

# [Re] Reproducibility report of "Conservative Q-Learning for Offline Reinforcement Learning"

Rui Wang 20337240

2023.1.6

## 1 Reproducibility Summary

The title of the article I chose to reproduce is "Conservative Q-Learning for Offline Reinforcement Learning", whose authors are Aviral Kumar, Aurick Zhou, George Tucker and Sergey Levine. This article provides the original code, and I have reproduced the code. See Gitee link for the code and reproducibility report:

<https://github.com/wtcz11/CQL>

I found some CQL materials on the network, and discussed with several students. Finally, I came to the reproduction results. I would like to express my thanks to them here.

## 2 Introduction

### 2.1 About the article

This article was proposed by Sergey Levine's team of UC Berkeley in 2020 and published at the NIPS 2020. The main idea of the paper is to add a regulator on the basis of Q value to learn a conservative Q function. The author theoretically proves that CQL can generate a lower bound of the real value of the current policy, and it is a process of policy evaluation and policy promotion. From the code point of view, the regularizer in this

paper can be implemented in only 20 lines of code, greatly improving the experimental results.

## **2.2 Difference between CQL algorithm and previous offline reinforcement learning algorithm**

Effective use of previously collected large data sets in reinforcement learning (RL) is a key challenge for large-scale real-world applications. Before the CQL algorithm came out, the solution to the distributed offset problem in offline reinforcement learning was to limit the action selection of the strategy to be optimized to the action distribution of the offline data set, so as to avoid the over-estimation of the Q value of the out-of-distribution actions, and thus reduce the impact of unknown actions in the strategy training and learning process. The offline reinforcement learning algorithm is expected to learn effective strategies from the collected static data set without further interaction.

The improvement usually only uses the value of the policy, and can learn a less conservative lower bound Q function, so that only the expected value of the Q function under the policy is lower bound, not lower bound point by point. The authors propose a new method to learn the conservative Q function by simply modifying the RL algorithm based on the standard value. The key idea behind this method is to minimize the value under the appropriately selected distribution on the state operation tuple, and then further tighten this limit by adding a maximization term on the data distribution.

## **3 Methodology**

### **3.1 Strategy iteration**

In the process of optimizing this strategy, the value function of the corresponding strategy will be obtained, and the value of the strategy will be estimated according to the value function. The strategy is further improved by maximizing the Q function, and then doing a greedy search on the Q

function. The following picture is the specific formula. This is the strategy evaluation and strategy improvement method.

$$\tilde{Q}^{k+1} \leftarrow \arg \min_Q \mathbb{E}_{\mathbf{s}, \mathbf{a}, \mathbf{s}' \sim \mathcal{D}} \left[ \left( \left( r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \hat{\pi}^k(\mathbf{a}'|\mathbf{s}')} [\tilde{Q}^k(\mathbf{s}', \mathbf{a}')] \right) - Q(\mathbf{s}, \mathbf{a}) \right)^2 \right] \quad (\text{policy evaluation})$$

$$\hat{\pi}^{k+1} \leftarrow \arg \max_{\pi} \mathbb{E}_{\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \pi^k(\mathbf{a}|\mathbf{s})} [\hat{Q}^{k+1}(\mathbf{s}, \mathbf{a})] \quad (\text{policy improvement})$$

### 3.2 Algorithm framework

The key of offline reinforcement learning algorithm is to avoid the over-estimation of  $Q$  value caused by distribution deviation. CQL algorithm starts directly from the value function, aiming to find the lower bound estimate of the original  $Q$ -value function, and then uses it to optimize the strategy with more conservative policy value. The following is the update formula of  $Q$  function:

$$\begin{aligned} \hat{Q}^{k+1} \leftarrow \arg \min_Q & \cdot \left( \mathbb{E}_{\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \mu(\mathbf{a}|\mathbf{s})} [Q(\mathbf{s}, \mathbf{a})] - \mathbb{E}_{\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \hat{\pi}_\beta(\mathbf{a}|\mathbf{s})} [Q(\mathbf{s}, \mathbf{a})] \right) \\ & + \frac{1}{2} \mathbb{E}_{\mathbf{s}, \mathbf{a}, \mathbf{s}' \sim \mathcal{D}} \left[ \left( Q(\mathbf{s}, \mathbf{a}) - \hat{B}^\pi \hat{Q}^k(\mathbf{s}, \mathbf{a}) \right)^2 \right] \end{aligned}$$

