Stable Conservative Q-Learning for Offline Reinforcement Learning

Zhenyuan Ji

Department of Automation, Shanghai Jiao Tong University, Shanghai, 200240, China

jizhenyuansjtu@sjtu.edu.cn

Abstract. Offline Reinforcement learning (RL) uses the collected data to train agents without interaction with the environment. However, due to the inconsistent distribution between the data set and the real world, the training samples collected in the real environment cannot be well applied to offline RL. In this paper, a model-free offline RL method is designed and is named Stable Conservative Q-Learning (SCQL). It uses a similar approach as Conservative Q-Learning (CQL) to limit the Q estimation of out-of-distribution (OOD) actions. The limitations on the estimation of OOD action is eliminated by combining Variational Autoencoders (VAE) with an estimation network that is trained without using OOD actions. It adopts a value-constrained approach to conservatively estimate the Q value, ensuring the stability of the algorithm's results while not affecting its generalization ability. Experimental results demonstrate that SCQL achieves conservative Q-function estimation while maintaining superior stability and generalization compared to baseline offline RL algorithms, including CQL. The proposed method effectively mitigates the negative impact of data distribution mismatch in offline RL, leading to improved performance and robustness.

Keywords: Stable Conservative Q-Learning, Offline Reinforcement Learning

1 Introduction

Reinforcement learning continuously strengthens the selection of high-return strategies to achieve the optimal goal through the reward signals obtained by the machine through interaction with the environment [1]. Currently, it has been widely used in robot control, gaming, autonomous driving, financial analysis, and other fields [2]. However, before the agent training obtains a better strategy, interaction with the environment may cause huge risks, such as the wrong estimation of stock price fluctuations and the possibility of accidents caused by autonomous vehicles. On the other hand, Reinforcement learning needs a lot of data for training. As a paradigm of online learning, Reinforcement learning is inefficient and time-consuming in online data sampling. Consequently, offline Reinforcement learning is introduced to avoid such problems.

P. Kar et al. (eds.), *Proceedings of the 2023 International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2023)*, Advances in Computer Science Research 108,
https://doi.org/10.2991/978-94-6463-300-9_18

Agents using offline reinforcement learning do not interact with the environment but only use collected datasets for training. Fig.1. shows the framework of offline reinforcement learning. Inspired by off-policy reinforcement learning, researchers believe that offline reinforcement learning can be a good way to learn strategies from datasets, but this is not the case. S Levine et. al. collects the trajectory data generated by intelligent agents trained under online algorithms [3]. The performance of the agent is very poor by using such data. Due to the lack of interaction with the environment, an error named “extrapolation error” occurs in policy evaluation during the training process, which makes the online learning experience cannot be directly transferred to offline Reinforcement learning. The reason for the extrapolation error is the distribution shift. The offline dataset used for training cannot contain all the data in the real world, thus inevitably resulting in inconsistent data distribution issues. In this case, the policy obtained from the offline dataset is only effective for the action-state pairs that exist in the dataset. Since the training goal is to maximize the value Q, when the training policy accesses actions that do not exist in the dataset, it will overestimate the value of Q. Incorrect estimation can make the selection of actions fall into areas that are not visible in the dataset. Due to the inability to evaluate the true value of these actions in the real environment, actions with low value may be selected, resulting in a decrease in training results.

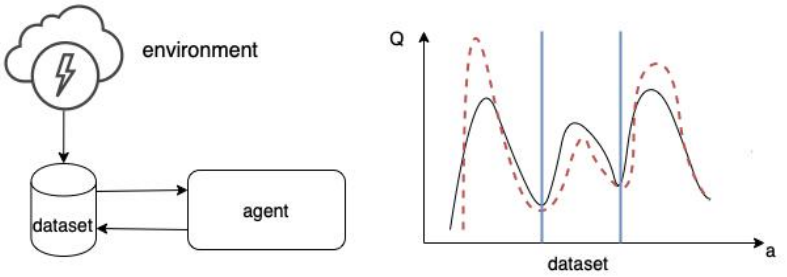

Fig. 1. Offline RL and Extrapolation error（Photo/Picture credit: Original）.

To eliminate extrapolation errors, researchers have conducted extensive research. The current methods can be divided into two types: model-based methods and model-free methods. Model-based methods reduce extrapolation errors by constructing models and generating more training samples [4]. Although it performs better than model-free methods, it has high computational complexity and is greatly limited by the model. The model-free methods directly operate on the update function, which has a wider range of applications [5]. However, although some current optimization methods can significantly address the impact of extrapolation errors, they cannot guarantee the stability of training results.

