurs because by the Lemma2 of IQL, the bigger the expectile is, the bigger the value function is.

The phenomenon can be shown by the experiment below



- Evolve from quantile regression loss

When compared the idea with Distributional RL method, such as quantile regression. We found that the quantile regression loss is similar to the expectile regression of IQL.

Therefore, we guess that the idea of IQL comes from quantile regression.

## Part III. Tech. 1 - Distribution model

- Motivation idea

When learning distribution RL in class, we find that by distributionalize the DQN, we could get the best result among all rainbow techniques. What if we could also turn IQL to distributional form, then will it improve the algorithm?

- Compared to quantile regression loss

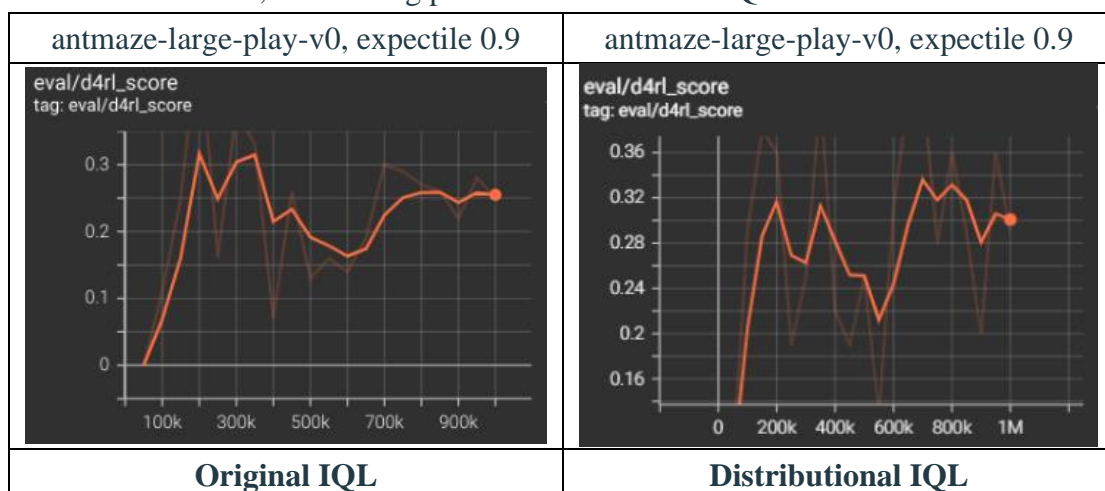
I have mentioned that the idea of IQL may come from quantile regression. Then, what is the difference between distributional IQL and quantile regression?

In our design, we use the method from implicit quantile regression to produce quantiles of  $Q$ ,  $V$  functions, while the loss equation is the same as original IQL. It turns out that we train two network  $Q$  and  $V$  in our design, while we train only  $Z$  network in quantile regression.

We believed that with training 2 networks instead of 1 network, we could get a tighter constraint on out of distribution data. Moreover, in our design we perform two step to perform policy improvement, which is the same as one step improvement method in quantile regression.

- The True result & analysis

However, the result of distribution IQL does not outperform original IQL too much. Besides, the training process of distribution IQL is unstable.



After thinking more about the theoretical issue, I find that our design may produce more error.

Given that distributional bellman operator is gamma-contraction in Wasserstein distance. We could derive the form of quantile regression loss like below.

$$\mathcal{L}_p(Z, \hat{T}^\pi \hat{Z}^k) \approx \mathcal{L}_\kappa(\delta; \tau) \quad \text{where} \quad \delta = r + \gamma G_{\theta'}(\tau'; s', a') - G_\theta(\tau; s, a).$$

Here,  $\delta$  is the distributional TD error,  $\theta'$  are the parameters of the target  $Q$ -network [30], and

$$\mathcal{L}_\kappa(\delta; \tau) = \begin{cases} |\tau - \mathbb{1}(\delta < 0)| \cdot \delta^2 / (2\kappa) & \text{if } |\delta| \leq \kappa \\ |\tau - \mathbb{1}(\delta < 0)| \cdot (|\delta| - \kappa / 2) & \text{otherwise} . \end{cases} \quad (7)$$

However, when we do MSE on loss function of  $Q$ , we get a non-accurate TQ, it means that our TQ does not approximate the one-step bellman operator well.