When  $\mu(a|s) = \pi(a|s)$

$$\forall \mathbf{s} \in \mathcal{D}, \hat{V}^\pi(\mathbf{s}) \leq V^\pi(\mathbf{s}) - \alpha \left[ (I - \gamma P^\pi)^{-1} \mathbb{E}_\pi \left[ \frac{\pi}{\hat{\pi}_\beta} - 1 \right] \right] (\mathbf{s}) + \left[ (I - \gamma P^\pi)^{-1} \frac{C_{r,T,\delta} R_{\max}}{(1 - \gamma) \sqrt{|\mathcal{D}|}} \right]$$

At this time, the estimated value is not necessarily the lower bound of the real  $Q$  value, but the value of the strategy is the lower bound of the real value function. At this time, similar to the process of strategy iteration in online,  $\mu$  is defined as a strategy that can maximize the  $Q$  value.

$$\begin{aligned} \min_Q \max_{\mu} & \left( \mathbb{E}_{\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \mu(\mathbf{a}|\mathbf{s})} [Q(\mathbf{s}, \mathbf{a})] - \mathbb{E}_{\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \hat{\pi}_\beta(\mathbf{a}|\mathbf{s})} [Q(\mathbf{s}, \mathbf{a})] \right) \\ & + \frac{1}{2} \mathbb{E}_{\mathbf{s}, \mathbf{a}, \mathbf{s}' \sim \mathcal{D}} \left[ \left( Q(\mathbf{s}, \mathbf{a}) - \hat{B}^{\pi_k} \hat{Q}^k(\mathbf{s}, \mathbf{a}) \right)^2 \right] + \mathcal{R}(\mu) \quad (\text{CQL}(\mathcal{R})). \end{aligned}$$

Theorem 3.1 guarantees that the lower bound of  $Q$  value can be obtained by iteration eq1. Theorem 3.2 ensures that the  $Q$  value obtained by

$$\forall \mathbf{s} \in \mathcal{D}, \mathbf{a}, \hat{Q}^\pi(s, a) \leq Q^\pi(\mathbf{s}, \mathbf{a}) - \alpha \left[ (I - \gamma P^\pi)^{-1} \frac{\mu}{\hat{\pi}_\beta} \right] (\mathbf{s}, \mathbf{a}) + \left[ (I - \gamma P^\pi)^{-1} \frac{C_{r,T,\delta} R_{\max}}{(1 - \gamma) \sqrt{|\mathcal{D}|}} \right] (\mathbf{s}, \mathbf{a}).$$

图 1: Theorem 3.1

$$\forall \mathbf{s} \in \mathcal{D}, \hat{V}^\pi(\mathbf{s}) \leq V^\pi(\mathbf{s}) - \alpha \left[ (I - \gamma P^\pi)^{-1} \mathbb{E}_\pi \left[ \frac{\pi}{\hat{\pi}_\beta} - 1 \right] \right] (\mathbf{s}) + \left[ (I - \gamma P^\pi)^{-1} \frac{C_{r,T,\delta} R_{\max}}{(1 - \gamma) \sqrt{|\mathcal{D}|}} \right] (\mathbf{s}).$$

图 2: Theorem 3.2

iteration eq2 can be used as the lower bound of V, not the point-by-point lower bound of Q value.

