Master's Thesis

Penalty Relaxation Techniques for Efficient Offline Reinforcement Learning

Junghyuk Yeom

Graduate School of Artificial Intelligence

Ulsan National Institute of Science and Technology

2024

Penalty Relaxation Techniques for Efficient Offline Reinforcement Learning

Junghyuk Yeom

Graduate School of Artificial Intelligence

Ulsan National Institute of Science and Technology

Thesis/Dissertation Title

Penalty Relaxation Techniques for Efficient Offline Reinforcement Learning

A thesis/dissertation submitted to
Ulsan National Institute of Science and Technology
in partial fulfillment of the
requirements for the degree of
Master of Science

Junghyuk Yeom

12.06.2023 of submission

Approved by

Advisor

Seungyul Han

Thesis/Dissertation Title

Penalty Relaxation Techniques for Efficient Offline Reinforcement Learning

Junghyuk Yeom

This certifies that the thesis/dissertation of Jung-hyuk Yeom is approved.

12.06.2023 of submission

Signature

_____

Advisor: Seung-yul Han

Signature

_____

Hyeon-woo Yu

Signature

_____

Sung-whan Yoon

Abstract

In reinforcement learning, an agent receives observations from the environment, makes decisions based on these observations, and receives rewards for certain actions. By repeatedly experiencing trial and error through this process, the agent learns a policy that yields higher rewards. However, traditional reinforcement learning models heavily depend on continuous interaction with the environment for new data acquisition. This reliance often results in significant time, cost, and even safety concerns, particularly in real-world applications like robotics and autonomous vehicles. To address these challenges, offline reinforcement learning (Offline RL) emerges as a viable alternative. By utilizing pre-collected data, Offline RL greatly diminishing the need for continuous data collection and thereby reducing time and safety risks. This approach provides a safer and more controlled learning environment, as it depends on data that is already available.

However, Offline RL encounters specific issues due to its reliance on a fixed dataset. A notable challenge emerges when the pre-existing data greatly differs from the target policy to be learned, leading to out-of-distribution problems that present significant obstacles in effectively implementing offline reinforcement learning. To address this problem, it is crucial for offline RL algorithms to be designed in a conservative way that can make the learned policy close with the behavior policy. Our baseline method, Conservative Q-Learning [1] addresses this challenge by applying penalties to state-action pairs generated by the policy. This approach enables the learning of a conservative Q-function that serves as a lower bound for the true value function. Yet, this method might lead to performance degradation from excessive constraints. Moreover, if the imposed penalties do not accurately reflect the dataset's characteristics, the algorithm's performance may become too dependent on the quality of the batch data.

In this paper, we propose a penalty relaxation technique by analyzing the penalty characteristics suggested in conservative Q-learning. By adjusting penalties to align with the batch data's characteristics, we can reduce reliance on the dataset for performance, thereby boosting the efficiency of Offline RL. This increased efficiency is achievable with a smaller number of networks, ensuring greater effectiveness. Furthermore, to tackle the suboptimal aspects of batch data, we propose an strategy for updating the behavior policy, advancing beyond simple replication of the current policy. Our method focuses on learning the behavior policy in a way that aligns with the quality of the batch data. This technique involves the categorize states and predict a more optimized policy. By incorporating it in Bellman updates and conservative Q-learning, we can enhance the performance of the behavior policy and mitigate the bias issues inherent in the dataset.

Contents

List of Figures

I Introduction

Machine learning is a subfield of artificial intelligence, comprising algorithms and technologies that enable learning from data and making predictions or decisions. At its core, machine learning is founded on pattern recognition and computational learning theory, and is primarily divided into supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training models with labeled data for making predictions on new data, while unsupervised learning uses unlabeled data to uncover hidden patterns or structures in the data, including clustering and dimensionality reduction. In our main research area of reinforcement learning, the agent receives observations from the environment, makes decisions based on these observations, and receives rewards for specific tasks. Through this process, the agent learns a policy that yields higher rewards. By repeatedly experiencing trial and error through this process, the agent learns a policy that yields higher rewards. Reinforcement learning has found practical applications in various domains. For example, in robotics, reinforcement learning algorithms have been used to teach robots how to perform tasks such as picking and placing objects with precision. In autonomous vehicles, reinforcement learning can be employed to train self-driving cars to navigate complex traffic scenarios safely and efficiently. These applications highlight the potential of reinforcement learning to revolutionize industries and improve our daily lives by enabling machines to learn and adapt to their environments autonomously. However, traditional reinforcement learning models heavily depend on continuous interaction with the environment for new data acquisition. This reliance often results in significant time, cost, and even safety concerns, particularly in real-world applications like robotics and autonomous vehicles.

To address these challenges, offline reinforcement learning (Offline RL) emerges as a practical solution. Unlike traditional reinforcement learning, Offline RL relies solely on pre-collected data, eliminating the need for real-time interaction with the environment. This approach significantly reduces the continuous data collection requirements by capitalizing on existing datasets. Consequently, it not only shortens the training time but also mitigates the challenges of learning from scratch, as Offline RL leverages valuable insights from available data. These advantages hold immense significance in the application of reinforcement learning to real-world industrial settings. In scenarios like construction sites or autonomous driving, conducting experiments in the actual environment can pose safety concerns. In contrast, Offline RL ensures that agent training occurs in a secure and controlled environment, whether through simulation or the utilization of pre-existing data, effectively addressing safety risks. Hence, research in Offline RL plays a pivotal role in advancing reinforcement learning, making it more practical and adaptable across a wide range of real-world scenarios.

However, to effectively deal with offline reinforcement learning, one must carefully consider the challenges posed by a fixed dataset. During training, the agent learns how to maximize the value of its current state and actions. However, when the pre-collected data significantly deviates from the learning policy, accurately predicting actions becomes difficult, leading to estimation errors. Unlike online reinforcement learning, where estimation errors can gradually decrease through interaction with the environment, offline reinforcement learning operates with limited data, making estimation errors more

critical. Particularly, when estimations are over-optimistic, these errors can propagate even more significantly, especially when the value is underestimated during value function learning. To tackle this issue, it is crucial for offline reinforcement learning algorithms to be designed conservatively, ensuring that the learned policy closely aligns with the data collection policy. Over time, offline reinforcement learning algorithms have evolved, incorporating various approaches, including constraining policies to match behavioral policies, measuring uncertainty in value functions, and utilizing batch-based supervised learning.

The foundational paper that introduced offline reinforcement learning, Behavioral Cloning Q-learning (BCQ) [3], delineated the challenges that emerge when moving from a traditional online setting to employing a constrained offline dataset. BCQ highlighted the issue that in Q-learning, extrapolation error can accumulate through the Bellman equation, positing that the target policy must be constrained towards the behavior policy. Hence, BCQ introduced the use of a generative model, specifically a Variational AutoEncoder, to approximate the behavior policy and demonstrated that adding a small amount of noise to the generated action can result in a policy that surpasses behavior cloning techniques. "Furthermore, the Bootstrapping Error Accumulation Reduction (BEAR) [4] and the Behavior Regularized Actor Critic (BRAC) [5] algorithms argue that constraining the target policy solely to the distribution of the behavior policy can lead to overly restrictive policy learning. Therefore, in BEAR, the maximum mean discrepancy distance is utilized to constrain the target policy not to the distribution of the behavior policy, but to its action space, thereby expanding the potential action space beyond what is typically allowed by batch data-centric methods. Specifically, BEAR projects the policy distribution into a Hilbert space through a kernel and then compares the distance in the projection space, effectively penalizing actions that occur outside of the data distribution. The subsequent paper, BRAC, sets a broader scope by providing a comprehensive framework for the conceptualization and execution of behavior regularization, thereby upgrading the previously proposed BEAR method. Our baseline algorithm, Conservative Q-learning (CQL) [1], employs a policy constraint approach to address this issue by penalizing state-action pairs generated by the policy, thus learning a conservative Q function that underestimates the true values. This method is highly effective and performs well in offline RL setups. However, it uniformly penalizes all policy actions without analyzing the characteristics of the data. This uniform penalty imposition can lead to excessive constraints, potentially resulting in performance degradation. Furthermore, if the imposed penalties do not accurately reflect the dataset's characteristics, the algorithm's performance may become overly dependent on the quality of the batch data.