Consequently, the convergence of distributional IQL may be unstable.

$L_Q(\theta) = \mathbb{E}_{(s,a,s') \sim \mathcal{D}} \left[ (r(s,a) + \gamma V_\psi(s') - Q_\theta(s,a))^2 \right]$	When doing MSE on $Q$ , $Q$ is just a approximation on $r + \gamma^* V (T V)$
$L_V(\psi) = \mathbb{E}_{(s,a) \sim \mathcal{D}} \left[ L_2^\tau (Q_{\hat{\theta}}(s,a) - V_\psi(s)) \right]$	Then when we do expectile regression on $Q - V$ , we will find out that $Q$ doesn't equal to $TV$ , which cause the unstable convergence

## Part IV. Tech. 2 - D2RL

- **Motivation**

Inspired by the paper *D2RL: Deep Dense Architectures in Reinforcement Learning* (Sinha et al.), we want to change the model that has been largely ignored. The idea of D2RL is inspired by successful architecture choices in computer vision. Sample-efficiency is fundamentally critical to RL agents and baking in minimal inductive biases into the framework is one effective mechanism to address the issue of sample-efficiency of deep reinforcement learning agents. Previous work has sought to improve the sample efficiency by adding an inductive bias of invariance, when learning from images, through techniques such as data augmentations. We want to imitate the success from training from images. Hence, we add skip connections in our neural networks.

- **Skip Connection**

Skip connections (SC) are implemented by concatenating the output of previous layer with the input state or state-action pair to form the new input for the next layer.

original:

nn.Linear(in\_dim, hidden\_dim)

nn.Linear(hidden\_dim, hidden\_dim)

nn.Linear(hidden\_dim, out\_dim)

with skip connections:

nn.Linear(in\_dim, hidden\_dim)

nn.Linear(in\_dim + hidden\_dim, hidden\_dim)

nn.Linear(hidden\_dim, out\_dim)

In the results of D2RL, the implementation of skip connections can help to alleviate the issue of rank collapse, which refers to the reduction of effective ranks. We want to verify this point in our own implementation and also test if we can obtain better performance.

- **Effective Rank**

Kumar et al. (2020) showed that using MLPs for function approximation of the policy and value functions in deep RL algorithms that use bootstrapping, leads to reduction in the effective rank of the feature. The effective rank of the feature is denoted  $srank_{\delta}(\Phi)$ , where  $\Phi$  is the weight matrix of the penultimate layer of the network, is given by

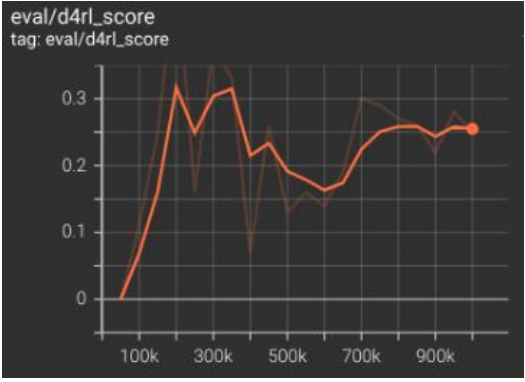
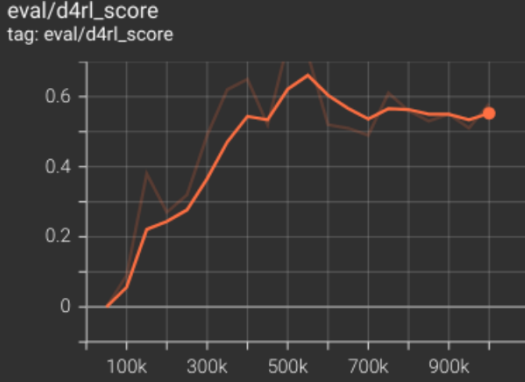
$$srank_{\delta}(\Phi) = \min \{k: \frac{\sum_{i=1}^k \sigma_i(\Phi)}{\sum_{i=1}^d \sigma_i(\Phi)} \geq 1 - \delta\}$$

where  $\{\sigma_i(\Phi)\}$  are the singular values of the matrix  $\Phi$  sorted in decreasing order.  $\delta$  is the threshold and here we set it to 0.01.

