ITM Web of Conferences **80**, 01043 (2025) https://doi.org/10.1051/itmconf/20258001043
ACAAI 2025

Evaluating Conservative Q-Learning Algorithms across Dataset Qualities: A Case Study on Hopper

Jiaheng Zeng*

Zhejiang University-University of Illinois Urbana-Champaign Institute, Zhejiang University, Jiaxing, Zhejiang, 314400, China

Abstract. Offline reinforcement learning (RL) has gained attention as an effective approach for training decision-making policies using pre-existing datasets, eliminating the need for additional interactions with the environment. However, the performance of offline RL algorithms strongly depends on the quality and coverage of the available data, which remains an open challenge in practical applications. Existing studies have primarily focused on algorithmic design but provided limited systematic comparisons across different dataset qualities. This study aims to evaluate and compare three representative algorithms—Q-Learning, Batch-Constrained Q-Learning (BCQ), and Conservative Q-Learning (CQL)—under varying dataset qualities on the Hopper benchmark from the D4RL suite. Using datasets of random, medium, and expert quality levels, this study systematically analyzes the performance of these three reinforcement learning algorithms. Experimental results show that both BCQ and CQL significantly outperform standard Q-learning, particularly on higher-quality datasets, while CQL demonstrates greater robustness. Nevertheless, both CQL and BCQ show a sensitivity for the hyperparameters, indicating room for optimization in stability and computational efficiency.

1 Introduction

Reinforcement learning represents a major area within the field of machine learning, and it has shown some promising applications in the field of medicine and robotics [1, 2], so it is receiving more and more attention. Reinforcement learning can be categorized into online learning and offline learning, depending on whether it should collect data in real time. Offline reinforcement learning only uses precollected fixed data sets, greatly improving the efficiency of training samples, while online reinforcement learning requires the agent to interact with the environment and use feedback from the environment to train itself [3]. Therefore, in many real-world scenarios, online interaction with the environment is costly and risky, especially in areas of medicine and robotics. This leads to growing attention in offline reinforcement learning.

However promising offline reinforcement learning is, it also faces some other challenges. When evaluating those actions that don't appear in the training dataset, standard Q-Learning

* Corresponding author: Jiaheng1.22@intl.zju.edu.cn

suffers from the overestimation of Q-value [4]. Fujimoto et al. introduced Batch-Constrained deep Q-Learning (BCQ), and evaluated it on D4RL datasets, showing that BCQ could receive about 1200 return values on Hopper-v1 and improve return values by 75% compared to traditional Deep Q-Learning (DQN) [5].

Similarly, Kumar et al. put forward Conservative Q-Learning (CQL), achieving up to 86.6 normalized score on D4RL Hopper-medium task, outperforming traditional Q-Learning, which only achieves 0.8 normalized score [6].

To systematically evaluate the effectiveness of CQL, this paper conducts experiments on gym-MuJoCo D4RL benchmarks [7]. This paper's experiments aim at demonstrating the comparison of CQL and other baseline methods, like BCQ, and normal Q-learning.

This study has the following contributions. First, the paper introduces CQL into the offline RL setting and provides a detailed analysis of its properties. Second, the research benchmarks CQL extensively on D4RL datasets [7], covering continuous control and navigation tasks. Finally, the paper shows that CQN achieves strong performance and improved stability compared to existing baseline approaches.

2 Method

2.1 CQL foundation

The fundamental principle underlying all forms of reinforcement learning is to train an agent so that it can achieve maximum rewards in a specific environment based on the Markov Decision Process model [1]. Fig. 1 shows how the Markov Decision Process model works.

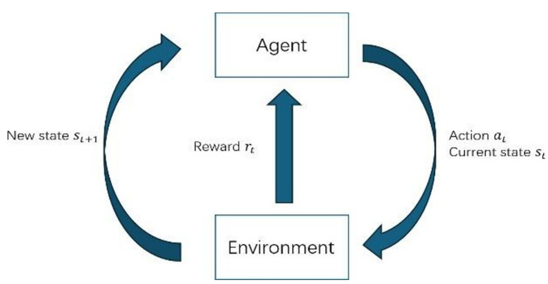

Fig. 1 Markov Decision Processes: at each time step, the agent will choose one action based on the environment and policy. After its action, the agent will receive a reward from the environment and change its state (Picture credit: Original).