In this paper, we have analyzed the penalty characteristics of Conservative Q-learning and presented necessary and sufficient conditions for penalties required to lower the actual state-action predictions. Based on these conditions, we adaptively impose penalties by adjusting them according to the characteristics of the batch data. This approach reduces the performance dependency on the quality of the dataset, enhancing the efficiency of offline reinforcement learning while providing a benchmark for penalty imposition. Notably, we achieved improved efficiency with a minimum number of networks, comparable to state-of-the-art algorithms. Furthermore, we argue that the target policy doesn't need to rely entirely on the current policy to address the suboptimal aspects of batch data. In real datasets, noisy data can

significantly disrupt offline learning. To mitigate this, we propose a behavior policy update approach that allows the target policy to determine its dependence on the current policy based on the quality of the batch data. This technique aims to classify states, predict values with respect to the current policy, and learn a more optimized policy. As a result, our experiments show performance enhancements in the behavior policy itself, and the reliance of the target policy on this behavior policy helps mitigate bias issues inherent in the dataset.

Following this paper, in Section 2, we provide the background information necessary to understand our work. We explain the update mechanisms in reinforcement learning and highlight the distinctions compared to traditional RL when considering Offline RL. In particular, the Offline RL section includes explanations of extrapolation and out-of-distribution, which form the baseline for our ideas, and the conservative Q-learning theorem. In Section 3, we present theoretical proofs of our proposed ideas and analyze the advantages compared to existing baselines. Finally, Section 4 contains the experimental results of our approach, along with an analysis of the results for each component.

II Background

2.1 Reinforcement Learning

Reinforcement Learning (RL), includes decision-making processes where a virtual agent observes a state in a given environment and selects actions influencing the environment. Specifically, the agent chooses the best actions based on a learned policy while interacting with the environment, which in return provides new states and rewards. The goal of the learned policy is to predict information about the environment and maximize obtainable rewards. Therefore, in reinforcement learning, the policy is crucial as it determines the actions an agent takes. In this context, we assume that the current state information follows the Markov Property for efficient learning and decision-making. The Markov Property implies that the probability of the next state depends only on the current state, not on the sequence of events that preceded it.

$$P(s_{t+1}|s_t) = P(s_{t+1}|s_1, s_2, \ldots, s_t) \tag{1}$$

Here, $P(s_{t+1}|s_t)$ represents the transition probability of moving to the next state s_{t+1} given the current state s_t. This allows us to interpret the problem we aim to solve as a Markov Decision Process (MDP) environment (S, A, P, γ, R), where S represents the state space, A the action space, P the transition probability, γ the discount factor, and R the reward function. By assuming a Markov Decision Process (MDP) in reinforcement learning, we can presume that each state is independent of the history of previous states and actions. This allows the agent to consider only the current state to determine the optimal action, thereby simplifying the modeling of the problem at hand. Additionally, MDP makes the behavior of the environment predictable, enabling the agent to engage in more stable and reliable learning.

When learning a policy, the most crucial aspect to consider is the balance between exploration and exploitation. Exploration offers the chance to discover better strategies by trying new actions, while exploitation involves using known information to make decisions that yield the maximum reward. Thus, a policy overly focused on exploration can lead to reduced immediate rewards, and excessive focus on exploitation might miss out on new opportunities for higher rewards. Therefore, in reinforcement learning, the value function is used to predict future information, helping the agent find the optimal path. The State Value Function numerically evaluates a specific state's value, representing the total expected future rewards under the policy followed by the agent in a given state s. The Action Value Function $Q(s,a)$ denotes the total expected future rewards for taking a specific action a in a specific state s. This function assesses the value of performing an action in a certain state.

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}[R_t|s_t = s] \\ Q^{\pi}(s,a) &= \mathbb{E}[R_t|s_t = s, a_t = a] \end{aligned} \tag{2}$$

Here, π represents the policy, R_t the discounted return $\sum_{t=0}^{t} \gamma^t r_t$ at time t, s_t the state at time t, and a_t the action at time t.

Therefore, in the process of learning the value function, we can propagate the information from future states into the current value function, allowing us to predict potential outcomes. This step enables

us to systematically and mathematically model the decision-making process using the Bellman Equation. The essence of this approach lies in the iterative nature of the updates, which is represented through the application of the Bellman Operator B. The Bellman equations for the state-value function $V(s)$ and the action-value function $Q(s,a)$ are defined as follows:

$$B^{\pi}V(s) = \sum_{a} \pi(a|s) \sum_{s',r} P(s',r|s,a)[r + \gamma V^{\pi}(s')] \tag{3}$$
$$B^{\pi}Q(s,a) = \sum_{s',r} P(s',r|s,a)[r + \gamma \sum_{a'} \pi(a'|s') Q^{\pi}(s',a')]$$

Here, $\pi(a|s)$ represents the probability of selecting action a in state s according to the policy, and a' denotes all possible actions that can be chosen in the next state s'. $P(s',r|s,a)$ is the probability of transitioning to state s' and receiving a reward r after taking action a in state s. γ is the discount factor, which determines the present value of future rewards.

From the Equation 3, when propagating information about the next state, we need to consider the action to be taken in that state. In reinforcement learning, the choice of action in the next state is a key factor that broadly classifies the approach into on-policy and off-policy. In on-policy algorithms, the agent selects actions directly based on the policy it is currently learning, using this to improve the value function. Conversely, in off-policy algorithms, the agent can choose actions based on a different policy but learns a separate optimized policy. A typical example of an on-policy algorithm is SARSA, which updates the action-value function $Q(s,a)$ based on the reward of the actual action taken.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \tag{4}$$

Here, α is the learning rate, γ the discount factor, r the reward, (s',a') represent the next state and action actually taken. SARSA adjusts the Q values based on the specific actions taken by the agent, fostering a cautious and considered learning methodology that avoids higher-risk actions. On the other hand, Q-learning systematically enhances the action-value function $Q(s,a)$ for each state-action pair by utilizing the highest anticipated future reward, demonstrating a strategy focused more on maximizing potential gains.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{5}$$

Here, a' represents all possible actions, different from those in Equation 4. Q-learning updates the Q value based on the action that provides the maximum reward, irrespective of the action actually taken by the agent, thus facilitating an automatic balance between exploration and exploitation.

Q-learning and SARSA, while being prominent algorithms in reinforcement learning, each have distinct issues concerning stability and convergence. Q-learning often faces overestimation problems where the maximum Q value used for updating current Q values can lead to instability in learning. Additionally, when combined with function approximators in practical applications, Q-learning may lose its theoretical convergence guarantees. On the other hand, SARSA, an on-policy method, tends to be more conservative due to its reliance on the current policy for learning, potentially limiting its ability to discover the optimal policy. To address these challenges, the Actor-Critic method was proposed,

introducing a dual-component system with an "Actor" for decision-making based on the policy and a "Critic" for evaluating the value of actions, thereby offering a more balanced and effective approach to learning in reinforcement environments [6]. The Actor-Critic method allows for a more focused approach by clearly defining the roles of the Actor and the Critic, which can enhance the overall stability and convergence of the learning process. Furthermore, the Actor-Critic approach is particularly well-suited for problems with continuous action spaces.