- **Results**

	antmaze-large		antmaze-medium		halfcheetah-expert	
1M-steps	IQL	IQL+SC	IQL	IQL+SC	IQL	IQL+SC
Policy	227	<b>232</b>	225	<b>232</b>	226	<b>232</b>
Q-network	223	<b>231</b>	221	<b>230</b>	225	<b>233</b>

The above table shows the srnk values of policy network and Q-network in three different environments. The results reveal that by combining IQL with skip connections, we can increase the effective rank of the feature matrix. With higher rank values, we suppose that there is less information loss throughout the forward propagation. In our experiments, IQL+SC obtains similar d4rl scores with original IQL in most environments. Noticeably, IQL+SC greatly outperforms original IQL in antmaze-large as shown in the figure below, which shows better convergence.

antmaze-large-play-v0, expectile 0.9	antmaze-large-play-v0, expectile 0.9
 <p>eval/d4rl_score tag: eval/d4rl_score</p>	 <p>eval/d4rl_score tag: eval/d4rl_score</p>
Original IQL	IQL+SC

It is clearly shown that skip connections can help to increase the effective ranks of the feature matrices and thus improve data utilization. We believe that by making more efforts on finetuning the parameters, the d4rl scores of IQL+SC will likely outnumber that of original IQL.

## Part V. Tech. 3 - Bonus Reward

- Motivation idea

Initially, we want to add intrinsic reward to encourage us choose not out of distribution state action pairs (it refers to [Never Give Up: Learning Directed Exploration Strategies](#) paper). But when we looked at this NGU implementation code, we found that it violated IQL setting (because of Offline reinforcement learning setting). Therefore, we tried to find something relative to “reward” and “offline environment”. We found that [Offline Reinforcement Learning as Anti-Exploration](#) this paper conforms to our setting. And the idea in this paper is very intuitive, it uses “penalty reward” to penalize the state action pairs which seldom shows in the dataset. Because if the state action pairs usually show in the dataset, this state action pairs are less probable to occur out of distribution. And we call this “penalty reward” as Bonus Reward. Following, we will talk about how to choose our Bonus Reward.

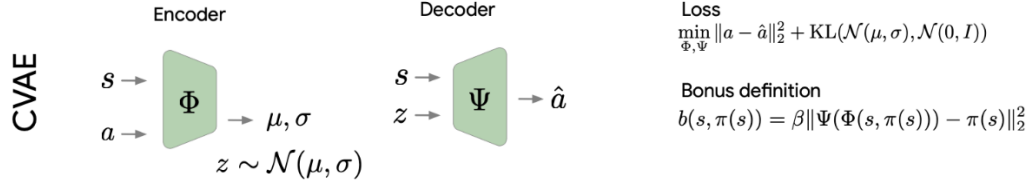
- Choose our Bonus Reward

In this paper, it provides many methods to choose Bonus Reward. But now we focus on Bouns Reward in the continuous space.

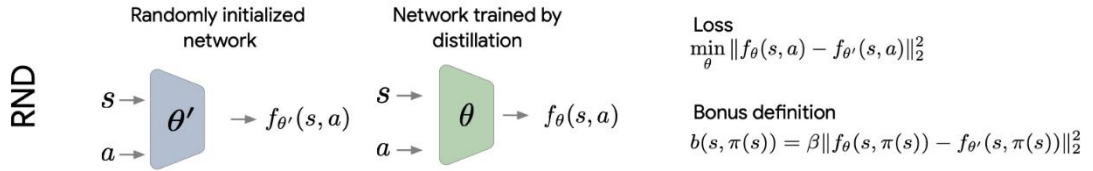
Method 1: We use “Algorithm 1 CVAE training” in [Offline Reinforcement Learning as Anti-Exploration](#) paper.



First it trains two additional network  $\Phi$  and  $\Psi$ . Second, it uses the following formula to get our Bonus Reward network. And the idea is that if I have more state action pairs  $(s, a)$ , then I can get more accurate action after encoding and decoding. Therefore, the distance of two terms in the following Bonus definition formula is less. So, the bonus reward is less.