In this paper, SCQL with the Actor-Critic structure is designed for offline RL. It adopts a value-constrained approach to conservatively estimate the Q value, ensuring the stability of the algorithm's results while not affecting its generalization ability. A VAE [6] and a State-Action-Reward-State-Action (SARSA) [7] style value function to construct penalty terms so that the penalty results are not too conservative.

2 Related Works

- Model-free Offline Reinforcement Learning

Model-free offline RL does not require model construction, making it more flexible and efficient. The existing model-free methods are mainly divided into three types [3]: policy constraints, value function constraints, and uncertainty methods. The method based on policy constraints reduces extrapolation errors by restricting the generation of OOD actions that creates by policy function. Typical methods include Batch-Constrained deep Q-learning (BCQ) [8]. Value constrained methods often limit the estimation of the Q function, such as CQL [9]. Uncertainty methods construct a sufficiently robust Q function to maintain a relatively efficient estimation of OOD actions.

- Generative Models

The generative model generates corresponding output data for the given input data. By analyzing the distribution of inputs, generative model can reconstruct samples which is similar to inputs. Currently, popular generative models include VAE, Generative Adversarial Networks (GAN), flow-based model, and diffusion model. VAE learns the distribution of inputs and reconstructs them through the decoder-encoder structure. GAN includes a generator and a discriminator. It generates realistic data samples through mutual confrontation and training. The flow-based model uses reversible transformations to construct the probability density function of the data. The diffusion model simulates the evolution process of data and generates samples by the reverse model.

3 Stable Conservative Q-Learning

The most direct way to solve the overestimation problem caused by extrapolation error is to directly restrict the Q function. Kumar. A et. al. proposed CQL in 2020 to solve the extrapolation error problem [9]. It introduced limitations to the Q value of the actions that are beyond the dataset. As shown in Fig.2, CQL constrains the Q value of actions beyond the support of the dataset to a relatively reasonable value. The critic network with penalty term is updated as：

$$Loss_Q = \beta(\mathbb{E}_{s_t\sim\mathcal{D},\, a_t\sim\mu(a_t\backslash s_t)}[Q_\xi(s_t,a_t)] - \mathbb{E}_{s_t\sim\mathcal{D},\, a_t\sim\pi(a_t\backslash s_t)}[Q_\xi(s_t,a_t)]) + \frac{1}{2}\mathbb{E}_{(s_t,a_t.s_{t+1},r_t)\sim\mathcal{D},\, a_{t+1}\sim\mu(\cdot\backslash s_{t+1})}[Q_\xi(s_t,a_t) - (r_t + \gamma(min_{j=1,2}Q_{\theta j}(s_{t+1},a_{t+1}))]^2 \quad (1)$$

where μ is the real policy and π is the policy the algorithm learned. The first term of the loss function is the core of CQL, which contains a penalty term and a compensation term. The penalty term limits the Q value of OOD actions while compensating the actions in the dataset.

However, in some cases, CQL's estimate of Q may be too conservative. This is because CQL only guarantees that the valuation is lower than the true value, but there

may be situations where it is too low. The overly conservative estimation depresses the generalization of the algorithm. When estimating the Q value of actions beyond the area that the dataset supports, significant errors occur and the algorithm becomes unstable.

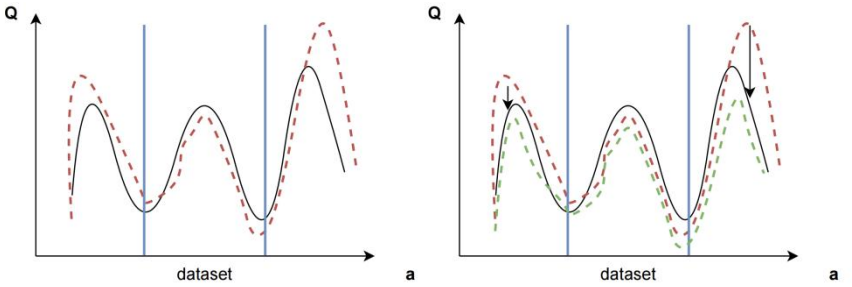

Fig. 2. The constriction on Q function introduced by CQL (Photo/Picture credit :Original).