The Actor-Critic approach encompasses two main categories: value-based and policy-based methods. In value-based methods, the value function is essential, necessitating the actual learning and computation of values. For examples, The Deep Q-Network (DQN) [7] extends traditional Q-learning using deep neural networks. DQN was the first to use deep learning to model the action-reward system and is particularly effective in processing high-dimensional input data, such as video game screens. DQN employs the experience replay and fixed target Q-networks for stable off-policy learning, enabling it to learn value functions with deep neural networks. These methods allow the agent to reuse past experiences for more efficient learning and mitigate the problem of correlations in data encountered during training. Also, Deep Deterministic Policy Gradient (DDPG) [8] builds upon the foundation of DQN by applying its concepts to continuous action spaces, based on the actor-critic methodology. The use of deterministic policies in DDPG enables more stable and efficient learning in these continuous action domains. This approach marks a significant advancement in reinforcement learning, particularly in handling complex, high-dimensional environments where actions are not discrete.

On the other hands, the policy optimisation methods represent the policy explicitly by mapping directly from state to action probability. This is in contrast to value-based methods, which infer the policy from the value function in an indirect manner by comparing different action-values. Trust Region Policy Optimization(TRPO) [9] and Proximal Policy Optimization(PPO) [10] are both advanced reinforcement learning algorithms that focus on stable and efficient policy updates. The commonality between them lies in their approach to managing policy changes: both aim to adjust the policy in a controlled manner to prevent drastic performance drops. While TRPO uses a trust region to limit policy updates, ensuring they don't deviate too much from the previous policy, PPO distinguishes itself with a unique clipped objective function. This function penalizes excessive changes in the policy update, effectively restricting updates within a safe range. This clipping mechanism is central to PPO's design, maintaining the balance between exploration and stability. These similarities and PPO's distinctive feature reflect their shared goal of achieving stable and reliable policy improvement in complex environments.Another example is Twin Delayed Deep Deterministic Policy Gradients (TD3) [11]. TD3 enhances reinforcement learning in continuous action spaces with twin Q-networks and delayed policy updates. The twin Q-networks address overestimation bias by taking the minimum estimate from both networks, leading to more accurate value assessments. The delayed policy updates, occurring less frequently than value network updates, contribute to stability in learning. Additionally, target policy smoothing, which adds noise to target policies, further enhances learning stability. These features make TD3 efficient and stable, particularly suited for tasks requiring precise control in high-dimensional spaces.Lastly, Soft Actor-Critic (SAC) [12] is a state-of-the-art reinforcement learning algorithm designed for environments with con-

tinuous action spaces. It is an off-policy algorithm that combines the strengths of actor-critic methods with the principles of entropy maximization, leading to a balanced approach between exploration and exploitation. SAC focuses on maximizing not just the expected return but also the entropy of the policy. Entropy, in this context, refers to the randomness of the policy's actions. By maximizing entropy, SAC encourages the policy to explore a wide range of actions, leading to more robust and effective learning, especially in complex environments. The balance between exploration (through entropy maximization) and exploitation (through effective policy evaluation) makes SAC a powerful tool for complex reinforcement learning tasks, particularly those involving high-dimensional, continuous action spaces.

2.2 Offline RL

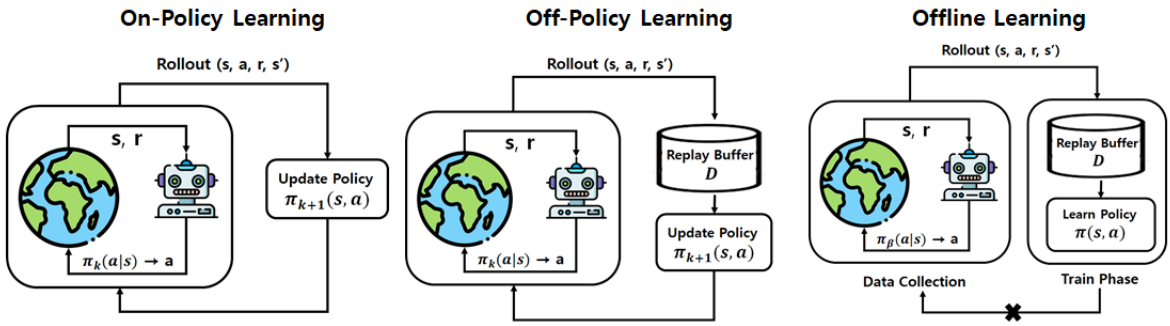


Figure 1: A pictorial illustration of (a) online reinforcement learning, where learning occurs from a policy actively interacting with the environment, (b) off-policy reinforcement learning, where the learning policy may differ from the one interacting with the environment, and (c) offline reinforcement learning, which involves learning solely from collected data without any further interaction with the environment.

In traditional reinforcement learning, agents learn through continuous interaction with the environment. During this process, the agent balances exploration and exploitation, learning the optimal action policy based on real-time data. However, this method has two major constraints: firstly, real-time interactions can entail significant costs and risks. For example, training autonomous driving systems requires vast amounts of data and experience from real road environments. However, experiments on actual roads carry the risk of traffic accidents and involve considerable time and expense, and collecting data in specific traffic situations or weather conditions can be even more challenging. Secondly, conducting appropriate exploration in actual complex environments can be difficult. In the field of financial investment, the market provides an extremely complex and unpredictable environment. In online reinforcement learning, exploring new investment strategies in real-time with the agent interacting with the market is high-risk, and using real money to test experimental strategies can lead to significant losses. Therefore, the goal of offline reinforcement learning is to develop and validate strategies with minimized risk using collected data.

Offline reinforcement learning, or batch reinforcement learning, is essential as a complement to traditional online reinforcement learning methods, especially in overcoming the limitations of the latter.

Fig. 1 illustrates the differences between traditional Reinforcement Learning (RL) and Offline Reinforcement Learning (Offline RL). In Online RL, agents interact with the environment in real-time during learning. This methodology allows agents to learn directly from their current situations, balancing exploration and exploitation to develop the optimal action policy. Real-time data collection and immediate feedback enable agents to continuously adjust and improve their actions. However, this approach can be limited in environments that involve high costs or risks. In Off-policy RL, agents collect data through ongoing interactions with the environment based on their current policy (behavior policy) and store this data in an experience replay buffer. This data is then used to optimize the policy that the agent wants to learn (target policy). This method allows agents to utilize experiences from other agents, enhancing data efficiency and enabling learning in a variety of situations. Offline RL, on the other hand, trains agents solely using pre-collected datasets. The most contrasting feature compared to traditional RL is that learning occurs without real-time interaction with the environment. The goal is to utilize data collected from various sources, allowing the agent to effectively learn and generalize across different scenarios and situations.

Offline RL can be compared with imitation learning RL, which primarily focuses on replicating actions performed by experts. Imitation learning is typically used in scenarios where the correct behavior is challenging for the agent to discover independently, such as robots performing complex tasks or learning specific movements. In contrast, offline reinforcement learning utilizes various policies derived from data, aiming to make optimal decisions for maximum reward. Thus, offline RL has several advantages over imitation learning. Firstly, offline RL can gather data from a diverse range of sources, allowing for more generalization across different scenarios. Secondly, through optimization, offline RL has the potential to exceed the performance of experts. While imitation learning is limited to the level of expert performance, offline RL can achieve better outcomes. Thirdly, offline RL is more flexible in complex and challenging environments, especially when the reward structure is well-defined, making it more advantageous in operation.