In short, the Markov Decision Model can be described by the five-element tuple (S, A, P, R, γ), where S denotes state space, and A denotes action space, and P denotes the probability of transferring into a new state given current state and current action, and R denotes reward given current state and current action, and γ indicates the discount factor [3]. Thus, the target of reinforcement learning is to learn the policy $\pi(a|s)$, denoting what action the agent should take, given the current state, so that it can maximize the following formula [1]:

$$\mathrm{J}(\pi) = E_{\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)] \tag{1}$$

Here $J(\pi)$ represents the cumulative reward under a given policy π, s_t indicates the agent's state at time step t, a_t denotes the action taken by the agent at time step t.

The value functions [8] can be described as:

$$Q^{\pi}(s, a) = E_{\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|\ s_0 = s, a_0 = a] \tag{2}$$

Formula (2) describes the total rewards the agent can get given current state and current action under a specific policy.

and:

$$V^{\pi}(s) = E_{\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s] \quad (3)$$

Formula (3) describes the total rewards the agent can get given the current state and specific policy.

Q-learning is one of the most fundamental algorithms in reinforcement learning, which aims to learn the optimal action-value function [8].

Bellman optimality equation defines the optimal Q-function [8]:

$$Q^*(s,a) = R(s,a) + \gamma \, E_{s'\sim P}\left[\max_{a'} Q^*(s',a')\right] \quad (4)$$

In practice, Q-learning updates its estimated Q-function by the following rule [8]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right) \quad (5)$$

Where α denotes the learning rate.

However, when it comes to offline reinforcement learning, people cannot guarantee that the dataset is good enough for the agent to learn useful information. Instead, its performance often deteriorates because of overestimating actions that lie outside the distribution of the training data [3]. To solve this problem, Kumar et al. put forward CQL, introducing a regularization term into (5) [6]. Formally, CQL optimizes the following objective:

$$\min_{Q} E_{(s,a,r,s')\sim\mathbb{D}}\left[\left(Q(s,a) - (r + \gamma \, E_{a'\sim\pi}[Q(s',a')])\right)^2\right] + \beta(E_{a\sim\pi}[Q(s,a)] - E_{a\sim\mathbb{D}}[Q(s,a)]) \quad (6)$$

where β is a hyperparameter used to control the regularization term $E_{a\sim\pi}[Q(s,a)] - E_{a\sim\mathbb{D}}[Q(s,a)]$, denoting the difference of expectation reward value between all possible action in current policy and all appeared action in offline dataset.

Kumar et al. prove that CQL can guarantee a lower bound of expected policy rewards for offline reinforcement learning, improving its stability [6].

2.2 Experiment evaluation

This paper compares CQL with other reinforcement learning algorithms: normal Q-learning and BCQ [5] on datasets from the D4RL benchmarks [7]. D4RL offers multiple datasets for the same task, varying in quality levels such as random, medium, expert, and others [7]. The dataset with the lowest quality, the random dataset, is created using a completely random policy and reflects trajectories of very low quality [7]. The medium dataset originates from a policy that has undergone partial training, resulting in performance that reaches roughly half of what an expert policy would achieve [7]. The expert dataset consists of trajectories generated by a near-optimal policy [7]. This paper chooses the Hopper task from the Gym-MuJoCo environment, and selects random, medium, and expert datasets to examine how these reinforcement learning algorithms behave in datasets with different quality because Hopper is a widely used continuous control benchmark with moderate difficulty and good reproducibility, making it suitable for comparing the stability and generalization ability. Besides, because different tasks in D4RL vary in training time and computational cost, this paper selects Hopper as a baseline benchmark under limited sources, ensuring the reproducibility and computational feasibility of this experiment.

During training on offline Hopper datasets, this research periodically evaluates the current policy in the online Hopper environment every 10 epochs. For each evaluation, the policy is executed for 10 episodes, and the reported reward is the average return over these episodes. In this way, randomness can be greatly reduced.

3 Results

This paper reports the final performance of BCQ, CQL, and Q-learning in Fig. 2. The figures below depict training performance, with epochs displayed on the horizontal axis and the corresponding average return in the Hopper environment shown on the vertical axis. The orange line represents the normal Q-learning algorithm, the green line represents the CQL algorithm, and the blue line represents the BCQ algorithm. From left to right, the subFigures correspond to the random, medium, and expert datasets.