Using the derived conservative Q for strategy evaluation can make the improved strategy value remain conservative. Theorem 3.2 give a tighter bound than 3.1, and select the appropriate alpha. Both bounds are still valid under the sampling error and function approximation. The lower bound of the learned Q value is:

$$\mathbb{E}_{\pi_{\hat{Q}^k}(\mathbf{a}|\mathbf{s})} \left[ \frac{\pi_{\hat{Q}^k}(\mathbf{a} | \mathbf{s})}{\hat{\pi}_\beta(\mathbf{a} | \mathbf{s})} - 1 \right] \geq \max_{\mathbf{a} \text{ s.t. } \hat{\pi}_\beta(\mathbf{a}|\mathbf{s}) > 0} \left( \frac{\pi_{\hat{Q}^k}(\mathbf{a} | \mathbf{s})}{\hat{\pi}_\beta(\mathbf{a} | \mathbf{s})} \right) \cdot \varepsilon$$

The distance between the estimated value of Q and the true value is increased, making Q more conservative.

### 3.3 Pseudocode

---

**Algorithm 1** Conservative Q-Learning (both variants)

---

- 1: Initialize Q-function,  $Q_\theta$ , and optionally a policy,  $\pi_\phi$ .
  - 2: **for** step  $t$  in  $\{1, \dots, N\}$  **do**
  - 3:   Train the Q-function using  $G_Q$  gradient steps on objective from Equation 4  
     $\theta_t := \theta_{t-1} - \eta_Q \nabla_\theta \text{CQL}(\mathcal{R})(\theta)$   
    (Use  $\mathcal{B}^*$  for Q-learning,  $\mathcal{B}^{\pi_{\phi_t}}$  for actor-critic)
  - 4:   (only with actor-critic) Improve policy  $\pi_\phi$  via  $G_\pi$  gradient steps on  $\phi$  with SAC-style entropy regularization:  
     $\phi_t := \phi_{t-1} + \eta_\pi \mathbb{E}_{\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \pi_\phi(\cdot|\mathbf{s})} [Q_\theta(\mathbf{s}, \mathbf{a}) - \log \pi_\phi(\mathbf{a}|\mathbf{s})]$
  - 5: **end for**
- 


## 4 Results

### 4.1 Code base

Codes of Atari:

此电脑 > windowsD (D:) > CQL-master > atari > batch_rl >				
名称	修改日期	类型	大小	
__pycache__	2023/1/9 19:20	文件夹		
baselines	2023/1/9 19:20	文件夹		
fixed_replay	2023/1/9 19:20	文件夹		
multi_head	2023/1/9 19:20	文件夹		
tests	2023/1/9 19:20	文件夹		
__init__.py	2020/9/15 3:20	JetBrains PyChar...	1 KB	

此电脑 > windowsD (D:) > CQL-master > atari > batch\_rl > fixed\_replay >

名称	修改日期	类型	大小
 __pycache__	2023/1/9 19:20	文件夹	
 agents	2023/1/9 19:20	文件夹	
 configs	2023/1/9 19:20	文件夹	
 replay_memory	2023/1/9 19:20	文件夹	
 __init__.py	2020/9/15 3:20	JetBrains PyChar...	1 KB
 run_experiment.py	2020/9/15 3:20	JetBrains PyChar...	5 KB
 train.py	2020/9/15 3:20	JetBrains PyChar...	4 KB

Codes of d4rl:

此电脑 > windowsD (D:) > CQL-master > d4rl >

名称	修改日期	类型	大小
 docs	2023/1/9 19:20	文件夹	
 environment	2023/1/9 19:20	文件夹	
 examples	2023/1/9 19:20	文件夹	
 rlkit	2023/1/9 19:20	文件夹	
 scripts	2023/1/9 19:20	文件夹	
 .gitignore	2020/9/15 3:20	文本文档	1 KB
 LICENSE	2020/9/15 3:20	文件	2 KB
 setup.py	2020/9/15 3:20	JetBrains PyChar...	1 KB