Offline reinforcement learning employs a fixed dataset, which may exhibit differences in the MDP model compared to the actual environment. While the interaction with the environment during online learning phases leads to an increase in the amount of collected data, reducing the disparity between the dataset and the real-world MDP model, offline reinforcement learning deals with a fixed dataset size, necessitating measures to address such distribution differences. These errors resulting from distribution differences are commonly referred to as "extrapolation errors". Extrapolation errors primarily stem from inaccurate predictions of state-action pairs and biased transition probabilities in the context of offline RL [3, 13]. Firstly, when a specific state-action pair (s, a) is absent from the dataset, the estimation of the Q function becomes significantly inaccurate, and the variance of predictions increases as the dataset differs more from the true distribution. Secondly, in the process of learning the value function, we propagate the estimated value function of the next state using the Bellman operator B. In situations involving a stochastic Markov Decision Process (MDP), if there's limited exploration leading to finite state-action visitations, it can result in skewed estimates of the transition probabilities. This issue particularly arises with actions that fall outside the scope of the batch data, commonly referred

to as out-of-distribution actions. In traditional machine learning, extrapolation errors can result in both overestimation and underestimation, and they tend to average out to near-zero as the data quantity increases. However, in reinforcement learning, overestimation is more detrimental than underestimation, as policies aim to maximize the value function by selecting actions with the highest expected value in the next state. As these errors accumulate, they lead to a discrepancy between the distribution of the behavior policy and the target policy, which we refer to as the 'Distribution Shift' problem.

Currently, offline reinforcement learning has developed with three major approaches to handle out-of-distribution actions and minimize risks: penalizing out-of-distribution actions, predicting uncertainty in the next state's transition from the current state, and adopting supervised learning techniques that pre-calculate returns from batch data and determine the behavior of the learning policy. Detailed explanations of each approach are provided below.

2.2.1 Penalty-based RL

The goal of Penalty-based Offline Reinforcement Learning (RL) is to prevent the occurrence of out-of-distribution actions by predicting the behavior policy and then constraining the target policy based on the predicted behavior policy. This is based on the most traditional method of behavior cloning. Various methods proposed so far include BCQ (Batch-Constrained deep Q-learning) [3], BEAR (Bootstrapping Error Accumulation Reduction) [4], and BRAC (Behavior Regularized Actor Critic) [5], which impose penalties based on the distance between policies. Additionally, there is the CQL (Conservative Q-Learning) [1] method, which performs a conservative value assessment for actions generated from the target policy.

Batch-Constrained deep Q-learning(BCQ) [3] was initially proposed in a paper addressing offline problems, providing an analysis of extrapolation error and batch constraint learning. A bias emerges when using restricted data, due to the possibility that the expected outcomes based on transitions in the batch D might not align with those of the actual MDP.

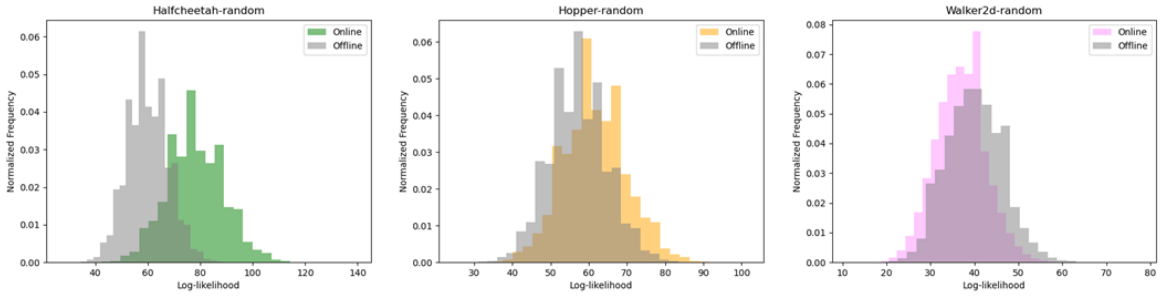


Figure 2: An example of distribution shift in the Mujoco [2] locomotion environment. We compare of the log-likelihood between data from offline training and the data generated from online interactions by an agent trained offline.

Figure 2 illustrates the difference in log-likelihood of the policy when using an actual offline dataset for training and then rolling out the data in an online environment. Despite sampling data using the

same policy, there is a clear difference between the offline and online datasets. This intuitively suggests that the distribution shift occurs due to out-of-distribution actions performed in the online interaction process, particularly for states not present in the data. This discrepancy is likely to intensify as the size of the offline dataset decreases. The concept of extrapolation error can be articulated more precisely. This error, denoted as ε_{MDP}, represents the variance between the value function Q^{π}, which is obtained from the genuine MDP M, and the value function Q_D^{π}, which is calculated using the batch dataset D in the approximated MDP M_D that is constructed from said batch data. This relationship is established in the following theorem:

Theorem 1 (Probagation of extrapolation error [3])**.** *When performing Q-learning utilizing the batch sample D under the Markov Decision Process M_D, it can be mathematically inferred that Q_D^{π} converges towards the optimal value function Q^{π}*

$$\varepsilon_{MDP}(s,a) = Q^{\pi}(s,a) - Q_D^{\pi}(s,a). \tag{6}$$

For any target policy π, $\varepsilon_{MDP}(s,a)$ involves the propagation of extrapolation error to the next state

$$\begin{aligned}\varepsilon_{MDP}(s,a) \propto \sum_{s'} \big(p_M(s' \mid s,a) - p_D(s' \mid s,a)\big) \\ + p_M(s' \mid s,a)\gamma \sum_{a'} \pi(a' \mid s')\varepsilon_{MDP}(s',a').\end{aligned} \tag{7}$$

Proof 1. *We can prove this using the Bellman equation. We will define extrapolation $\varepsilon_{MDP}(s,a)$ for each state and actions as*

$$\begin{aligned}\varepsilon_{MDP}(s,a) &= \sum_{s'}(p_M(s'|s,a) - p_D(s'|s,a))r(s,a,s') - p_D(s'|s,a)\gamma\sum_{a'}\pi(a'|s')Q_D^{\pi}(s',a') \\ &+ p_M(s'|s,a)\gamma\sum_{a'}\pi(a'|s')(Q_D^{\pi}(s',a') + \varepsilon_{MDP}(s',a')) \\ &= \sum_{s'}(p_M(s'|s,a) - p_D(s'|s,a))r(s,a,s') + p_M(s'|s,a)\gamma\sum_{a'}\pi(a'|s')(Q_D^{\pi}(s',a') + \varepsilon_{MDP}(s',a')) \\ &+ p_M(s'|s,a)\gamma\sum_{a'}\pi(a'|s')(\varepsilon_{MDP}(s',a') - \varepsilon_{MDP}(s',a')) - p_D(s'|s,a)\gamma\sum_{a'}\pi(a'|s')Q_D^{\pi}(s',a') \\ &= \sum_{s'}(p_M(s'|s,a) - p_D(s'|s,a))\left(r(s,a,s') + \gamma\sum_{a'}\pi(a'|s')Q_D^{\pi}(s',a')\right) \\ &+ p_M(s'|s,a)\gamma\sum_{a'}\pi(a'|s')\varepsilon_{MDP}(s',a')\end{aligned}$$

Therefore, we observe that extrapolation, $\varepsilon_{MDP}(s,a)$, propagates from the next state, and the greater the discrepancy between the transition dynamics in the actual environment and the dataset, the more error accumulates. □

According to theorem 1, we understand that extrapolation is propagated through Bellman updates, and we can see that the error intensifies depending on the difference in transition probabilities between the batch data we want to learn from and the actual MDP M. These extrapolation errors are closely

related to the out-of-distribution (OOD) problem. The OOD problem implies that the agent's predictions might be inaccurate for new situations or data not present in the training dataset, and such errors can accumulate due to the Bellman operator. Based on these theoretical foundations, Batch-Constrained deep Q-learning (BCQ) aims to prevent the occurrence of actions that deviate from the batch data distribution. It achieves this by using a generative model to predict the distribution of the batch data, and then it extends the action space by introducing a slight perturbation.