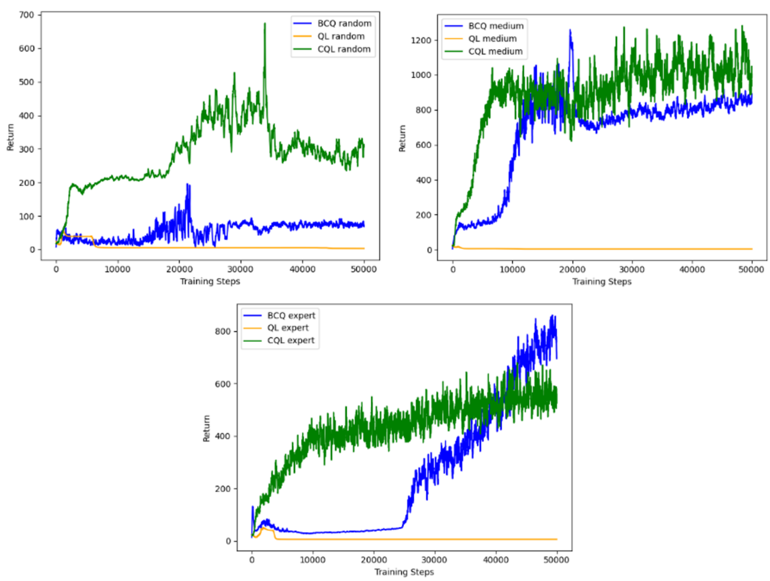

Fig. 2 CQL, BCQ, and Q-learning performance on three kinds of datasets (Picture credit: Original).

When they are trained with the random dataset, CQL has a dominant performance over the other two, and BCQ performs better than normal Q-learning. For the medium dataset, BCQ improves its performance greatly, with slightly worse performance than CQL. When it comes to the “expert” dataset, BCQ outperforms CQL in the late stage of training.

This paper also records the best performance of BCQ.CQL and Q-learning on Table 1.

Table 1. Best performance comparison of BCQ, Q-learning, and CQL on datasets with different qualities

Task name	BCQ	Normal Q-learning	CQL
Hopper-random-v2	195.9210222206679	55.6178769649945	674.4276257891963
Hopper-medium-v2	1258.660284410739	24.207457506277702	1281.575056570563
Hopper-expert-v2	860.8868747697145	51.23537864462746	679.8087210645971

According to Table 1, both BCQ and CQL have a better performance than normal Q-learning. And CQL demonstrates significant superiority on the medium and random datasets, where the data quality is relatively suboptimal. In contrast, BQN exhibits stronger performance on the expert datasets, which are dominated by near-optimal actions.

4 Discussion

The experiment results highlight the complementary strengths of CQL and BCQ when trained on datasets with different qualities. CQL achieves the dominant performance over random and medium datasets. This is due to the regularization term in formula (6), which penalizes the out-of-distribution actions while providing a worst-case estimate of the Q-function [6]. This mechanism achieves the robustness of CQL, especially when the only accessible training dataset doesn’t have high quality but contains a large proportion of suboptimal trajectories.

In contrast, BCQ demonstrates a stronger performance on the expert dataset, especially in the late stage of training. This effect can be explained by BCQ’s framework, which limits the learned policy from straying far from the behavior policy [5]. When the dataset is filled with almost optimal trajectories, this kind of constraint guides BCQ to learn expert-like behavior from the expert dataset, resulting in better performance than CQL.

These findings suggest that the choice of reinforcement learning algorithms should depend on the quality of the dataset. For those datasets mixed with noisy, low-quality trajectories or generated by random policies, CQL provides a more reliable choice, ensuring that the result will not be too bad. By contrast, when high-quality datasets are available, BCQ can better fit the high-quality datasets, leading to better performance.