Lyu et al. proposed Mildly Conservative Q-Learning(MCQ) which can mildly conservatively estimate the Q value [10]. MCQ could prove sufficient conservatism while improving the generalization of CQL. It assigns appropriate pseudo-Q values to OOD actions by training VAE to avoid overestimation problems. The previous model-free method BCQ used VAE to generate data that limited actions not beyond the dataset, but the effectiveness was often not as good as the method of value constrained methods. VAE is divided into two parts: encoder and decoder. Encode is used to recognize inputs, and decoder is used to reconstruct inputs. VAE evaluates the effectiveness through evidence lower bound (ELBO) loss which has two parts. The first part reconstruction loss is used to calculate the similarity between the generated data and the original data, while the second part Kullback-Leible divergence loss is used as an additional loss to guarantee the similar distribution. ELBO can be calculated as:

$$ELBO = \alpha \frac{1}{m} \sum_{i=1}^{m} \|x - y\|^2 + \beta(-0.5) \sum_{i=1}^{n} \left(1 + \log\left(\sigma_i^2\right)\right) - \mu_i^2 - \sigma_i^2) \quad (2)$$

where μ and σ is the mean value and standard deviation of input x respectively. y is the output that reconstructed by decoder.

Generative Value Constraint: Inspired by MCQ, the behavior policy is reconstruct by using VAE. The key idea is that VAE can be trained to generate actions that don't beyond the area the datatset supports. In this way, OOD actions can be converted into actions that conform to the distribution of the dataset. Although some errors may still occur because the next step state used for generation is not used in training, the results have remained relatively effective. This is much more conservative than directly sampling OOD actions and estimating them. VAE is optimized by the following function.

$$L_{VAE} = \mathbb{E}_{(s,a)\sim\mathcal{D},\, z\sim E(s,a)}[(a - D(s,z))^2 + KL(E(s,a), \mathcal{N}(0,I)] \quad (3)$$

where D and E represent the decoder and encoder of VAE. z is the output of encoder and (s,a) is the state action pairs in dataset.

Meanwhile, a SARSA-style value function calculation method is adopted . Specifically, modeling a state value function network V separately. To avoid accessing OOD actions, this value function network directly estimates the state values. In the general computing process, target value $y = r_t + \gamma min_{j=1,2} Q_{\theta j}(s_{t+1}, a_{t+1})$. Since a_{t+1} is generated by an actor network, it may be an OOD action. The OOD actions can be avoided to join the computing process by using V network to $y = r_t + \gamma V(s_{t+1})$. The V network is optimized as follows

$$L_V = \mathbb{E}_{(s,a)\sim\mathcal{D}}[(0.9(1-v_f) + 0.1v_f)(V(s) - min_{i=1,2} Q_{\theta i}(s,a)^2 \quad (4)$$

where $Q_{\theta i}$ is the estimation of target_critic network. v_f is defined as:

$$v_f = \begin{cases} 1, V(s) - min_{i=1,2} Q_{\theta i}(s,a) > 0 \\ -1. V(s) - min_{i=1,2} Q_{\theta i}(s,a) < 0; \end{cases} \quad (5)$$

By combining VAE and V network, the penalty term is described as：

$$L_P = \beta \mathbb{E}_{s_t, s_{t+1},\, r\sim\mathcal{D},\, a^{'}\sim VAE(a)}[(log \quad _{a^{'}} exp(Q_\xi(s_t, a^{'})) - (r + \gamma V(s_{t+1})] \quad (6)$$

where $a^{'}$ is the action generated by VAE, (s_t, s_{t+1}, r) is (action, next action, reward) from dataset.

SCQL Algorithm: SCQL adopts an Actor-Critic structure like CQL. The Actor-Critic structure has a fast rate of convergence and updates through value function. Thus, the designed generative value constraint could apply to it. SCQL consists of seven networks, including two Citric networks, 2 target Citric networks, one actor network, one VAE generator, and one state value network. Fig. 3 shows the computing process of SCQL. And the critic network in SCQL is updated as：

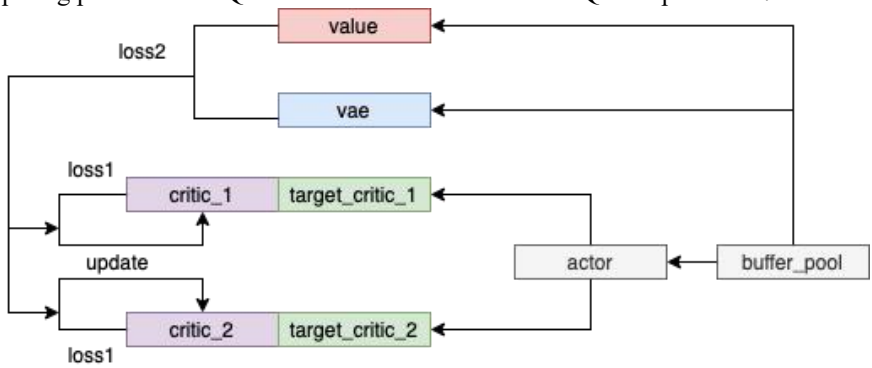

Fig. 3. The framework of SCQL(Photo/Picture credit :Original).