Conservative Q-Learning (CQL) [1] is one of the methods that apply the traditional Q-learning technique mentioned in Equation 5 to constrain the target policy towards the behavior policy. Q-learning employs the Bellman equation to learn the action value function, after which the policy is updated to maximize this value function. However, it is crucial to consider the prediction error of the value function. In traditional Q-learning, the error converges to zero as the number of data points increases. But in Offline RL, using limited data means the value function's error is propagated and accumulates via the Bellman equation. Since the policy is updated to maximize rewards in the next state, extrapolation error in Offline RL can lead to severe overestimation issues. ensures safer policy evaluation by learning a value function that lower bounds the value function learned in the actual MDP, using a loss function that minimizes the predicted action value function of the target policy.

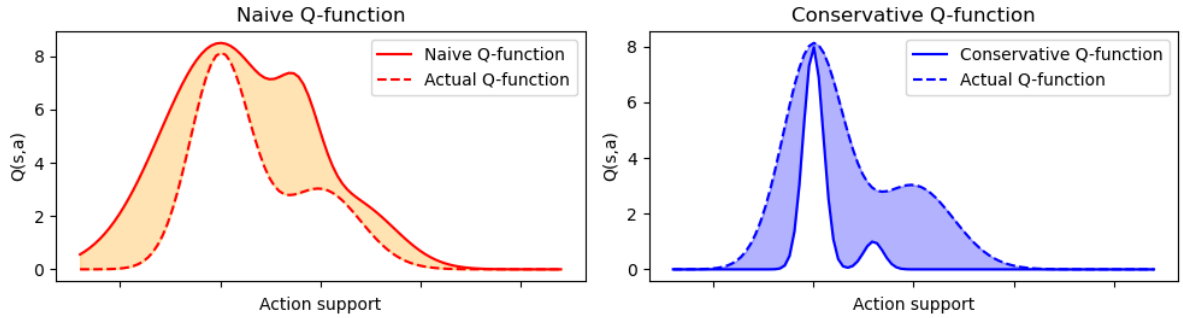


Figure 3: Comparative Illustration of Q-Function Estimations. (a) depicts a Naive Q-function estimation that tends to overestimate the action values, as shown by the area where the Naive Q-function exceeds the Actual Q-function. (b) demonstrates the Conservative Q-function approach used in Conservative Q-Learning (CQL), where the estimated Q-values are intentionally lower than the Actual Q-function for actions outside the support of the offline data to prevent overestimation.

Through Figure [1], we can understand the differences between traditional Q-function learning methods and the conservative Q-learning approach. Naive Q-learning assumes that errors decrease as the number of data points increases, with no constraints on the Q-function, leaving it vulnerable to erroneous value function predictions. This can lead to an overestimation of Q-values for actions that do not occur frequently in the offline dataset, potentially misguiding the actual policy in a detrimental direction. However, conservative Q-learning ensures that the action value function of the target policy is trained to lower bound the value function of the behavior policy, resulting in Q-value estimates for actions outside of the batch data being lower than the actual function values. This approach leads to the selection of more stable and reliable actions by the policy, reducing the risks associated with overestimation.

The intuitive idea of the Conservative Q-learning theorem is to apply a penalty to the state-action pairs outside of the batch data so that, when the policy is trained to maximize the Q-value, it is encouraged to take actions that are similar to those within the batch dataset. The objective function of CQL can be represented by the following equation:

$$\min_{Q}\max_{\mu}\alpha\left(\mathbb{E}_{s\sim D,a\sim\mu(a|s)}[Q(s,a)]-\mathbb{E}_{s,a\sim D}[Q(s,a)]\right) \\ +\frac{1}{2}\mathbb{E}_{s,a,s'\sim D}\left[\left(Q(s,a)-\hat{B}^{\pi_k}\hat{Q}^k(s,a)\right)^2\right]+R(\mu). \tag{8}$$

Theorem 2 (The underestimations of the actual value function [1])**.** *The policy value as determined by the Q-function in Equation 8, denoted by $\hat{V}^\pi(s)=\mathbb{E}_{\pi(a|s)}[\hat{Q}^\pi(s,a)]$, establishes a lower boundary for the genuine policy value, which is $V^\pi(s)=\mathbb{E}_{\pi(a|s)}[Q^\pi(s,a)]$. This is applicable in cases where $\mu=\pi$, as described in the following:*

$$\forall s\in D,\hat{V}^\pi(s)\leq V^\pi(s)-\alpha\left[(I-\gamma P^\pi)^{-1}\mathbb{E}_\pi\left[\frac{\pi}{\hat{\pi}_\beta}-1\right]\right](s)+\left[(I-\gamma P^\pi)^{-1}\frac{C_{r,T,\delta}R_{\max}}{(1-\gamma)\sqrt{|D|}}\right](s). \tag{9}$$

If the parameter α is greater than the product of $\frac{C_{r,T}R_{\max}}{1-\gamma}$, the maximum value among all states s in the dataset D of $\frac{1}{\sqrt{|D|}}$, and the inverse of the sum over all actions a of $\pi(a|s)$ multiplied by the difference between $\frac{\pi(a|s)}{\hat{\pi}_\beta(a|s)}$ and 1, then for every state s in the dataset D, the estimated value function $\hat{V}^\pi(s)$ is less than or equal to the true value function $V^\pi(s)$, with a probability of at least $1-\delta$. Additionally, when the estimated Bellman operator $\hat{B}^\pi$ matches the actual Bellman operator B^π, any positive value of α will guarantee that $\hat{V}^\pi(s)$ is less than or equal to $V^\pi(s)$ for every state s in the dataset D.

Proof 2. *For the proof of Theorem 2, it is necessary to demonstrate that the expectation of the penalty term $D_{CQL}(s):=\sum_a\pi(a|s)\left[\frac{\pi(a|s)}{\pi_\beta(a|s)}-1\right]$ is always greater than or equal to 0, with equality holding when $\pi=\pi_\beta$. To illustrate this, we provide the following derivation:*

$$\begin{aligned}D_{CQL}(s)&:=\sum_a\pi(a|s)\left[\frac{\pi(a|s)}{\pi_\beta(a|s)}-1\right]\\&=\sum_a(\pi(a|s)-\pi_\beta(a|s)+\pi_\beta(a|s))\left[\frac{\pi(a|s)}{\pi_\beta(a|s)}-1\right]\\&=\sum_a\frac{\left(\pi(a|s)-\pi_\beta(a|s)\right)^2}{\pi_\beta(a|s)}+0\end{aligned} \tag{10}$$