此电脑 > windowsD (D:) > CQL-master > d4rl > examples >

名称	修改日期	类型	大小
doodad	2023/1/9 19:20	文件夹	
her	2023/1/9 19:20	文件夹	
skewfit	2023/1/9 19:20	文件夹	
cql_antmaze_new.py	2020/9/15 3:20	JetBrains PyChar...	7 KB
cql_mujoco_new.py	2020/9/15 3:20	JetBrains PyChar...	7 KB
ddpg.py	2020/9/15 3:20	JetBrains PyChar...	4 KB
dqn_and_double_dqn.py	2020/9/15 3:20	JetBrains PyChar...	3 KB
sac.py	2020/9/15 3:20	JetBrains PyChar...	4 KB
td3.py	2020/9/15 3:20	JetBrains PyChar...	4 KB

## 4.2 Execution method

For running CQL, use the following command:

```
python -um batch_rl.fixed_replay.train \
  --base_dir=/tmp/batch_rl \
  --replay_dir=$DATA_DIR/Pong/1 \
  --agent_name=quantile \
  --gin_files='batch_rl/fixed_replay/configs/quantile_pong.gin' \
  --gin_bindings='FixedReplayRunner.num_iterations=1000' \
  --gin_bindings='atari.lib.create_atari_environment.game_name = "Pong"' \
  --gin_bindings='FixedReplayQuantileAgent.min_q_weight=1.0'
```

For running CQL on the MuJoCo environments, use the following command:

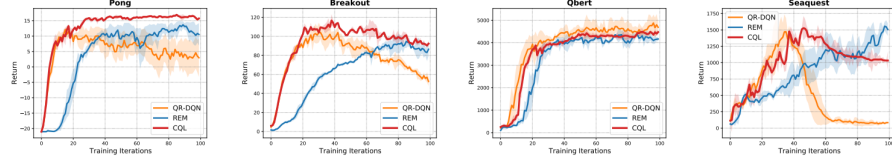
```
python examples/cql_mujoco_new.py --env=<d4rl-mujoco-env-with-version e.g.
  hopper-medium-v0>
  --policy_lr=1e-4 --seed=10 --lagrange_thresh=-1.0
  --min_q_weight=(5.0 or 10.0) --gpu=<gpu-id> --min_q_version=3
```

In conclusion, the experimental results are very good.

### 4.3 Results of Gym

Task Name	SAC	BC	BEAR	BRAC-p	BRAC-v	CQL( $\mathcal{H}$ )
halfcheetah-random	30.5	2.1	25.5	23.5	28.1	<b>35.4</b>
hopper-random	<b>11.3</b>	9.8	9.5	<b>11.1</b>	<b>12.0</b>	<b>10.8</b>
walker2d-random	4.1	1.6	<b>6.7</b>	0.8	0.5	<b>7.0</b>
halfcheetah-medium	-4.3	36.1	38.6	<b>44.0</b>	<b>45.5</b>	<b>44.4</b>
walker2d-medium	0.9	6.6	33.2	72.7	<b>81.3</b>	79.2
hopper-medium	0.8	29.0	47.6	31.2	32.3	<b>58.0</b>
halfcheetah-expert	-1.9	<b>107.0</b>	<b>108.2</b>	3.8	-1.1	104.8
hopper-expert	0.7	<b>109.0</b>	<b>110.3</b>	6.6	3.7	<b>109.9</b>
walker2d-expert	-0.3	125.7	106.1	-0.2	-0.0	<b>153.9</b>
halfcheetah-medium-expert	1.8	35.8	51.7	43.8	45.3	<b>62.4</b>
walker2d-medium-expert	1.9	11.3	10.8	-0.3	0.9	<b>98.7</b>
hopper-medium-expert	1.6	<b>111.9</b>	4.0	1.1	0.8	<b>111.0</b>
halfcheetah-random-expert	53.0	1.3	24.6	30.2	2.2	<b>92.5</b>
walker2d-random-expert	0.8	0.7	1.9	0.2	2.7	<b>91.1</b>
hopper-random-expert	5.6	10.1	10.1	5.8	11.1	<b>110.5</b>
halfcheetah-mixed	-2.4	38.4	36.2	<b>45.6</b>	<b>45.9</b>	<b>46.2</b>
hopper-mixed	3.5	11.8	25.3	0.7	0.8	<b>48.6</b>
walker2d-mixed	1.9	11.3	10.8	-0.3	0.9	<b>26.7</b>