$$L_c = L_p + \frac{1}{2}\mathbb{E}_{(s_t, a_t, s_{t+1}, r_t)\sim\mathcal{D},\, a_{t+1}\sim\mu(\cdot\backslash s_{t+1})}[Q_\xi(s_t, a_t) - (r_t + \gamma(min_{j=1,2}\, Q_{\theta j}(s_{t+1}, a_{t+1}))]^2 \quad (7)$$

where L_p is the value function constraint in this work. Table 1 is the pseudo code of SCQL.

Table 1. Algorithm 1: Stable conservative Q-learning

Algorithm 1: Stable conservative Q-learning

Input: (s_t, a_t, r_t, s_{t+1}) in buffer pool D.

1: Initialize value network V, critic network $Q_{\xi 1}$, $Q_{\xi 2}$, VAE network and actor network π with random parameters.

2: **Copy parameter from** critic network $Q_{\xi 1}$, $Q_{\xi 2}$ to target critic network $Q_{\theta 1}$, $Q_{\theta 2}$

3: **while** t∈(1,K) **do**

4: Train VAE by equation 3

5: Train value network V by equation 4

6: Compute target value: $t = r_t + \gamma(min_{j=1,2}Q_{\theta j}(s_{t+1}, a_{t+1}) - \alpha log_{\pi}(a_{t+1}|s_{t+1}), a_{t+1} = \pi(\cdot|s_{t+1})$.

7: Compute generated value: $y_g = r_t + \gamma V(s_{t+1})$

8: Update critic network by equation 6

9: Update actor network: $\pi: = max\mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi(\cdot|s)}[min_{j=1,2}Q_{\theta j}(s_t, a_t) - \alpha log_{\pi}(\cdot|s_t)$.

10: Update target critic network: $\theta_j = \lambda\xi + (1-\lambda)\theta_j, j = 1,2$

11: **end while**

4 Experiments

4.1 Backgrounds

Environment: OpenAI Gym is a toolbox for developing and comparing RL algorithms. A classical continuous control environment named “Pendulum” is chosen to conduct the experiments.

Dataset: The Soft Actor Critic(SAC) algorithm is used to train the agent in the Pendulum environment [11]. The total epoch of training is 200. The data during training is collected as an offline dataset. The total data in the whole dataset is 20000. Fig.4. depicts the returns of off-policy reinforcement learning based on SAC. It can be seen that the off-policy approach can learn excellent strategies, but there are still unstable fluctuations in the 180th epoch.

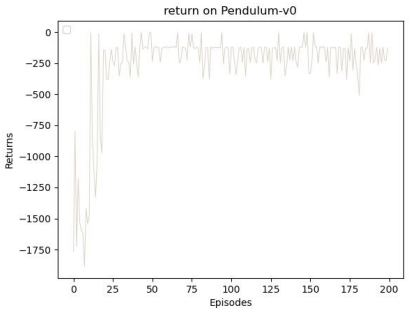

Fig. 4. Returns of SAC that is trained for dataset (Photo/Picture credit : Original).

Settings: The experiments are conducted on the virtual workstation with an 8G memory Apple M1 CPU. Learning rate(*lr*): For the network related to actions like actor network and VAE, *lr* is set to 0.0003. For value network and critic network which is related to Q value, *lr* is set to 0.003.

4.2 Experiment Results

Firstly, the results of unoptimized algorithm SAC, model-free baseline method CQL, and SCQL in an offline environment are compared . Fig.5. compares the returns of SCQL with SAC and CQL respectively. In the experimental environment of this paper, all three algorithms train 300 epochs, with each epoch taking 500 steps. SCQL can consistently achieve good results after convergence. Compared with SAC, SCQL has a slower rate of convergence, which requires 50 epochs for stable convergence. However, the results of SCQL are more stable and there are fewer burrs on the curve. Compared with CQL, although the final results of SCQL and CQL are very similar, it can be seen from the curve that CQL occasionally shows very poor performance at some points, which also proves the stability of SCQL.