As π and π_β are both policy distributions, their sums over all actions in the space sum up to one. This fact allows for a more streamlined representation, as shown in Equation 10. In this equation, the term $D_{CQL}(s)$ is consistently positive, as both its numerator and denominator are inherently positive, thus confirming the inequality $D_{CQL}(s)\geq 0$. It's also pertinent to note that the scenario where $D_{CQL}(s)=0$ happens exclusively when $\pi(a|s)$ equals $\pi_\beta(a|s)$, indicating a regular undervaluation in value iteration, as seen in the expression $V^{k+1}(s)\leq B^\pi V^k(s)$. Further elaborating, theorem 2 suggests that although $\hat{Q}_\pi$ may not consistently represent a precise lower boundary for each data point, the equation $E_\pi(a|s)[\hat{Q}_\pi(s,a)]\leq V_\pi(s)$ remains valid in situations where the implemented policy is in alignment with $\pi(a|s)$. □

Through the policy regularization term $R(\mu)$ in Equation 8, we can create various variations of conservative Q-learning for our desired policy. In our pursuit to optimize the objective function 8, we implement a KL-divergence regularizer on the policy μ, aligning it towards our desired policy distribution ρ. This is mathematically formulated as:

$$\max_{\mu} \mathbb{E}_{x\sim\mu}[f(x)] + D_{KL}(\mu||\rho) \quad \text{subject to} \quad \sum_{x} \mu(x) = 1, \text{and} \mu(x) \geq 0 \text{ for all } x.$$

Solving this optimization problem, we find that the optimal version of the policy μ is expressed as $\mu^*(x) \approx \rho(x)\exp(f(x))$. As an example of the prior distribution of the policy, you can set the policy distribution before the update as the prior distribution. This approach helps maintain a more stable learning by ensuring that the updated policy does not differ significantly from the previous policy. Another example is setting ρ to be a uniform distribution. In this case, the policy is adjusted to emphasize entropy maximization, allowing for maximum exploration within the constrained range. We used conservative Q-learning as the baseline when ρ is a uniform distribution.

Value constraint methods in offline reinforcement learning, such as Conservative Q-Learning (CQL), offer significant advantages over direct policy constraint approaches. These methods adeptly tackle distributional shift issues by learning a conservative estimate of the value function. This approach inherently underestimates the values of state-action pairs that are less represented in the training data, thereby preventing overestimation and enhancing stability and robustness against errors. Furthermore, value constraint methods do not directly restrict the policy. Instead, they focus on the estimated values, allowing for greater flexibility in policy optimization and broader applicability across various environments and tasks. Crucially, these methods are supported by theoretical guarantees. In offline settings, where interaction with the environment is absent, such guarantees are vital, offering a degree of predictability in the algorithm's performance. Despite these advantages, there is still a significant risk of underestimation for policy actions, which can lead to a degradation in algorithm performance. In this paper, we aim to propose a more efficient penalty term by analyzing offline data, addressing this issue and enhancing the overall effectiveness of the approach.

2.2.2 Uncertainty Estimation

When learning from an offline dataset, if the data does not fully specify information about the actual environment, uncertainty can arise as implicit partial observability within the original fully observable learning problem in a true MDP. This uncertainty challenge can result in difficulties when estimating transition probabilities and reward functions, leading to an inflation of learned estimations. Established approaches for managing uncertainty in such scenarios employ value functions to evaluate the associated risk of forecasting both transition probabilities and reward functions for a given policy. Subsequently, they adopt a cautious learning approach grounded in this evaluation when handling models characterized by a significant degree of incompleteness.

Ensemble Distributional Actor-Critic(EDAC) [14] is a form of uncertainty-based offline reinforcement learning method. It leverages the clipped Q-learning algorithm, which was originally proposed in online reinforcement learning, to facilitate conservative learning in offline settings. This is

achieved through the ensemble effect of value functions. Clipped Q-learning is a technique designed to maximize the minimum Q-value among several Q-value functions. In EDAC, it can be explained as a method that penalizes deviations based on the standard deviation between Q-value functions, where the number of these functions is denoted as N. When considering $Q(s,a)$, which follows a Gaussian distribution with mean $m(s,a)$ and standard deviation $\sigma(s,a)$, clipped Q-value can be seen as a way to discourage deviations from the ensemble mean of the Q-values [14] :

$$\mathbb{E}\left[min_{j=1,\cdots,N}Q_j(s,a)\right] \approx m(s,a) - \phi^{-1}\left(\frac{N-\frac{\pi}{8}}{N-\frac{\pi}{4}+1}\right)\sigma(s,a). \tag{11}$$

Therefore, in EDAC, the ensemble effect of a large number of Q prediction neural networks can be utilized to minimize the uncertainty in Q predictions. It is argued that by increasing the gradient diversity of the Q function in this context, the effective utilization of clipped Q-learning can be achieved, reducing the required number of Q networks for uncertainty prediction

2.2.3 Supervised-learning Based

Supervised learning approaches aim to make accurate predictions by utilizing the labels of data. In Offline RL, the reward information given for actions is used as labels, and the goal is to accurately predict the Q-values using this information. Therefore, supervised learning-based offline RL primarily addresses the issue of distribution shift, which often leads to inaccurate Q-value predictions, by updating using only the behavior policy instead of using target values generated from the target policy during the policy evaluation process [15].

The **One-step algorithm** [16] suggests that high variance in Q evaluations is a result of distribution shifts and the repetitive use of errors. To address this, the algorithm advocates for a novel approach: pre-training the behavior policy and then using it to update the value function. This strategy is designed to reduce the instability that occurs during the simultaneous updates of the policy and value function. The algorithm employs supervised learning for the value function using pre-collected data, where return values are calculated beforehand and used as labels. By leveraging the returns from the dataset for supervised learning, it can avoid out-of-distribution problems typically associated with the target policy, enabling more accurate predictions for the batch dataset without necessitating penalties or uncertainty predictions. Subsequently, policy learning is conducted to maximize the learned value function. This method differs from previous approaches as it leads to reduced variation in the value function, thereby minimizing prediction uncertainty. However, the effectiveness of this method is highly dependent on the congruence between the behavior and target policies, and its performance is significantly influenced by the value of the batch data, posing challenges in addressing the discrepancies between the actual environment and the dataset's Markov Decision Process.

III Penalty Relaxation

Offline RL is vulnerable to out-of-distribution problems due to the occurrence of extrapolation errors, as it can only utilize limited data for learning. Therefore, it is very important to train the target policy based on the behavior policy. The previously proposed Conservative Q-learning (CQL) [1] is a representative value-based offline RL method, which suggests a conservative Q-learning method utilizing a penalty term.

3.1 Peanlty Variants

As shown in Equation 8, the original CQL defines the penalty as $\left(\frac{\pi}{\pi_\beta} - 1\right)$. Therefore, in the proof of Theorem 2 for conservative Q-learning, we can observe that the following conditions are necessary:

1. $\sum_a \pi(a|s) \left[\frac{\mu(a|s)}{\pi_\beta(a|s)} - 1\right] \geq 0$
2. $\sum_a \pi_\beta(a|s) \left[\frac{\mu(a|s)}{\pi_\beta(a|s)} - 1\right] = 0$

However, this approach applies a uniform penalty to all actions within the action space, excluding the behavior policy. This uniformity in penalty application can potentially lead to underestimation of performance. This issue arises because it doesn't take into account the degree of deviation between the target and behavior policies or the specific attributes of the behavior policy.