Nevertheless, the present research is limited to the Hopper task, and further investigation is required to verify whether the same conclusions hold across more complex environments or tasks with different reward rules. Besides, both BCQ and CQL still have a slight tendency to improve performance in the very late stage of training, meaning that they don’t completely converge. Additionally, both BCQ and CQL are sensitive to the hyperparameters, resulting in a slight change of hyperparameters may cause a totally different performance [9]. Even though this experiment tries to use the same hyperparameters, there are still some different hyperparameters since they have different network structures. Further investigation should increase the number of training epochs and use more advanced computational power so that they can converge completely. Besides, to achieve a fairer comparison, the selection for the hyperparameters should not be randomly chosen but by the optimization algorithm, like Bayesian optimization [10].

5 Conclusion

This study conducted a comprehensive comparison of three representative reinforcement learning algorithms—Q-learning, Batch-Constrained Q-learning (BCQ), and Conservative Q-learning (CQL)—on the Hopper task from the D4RL benchmark under datasets of different qualities, including random, medium, and expert. The experimental results clearly demonstrate that the algorithm's effect relies greatly on the quality of the training dataset. All methods exhibited lower performance when trained on the random dataset, reflecting the limited learning potential from low-quality trajectories. However, CQL has obviously better performance than the other two algorithms, suggesting that CQL can better hold the lower bound. As dataset quality improved to medium and expert, both BCQ and CQL achieved higher returns and more stable convergence, with CQL showing slightly better robustness across data qualities and BCQ showing its high demand of the quality of the training dataset.

The results further confirm the effectiveness of incorporating conservative or behavior-constraint mechanisms in offline reinforcement learning. Compared with standard Q-learning, both BCQ and CQL significantly mitigated the issue of overestimation on unseen actions, which represents a major challenge in offline scenarios, where additional interactions with the environment are not possible. They show a promising application in some critical areas, especially for those that will cost a lot to collect online data, like robotics and medicine.

Despite the promising results, this study also observed that both BCQ and CQL failed to converge completely in the expert dataset and the sensitivity of the hyperparameters for both BCQ and CQL, such as the regularization weight α in CQL and the perturbation range Φ in BCQ. Future studies may explore automated hyperparameter optimization techniques to improve performance consistency. Overall, this work provides valuable empirical insights into how offline RL algorithms behave under varying dataset qualities and contributes to a deeper understanding of their strengths and limitations in real-world offline scenarios.

References

1. A. Coronato, M. Naeem, G. De Pietro, G. Paragliola, Reinforcement learning for intelligent healthcare applications: A survey. Artif. Intell. Med. **109**, 101964 (2020)
2. P. Wang, C.-Y. Chan, A. de La Fortelle, A reinforcement learning based approach for automated lane change maneuvers. Proc. IEEE Intell. Veh. Symp. (IV), 1379–1384 (2018)
3. X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, Q. Miao, Deep reinforcement learning: A survey. IEEE Trans. Neural Netw. Learn. Syst. **35**(4), 5064–5078 (2024)
4. R. F. Prudencio, M. R. O. A. Maximo, E. L. Colombini, A survey on offline reinforcement learning: Taxonomy, review, and open problems. IEEE Trans. Neural Netw. Learn. Syst. **35**(8), 10237–10257 (2024)
5. S. Fujimoto, D. Meger, D. Precup, Off-policy deep reinforcement learning without exploration. Proc. 36th Int. Conf. Mach. Learn. (ICML), 2052–2062 (2019)
6. A. Kumar, A. Zhou, G. Tucker, S. Levine, Conservative Q-learning for offline reinforcement learning. Adv. Neural Inf. Process. Syst. **33**, 1179–1191 (2020)
7. J. Fu, A. Kumar, O. Nachum, G. Tucker, S. Levine, D4RL: Datasets for deep data-driven reinforcement learning. arXiv preprint arXiv:2004.07219 (2020)
8. B. Jang, M. Kim, G. Harerimana, J. W. Kim, Q-learning algorithms: A comprehensive classification and applications. IEEE Access **7**, 133653–133667 (2019)
9. T. Eimer, M. Lindauer, R. Raileanu, Hyperparameters in reinforcement learning and how to tune them. Proc. 40th Int. Conf. Mach. Learn. (ICML) **202**, 9104–9149 (2023)
10. M. T. Young, J. D. Hinkle, R. Kannan, A. Ramanathan, Distributed Bayesian optimization of deep reinforcement learning algorithms. J. Parallel Distrib. Comput. **139**, 43–52 (2020)