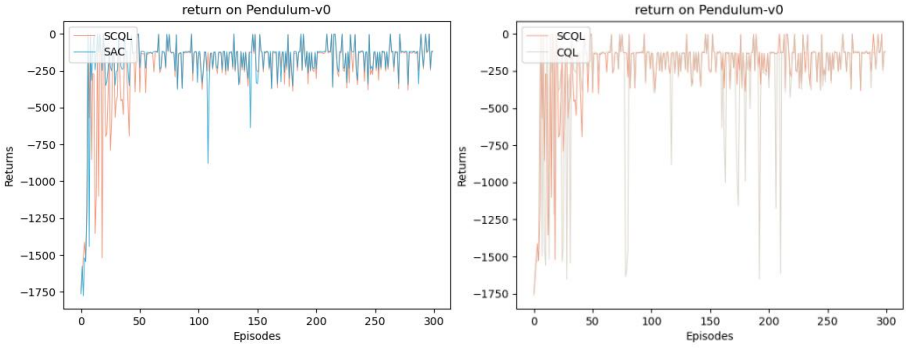

Fig. 5. Returns of SCQL, SAC and CQL (Photo/Picture credit : Original).

Furthermore, SCQL has better results when compared with the model-based offline reinforcement learning method (MOReL) [12]. The result of SCQL and MOReL is

shown in Fig.6. In the first 30 epochs, the fluctuation of MOReL is much smaller than that of SCQL. However, after entering the convergence stage, the performance of SCQL is significantly superior to that of MOReL. The worst result for SCQL is around -250, while the worst result for MOReL is -700. And the mean result of SCQL is -200, which is better than the -400 of MOReL. The incomplete model built by MOReL makes its results lower than SCQL. If a superior model can be established, MOReL can achieve very good results. However, establishing a superior model will incur additional computational overhead. Compared with MoReL, SCQL can achieve better results and efficiency than MOReL without spending a lot of resources.

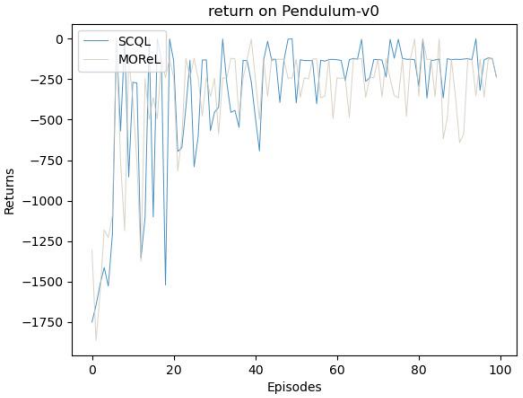

Fig. 6. Returns of SCQL and MOReL (Photo/Picture credit : Original) .

To further investigate the stability of SCQL, the relationship between the rewards obtained by the algorithm and the number of training epochs is investigated. In this experiment, SCQL is trained for 100, 300, and 500 epochs respectively. It can be observed (Fig. 7) that as the number of epochs increases, the rewards obtained by SCQL are relatively stable and do not fluctuate significantly. The algorithm converges quickly and remains stable after training for 50 epochs. This is because the use of VAE ensures that the generated actions have a similar distribution to the actions in the dataset and do not deviate too much. And using the V network to avoid the use of OOD actions generates relatively accurate estimates. The combination of the two modules as punishment term can not only suppress overestimation of OOD actions, but also ensure that the suppression is not too conservative, which affecting universality and causing instability.