To address the problem of underestimation resulting from the penalty, we introduce an additional penalty term. This term can vary depending on the characteristics of the dataset and the attributes of the target policy. As stated in Theorem 2, by integrating a penalty loss function into the standard Q-learning framework, we can develop a Q-function that forms a lower boundary for the true Q-function. This formulation is essentially made up of a linear mix of convex functions. Hence, the theorem still applies even if this penalty is scaled by a constant factor $k(s, \pi_\beta(a|s))$ ranging between 0 and 1, ensuring that the resultant Q-values maintain their status as lower bounds. Hence, we propose a penalty function $k(s, \pi_\beta(a|s))$ that is independent of the target policy and can adapt to the dataset's characteristics. It's important to emphasize the role of $k(s, \pi_\beta(a|s))$. By simply adding a constant value independent of π, we gain flexibility in designing the penalty, allowing us to adjust the constant C flexibly depending on the dataset's quality or unique characteristics. In this context, we have the freedom to design k as a continuous decreasing function with values between 0 and 1, and when using such a decreasing function, we can flexibly adjust the penalty of the target policy based on the deviation from the behavior policy.

Therefore, based on Theorem 2, we propose a scaling factor denoted as $\left(1 - \frac{\pi_\beta}{C}\right)_+^\alpha$, where the "+" operator serves as a clipping operator to keep the scaling factor within the range of 0 to 1. Here, α is a temperature parameter that can control the influence of the scaling factor. It's essential to emphasize the significance of the constant value C. By utilizing C, we can adjust the penalty based on the characteristics of the behavior policy π_β, and we can adaptively adjust the constant C based on the dataset's quality or unique characteristics. This adaptability allows for a more detailed Q evaluation of the target policy. Here, we can utilize the simplest random density ρ without any parameter adjustments, and we can

prove the stability of the proposed penalty. As a result, this enhances the diversity of actions taken in specific states when using a random behavior policy. An interesting point is that this adjustment ensures that even in situations with low penalties, the policy closely aligns with the behavior policy, minimizing significant deviations. These penalties provide a tighter bound on the actual Q value by imposing lower penalties on the target policy compared to the original CQL. This can be demonstrated using methods similar to Yu. et al [17].

Lemma 1. *Assuming previous notation, let*

$$\Delta^{\pi}_{CQL} := \mathbb{E}_{s\sim D, a\sim\pi}\left[\hat{Q}^{\pi}_{CQL}(s,a)\right]$$

and

$$\Delta^{\pi}_{RCQL} := \mathbb{E}_{s\sim D, a\sim\pi}\left[\hat{Q}^{\pi}_{RCQL}(s,a)\right]$$

represent the mean values over the dataset obtained from the Q-functions learned by CQL and RCQL, respectively. Then, $\Delta^{\pi}_{RCQL} \geq \Delta^{\pi}_{CQL}$ if:

$$\mathbb{E}_{s\sim D, a\sim\pi(a|s)}\left[\left(1-\frac{\pi_\beta(a|s)}{\rho(a|s)}\right)^{\alpha}_{+}\right] \leq 1.$$

Proof 3. *In this proof, we will show how conservative penalties have been mitigated in terms of the estimated value. Let's define the penalty for the policy π of CQL and the RCQL we proposed as follows:*

$$\Delta^{\pi}_{CQL} := \mathbb{E}_{s\sim D, a\sim\pi}\left[\hat{Q}^{\pi}_{CQL}(s,a)\right]$$
$$\Delta^{\pi}_{RCQL} := \mathbb{E}_{s\sim D, a\sim\pi}\left[\hat{Q}^{\pi}_{RCQL}(s,a)\right].$$

From Kumar et al. [1], we obtain that

$$\hat{Q}^{\pi}_{CQL}(s,a) := Q^{\pi}(s,a) - \eta\frac{\pi(a|s)-\pi_\beta(a|s)}{\pi_\beta(a|s)}$$

To derive the condition when $\Delta^{\pi}_{RCQL} \geq \Delta^{\pi}_{CQL}$, we note the following equations:

$$\begin{aligned}
&\Delta^{\pi}_{RCQL} \geq \Delta^{\pi}_{CQL} \\
&\Rightarrow \sum_{s,a}\pi(a|s)\hat{Q}^{\pi}_{RCQL}(s,a) \geq \sum_{s,a}\pi(a|s)\hat{Q}^{\pi}_{CQL}(s,a) \\
&\Rightarrow \sum_{s,a}\pi(a|s)\left(\frac{\pi(s,a)-\pi_\beta(a|s)}{\pi_\beta}\right)\left(1-\frac{\pi_\beta}{\rho}\right)^{\alpha}_{+} \leq \sum_{s,a}\pi(a|s)\left(\frac{\pi(a|s)-\pi_\beta(a|s)}{\pi_\beta(a|s)}\right).
\end{aligned} \tag{12}$$

Therefore, we can see that if the additional penalty relaxation term is less than 1, the inequality in Equation 12 is satisfied. We must note the role of the clipping operator in this context. Finally, if we rearrange Equation 12 into expectations, we can obtain the result of Lemma 1. □

Lemma 1 establishes a condition where the estimated values of the Q-function in Regularized Conservative Q-Learning (RCQL) are equal to or greater than those in Conservative Q-Learning (CQL).

This condition is articulated through the equation $\mathbb{E}_{s\sim D,a\sim\pi(a|s)}\left[\left(1-\frac{\pi_\beta(a|s)}{\rho(a|s)}\right)_+^\alpha\right]\leq 1$, signifying the necessary magnitude of the additional regularization term in RCQL. Furthermore, the use of the proposed diminishing penalty function allows for adjusting the degree of difference between the target and behavior policies. It indicates that the closer the target policy is to the behavior policy, the lower the penalty imposed by RCQL compared to CQL. Hence, this lemma demonstrates that RCQL offers a more flexible approach than CQL, enabling more precise adjustments of the Q-function's estimated values. This difference is especially significant in enhancing the performance of reinforcement learning algorithms in uncertain environments or situations with limited data.

However, we must know that the introduction of an additional penalty term may not fulfill the conditions necessary for conservative Q-learning. To resolve this issue, we have introduced an additional compensate loss function for the behavior policy in our objective function, specifically to avoid imposing penalties on the behavior policy. This reward function not only facilitates conservative Q-learning but also addresses the errors associated with the logsumexp function used in the original CQL. Consequently, the revised objective function is as follows:

$$\begin{aligned}\min_Q\ &\mathbb{E}_{s\sim D,a\sim\pi(a|s)}\left[\left(1-\frac{\pi_\beta}{\rho}\right)_+^\alpha Q(s,a)\right]-\mathbb{E}_{s,a\sim D}\left[\left(1-\frac{\pi_\beta}{\rho}\right)_+^\alpha Q(s,a)\right]\\&-\left(\mathbb{E}_{s\sim D,a\sim\pi(a|s)}\left[\left(1-\frac{\pi_\beta}{\rho}\right)_+^\alpha\right]-\mathbb{E}_{s,a\sim D}\left[\left(1-\frac{\pi_\beta}{\rho}\right)_+^\alpha\right]\right)\mathbb{E}_{s,a\sim D}[Q(s,a)]\\&+\frac{1}{2}\mathbb{E}_{s,a,s'\sim D}\left[\left(Q(s,a)-\hat{B}^{\pi_k}\hat{Q}^k(s,a)\right)^2\right].\end{aligned}\tag{13}$$

The inclusion of an additional reward term effectively compensates for the penalties applied to the batch dataset, thereby enhancing the stability of our approach's convergence towards the batch dataset. Furthermore, this strategic addition facilitates the validation of the conditions necessary for conservative q-learning, solidifying the theoretical foundation of our methodology.