Method 2: We use “RND” in [Offline Reinforcement Learning as Anti-Exploration](#) paper. The idea is similar to previous ones. It only changes two training networks and the Bonus definition, we can see following:



- Apply Bonus Reward idea to our IQL algorithm

Let us look at “Algorithm 2 Modified TD3 training” in [Offline Reinforcement Learning as Anti-Exploration](#) paper together.

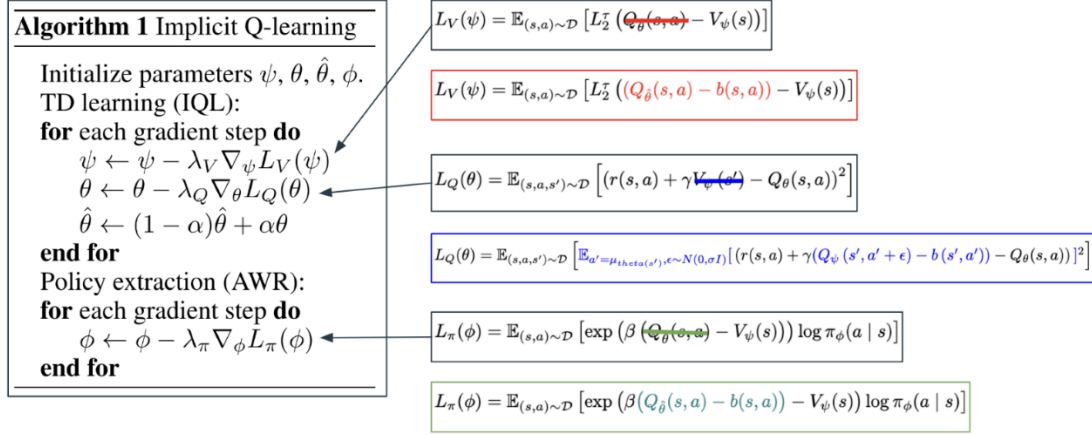
---

**Algorithm 2** Modified TD3 training.

---

- 1: Initialize policy  $\pi_{\theta}$ , action-value network  $Q_{\omega}$  and target network  $Q_{\bar{\omega}}, Q_{\bar{\omega}}$
  - 2: **for** step  $i = 0$  to  $N$  **do**
  - 3:   Sample a minibatch of  $k$  transitions  $\{(s_t, a_t, r_t, s_{t+1}), t = 1, \dots, k\}$  from  $\mathcal{D}$
  - 4:   For each transition, decode the action and computer the bonus, see Eq. (6)
  - 5:   Update critic: gradient step on  $J_{\text{td3, critic}}, b(\omega)$ , see Eq. (4)
  - 6:   Update actor: gradient step on  $J_{\text{td3, actor}}, b(\theta)$ , see Eq. (3)
  - 7:   Update target network  $Q_{\bar{\omega}} := Q_{\omega}$
- 

The updating formula in (3) and (4) like we change  $Q(s, a)$  as  $Q(s, a) - b(s, a)$ . Therefore, we use this idea to modify our IQL updating formula like following



And we finish our modified IQL pseudo code.

## Part VI. Conclusion

In this technical report of our team project, we first made an introduction to IQL and mention some interesting details of implementation. Second, we propose three methods of IQL improvements, which are

- Distribution model (modify update formulas to distribution form)
- D2RL (modify the architecture of the networks)
- Bonus Reward (modify  $Q$  as  $Q-b$ )

Each of these enhancements represents a step towards refining IQL for more effective and stable performance in challenging environments. While distributional IQL showed promising theoretical foundations, practical implementations revealed challenges in achieving stable convergence. On the other hand, integrating skip connections proved effective in enhancing data utilization, particularly in complex environments.

Future work could focus on refining the distributional model approach to address stability issues and exploring additional architectural modifications. More experiments are also expected to be conducted in other environments to prove the generalizability of these improvement methods. Ultimately, these efforts aim to push the boundaries of offline reinforcement learning, making it more robust and applicable to real-world scenarios.