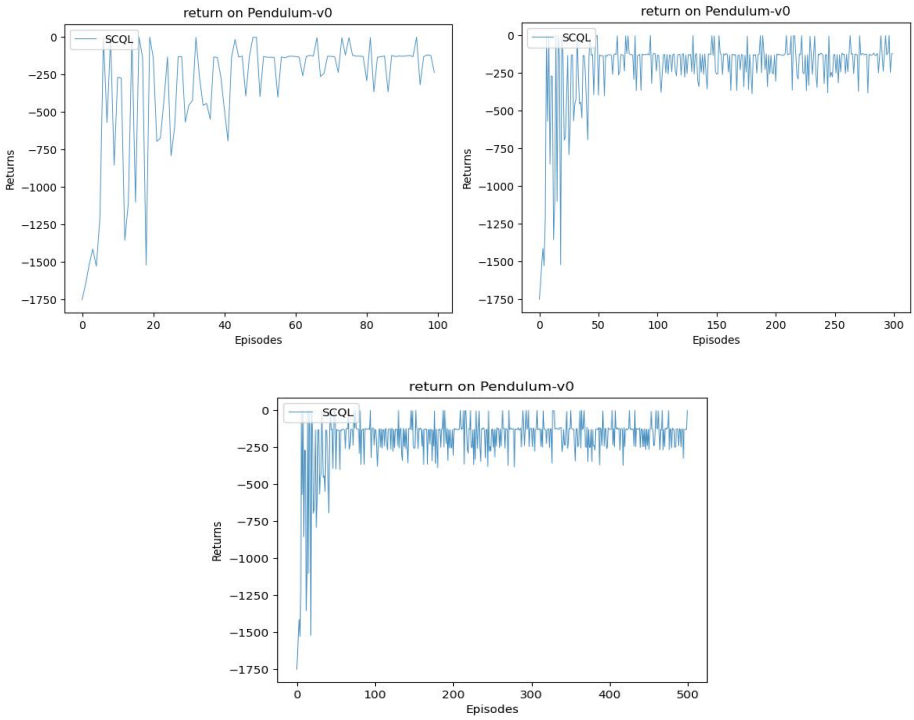

Fig. 7. Relationship of episodes and returns of SCQL (Photo/Picture credit : Original).

5 Conclusion

In this paper, an offline RL method named SCQL is proposed to solve the problem in offline reinforcement learning. SCQL has two improvements. Firstly, SCQL imporves the update function of CQL, which makes it can maintain excellent performance while making the algorithm's results more stable. It is because that the value constraint method of CQL is effective but overly conservative. And this kind of constraints affect the generalization of the algorithm. Secondly, a VAE and a value function network are combined to construct penalty term and estimate the value of Q relatively conservative. In this work, three experiments are conducted to test the effectiveness of SCQL. In the first two experiments, the experimental results of SCQL are compared with the results of the model-free baseline method CQL and the model-based baseline method MOReL respectively. It shows that SCQL can achieve better performance in the rate of convergence and final results. In addition, the third experiment shows that the convergence effect of SCQL is very stable and will not change significantly with the increase in training times. This indicates that the value constraint of SCQL is effective, as it reduces the over-conservative penalty for OOD action and enhances the stability of the algorithm. Of course, the scenarios for the

algorithm testing in this work are still relatively limited at present. Thus, more experiments will be conduct to improve SCQL in the next works.

References

1. Mnih. V, Kavukcuoglu. K, Silver. D, et al. “Playing Atari with Deep Reinforcement Learning”. Computer Science, 2013.
2. Arulkumaran.K, Deisenroth. M. P, Brundage. M, et al. “Deep Reinforcement Learning: A Brief Survey”. IEEE Signal Processing Magazine, 2017, 6, pp.34–42.
3. Levine. S, Kumar. A, Tucker. G, et al. “Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems”. 2020.DOI:10.48550/arXiv.2005.01643.
4. Moerland. T. M, Broekens. J, Jonker. C. M. “Model-based Reinforcement Learning: A Survey”. Foundations and Trends in Machine Learning, 2023, 16, pp. 1-118.
5. Fujimoto. S, Gu. S. S. “A Minimalist Approach to Offline Reinforcement Learning”. in Neural Information Processing System 2021, 34, pp: 20132-20145.
6. Kingma. D. P, Welling. M. “Auto-Encoding Variational Bayes”. International Conference on Learning Representation. 2014.
7. Rummery. G. A, Niranjan. M. “On-Line Q-Learning Using Connectionist Systems”. Technical Report, 1994.
8. Fujimoto .S, Meger. D, Precup .D. “Off-Policy Deep Reinforcement Learning without Exploration”. Proceedings of the 36th International Conference on Machine Learning, 2019, pp: 2052-2062.
9. Kumar. A, Zhou. A, Tucker. G, et al. “Conservative Q-Learning for Offline Reinforcement Learning”. in Neural Information Processing System 2020, 33.
10. Lyu. J, Ma. X, Li. X, Lu. Z, “Mildly Conservative Q-Learning for Offline Reinforcement Learning”. Neural Information Processing Systems, 2022, 35.
11. Haarnoja. T, Zhou. A, Abbeel. P, et al. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. International conference on machine learning, 2018, pp: 1862-1870.
12. Kidambi. R, Rajeswaran. A, Netrapalli. P, et al. “MOReL: Model-Based Offline Reinforcement Learning”. in Neural Information Processing System 2020, 33.