### 4.4 Results of Atari



### 4.5 Results of D4RL

Domain	Task Name	BC	SAC	BEAR	BRAC-p	BRAC-v	CQL( $\mathcal{H}$ )	CQL( $\rho$ )
AntMaze	antmaze-umaze	65.0	0.0	<b>73.0</b>	50.0	70.0	<b>74.0</b>	<b>73.5</b>
	antmaze-umaze-diverse	55.0	0.0	61.0	40.0	70.0	<b>84.0</b>	61.0
	antmaze-medium-play	0.0	0.0	0.0	0.0	0.0	<b>61.2</b>	4.6
	antmaze-medium-diverse	0.0	0.0	8.0	0.0	0.0	<b>53.7</b>	5.1
	antmaze-large-play	0.0	0.0	0.0	0.0	0.0	<b>15.8</b>	3.2
	antmaze-large-diverse	0.0	0.0	0.0	0.0	0.0	<b>14.9</b>	2.3
Adroit	pen-human	34.4	6.3	-1.0	8.1	0.6	37.5	<b>55.8</b>
	hammer-human	1.5	0.5	0.3	0.3	0.2	<b>4.4</b>	2.1
	door-human	0.5	3.9	-0.3	-0.3	-0.3	<b>9.9</b>	<b>9.1</b>
	relocate-human	0.0	0.0	-0.3	-0.3	-0.3	0.20	<b>0.35</b>
	pen-cloned	<b>56.9</b>	23.5	26.5	1.6	-2.5	39.2	40.3
	hammer-cloned	0.8	0.2	0.3	0.3	0.3	2.1	<b>5.7</b>
	door-cloned	-0.1	0.0	-0.1	-0.1	-0.1	0.4	<b>3.5</b>
	relocate-cloned	<b>-0.1</b>	-0.2	-0.3	-0.3	-0.3	<b>-0.1</b>	<b>-0.1</b>
Kitchen	kitchen-complete	33.8	15.0	0.0	0.0	0.0	<b>43.8</b>	31.3
	kitchen-partial	33.8	0.0	13.1	0.0	0.0	<b>49.8</b>	<b>50.1</b>
	kitchen-undirected	47.5	2.5	47.2	0.0	0.0	<b>51.0</b>	<b>52.4</b>



## 5 Discussion

### 5.1 What contribution has this article made

This article makes some changes to the objective of the Q function. In addition to the original minimum MSE, on the one hand, it minimizes the estimation of actions outside the distribution, on the other hand, it maximizes the estimation of actions inside the distribution. In this way, the impact of OOD actions can be minimized in value estimation. Because this idea is relatively simple to implement, it can be easily applied to many algorithms

### 5.2 Advantages of the article

The article has well grasped the key points of the algorithm. In fact, if reinforcement learning is to make the agent perform well, the key is to accurately estimate the value function. The starting point of the article is the estimation of the Q function. It is equivalent to conducting policy evaluation first, and then optimizing the strategy based on a better value function.

And the theoretical analysis of this article is very detailed, from whether the value estimate itself is the lower bound to the analysis of the strategy improvement, it is very comprehensive.

CQL optimizes a well-defined and punitive empirical RL goal, and improves the behavior policy with high reliability. The degree of improvement is negatively affected by the higher sampling error. The more samples observed, the smaller the sampling error.

### 5.3 Is there anything that can be improved

As a college student, I indeed don't know much about deep learning. So when I reproduce the code, the installation of some environments makes me headache. And some formulas in the article are obscure and difficult to understand. But it cannot be denied that this article is an excellent article about offline reinforcement learning, and has made some amazing results.

In addition, the article does not provide proof of sampling error, and directly incorporates sampling error into  $\alpha$  selected. And we see that the expected amount of additional underestimation introduced by iteration  $k$  under the action sampled from the behavior strategy is 0. This may be a flaw in this article.