Lemma 2 (Stability Under Constant Scaling of the Penalty Term)**.** *Applying any positive scaling factor* $\left(1-\frac{\pi_\beta(a|s)}{\rho(a|s)}\right)_+^\alpha$ *to the penalty term continues to ensure that the policy's value remains a lower bound according to the Q-function, as compared to the true value obtained through exact policy evaluation:*

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

Thus, if $\eta>\frac{C_{r,T}R_{\max}}{1-\gamma}\cdot\max_{s\in D}\frac{1}{|\sqrt{D}|}\cdot\left[\sum_a\pi(a|s)\left(\frac{\pi(a|s)}{\hat{\pi}_\beta(a|s)}-1\right)\left(1-\frac{\pi_\beta(a|s)}{\rho(a|s)}\right)_+^\alpha\right]^{-1}$*, then it is highly likely that for all states s in the set D, the estimated value* $\hat{V}^\pi(s)$ *is less than or equal to the true value* $V^\pi(s)$*. In cases where the empirical Bellman operator* $\hat{B}^\pi$ *equals the true Bellman operator* B^π*, any positive value of* η *ensures that* $\hat{V}^\pi(s)$ *remains less than or equal to* V^π*(s for all states s in D [18, 19].*

Proof 4. *In the discrete state-action space scenario, we can compute the derivative of the modified objective in Equation 13 and perform Q-function updates in the exact environment, assuming* $\hat{B}^\pi=B^\pi$

and $\pi_\beta(a|s) = \hat{\pi}_\beta(a|s)$. *With our objective function defined as Equation 13, the penalty term is expressed as:*

$$\left(\frac{\pi(a|s)}{\pi_\beta(a|s)} - 1\right)\left(1 - \frac{\pi_\beta(a|s)}{\rho(a|s)}\right)_+^\alpha - \mathbb{E}_{\pi_\beta(a|s)}\left[\left(\frac{\pi(a|s)}{\pi_\beta(a|s)} - 1\right)\left(1 - \frac{\pi_\beta(a|s)}{\rho(a|s)}\right)_+^\alpha\right]$$

Then,

$$\begin{aligned}
D^\pi_{Ours}(s) &:= \sum_a \pi(a|s)\left[\left(\frac{\pi(a|s)}{\pi_\beta(a|s)} - 1\right)\left(1 - \frac{\pi_\beta(a|s)}{\rho(a|s)}\right)_+^\alpha - \mathbb{E}_{\pi_\beta(a|s)}\left[\left(\frac{\pi(a|s)}{\pi_\beta(a|s)} - 1\right)\left(1 - \frac{\pi_\beta(a|s)}{\rho(a|s)}\right)_+^\alpha\right]\right] \\
&= \sum_a (\pi(a|s) - \pi_\beta(a|s)\left(\frac{\pi(a|s)}{\pi_\beta(a|s)} - 1\right)\left(1 - \frac{\pi_\beta(a|s)}{\rho(a|s)}\right)_+^\alpha \\
&= \sum_a \frac{(\pi(a|s) - \pi_\beta(a|s))^2}{\pi_\beta(a|s)} + \sum_a \pi_\beta(a|s)\left[\frac{\pi(a|s)}{\pi_\beta(a|s)} - 1\right] \\
&= \sum_a \frac{(\pi(a|s) - \pi_\beta(a|s))^2}{\pi_\beta(a|s)} + 0, \textit{ since } \sum_a \pi(a|s) = \sum_a \pi_\beta(a|s) = 1.
\end{aligned}$$

Furthermore, given that we apply the penalty to the behavior policy π_β, $D^{\pi_\beta}_{Ours}(s)$ *always remains zero. With this, we can calculate the fixed point of the recursion presented in Equation 13, which provides us with a lower bound on the true value function. This formulation is more stringent than the policy value expression in Equation 9 because it assigns a weight ranging from 0 to 1 to the Q-values established in CQL [1].* □

By applying the Q function learning method we've developed, it's feasible to demonstrate that safe policy improvement through Q-learning can be achieved using the approach outlined in Conservative Q-Learning (CQL) [1]. Consider $R(\pi, D)$ to symbolize the returns of policy π within the MDP D. The optimal policy $\pi^*(a|s)$ emerges by resolving:

$$\pi^*(a|s) \leftarrow \arg\max_\pi R(\pi, D) - \frac{\alpha}{1-\gamma}\mathbb{E}_{s\sim d^\pi_D}\left(\underbrace{\mathbb{E}_\pi\left[k(s,a)\left(\frac{\pi(a|s)}{\pi_\beta(a|s)} - 1\right)\right]}_{\text{referred to as } D^\pi_{RCQL}}\right),$$

where Q^π aligns with the equilibrium state of Equation 13. This approach guarantees policy improvement under the actual MDP M.

Theorem 3 (Safe Policy Improvement Guarantees)**.** *The policy distribution* $\pi^*(a|s)$ *constitutes a* ζ*-safe advancement compared to* π_β *within the true MDP M. In essence, this implies that* $R(\pi^*, M)$ *is at least* $R(\pi_\beta, M) - \zeta$*, where the value of* ζ *is determined as*

$$\begin{aligned}
\zeta = 2\left(\frac{C_{r,\delta}}{1-\gamma} + \frac{\gamma r_{max} C_{T,\delta}}{(1-\gamma)^2}\right)\mathbb{E}_{s\sim d^{\pi^*}_D(s)}\left[\sqrt{\frac{|A|}{|D(s)|}} D_{RCQL}(\pi^*, \pi_\beta)(s) + k(s,a)\right] \\
- \left(\frac{R(\pi^*, D) - R(\pi_\beta, D)}{\alpha}\right).
\end{aligned} \tag{14}$$

Proof 5. *Revisiting the definitions of $R(\pi^*, M)$ and $R(\pi_\beta, M)$, the discrepancy in returns can be bounded as*

$$R(\pi, D) - R(\pi, M) = \frac{1}{1-\gamma}\left(\sum_{s,a} d_D^\pi(s)\pi(a|s)r(s,a) - \sum_{s,a} d_M^\pi(s)\pi(a|s)r(s,a)\right)$$
$$\leq \frac{1}{1-\gamma}\left|\sum_{s,a} d_D^\pi(s)\left[\pi(a|s)(r(s,a) - \bar{r}(s,a))\right]\right| + \frac{1}{1-\gamma}\left|\sum_{s,a}(d_D^\pi(s) - d_M^\pi(s))\pi(a|s)r_M(s,a)\right|$$

By applying the triangle inequality, we can establish an upper boundary for the variation in returns, influenced by disparities in both the reward functions and transition dynamics. The initial term, which reflects the variability in the reward functions, is constrained and can be articulated in terms of $C_{r,\delta}$ and $C_{T,\delta}$, as detailed in [20].

$$\sum_a \pi(a|s)\left|r(s,a) - \bar{r}(s,a)\right| \leq \sum_a \pi(a|s)\frac{C_{r,\delta}}{\sqrt{|D(s)|}} = \frac{C_{r,\delta}}{\sqrt{|D(s)|}}\sum_a \frac{\pi(a|s)}{\sqrt{\pi_\beta(a|s)}}$$

Further, we introduce G as the reciprocal of $I - \gamma P_M^\pi$ and $\bar{G}$ as the reciprocal of $I - \gamma P_D^\pi$, with Δ denoting the difference between P_M^π and P_D^π. This approach allows us to set a limit on the overall variation in state distribution margins. Specifically, the variance in discounted state visitation distributions under policy π for the model M relative to the empirical model D, described as $d_M^\pi - d_D^\pi$, can be represented by $(1-\gamma)\gamma G\Delta G\bar{\mu}$, and similarly, by $\gamma\bar{G}\Delta d_M^\pi$ as per [21]. Consequently:

$$\begin{aligned}
\|\Delta d_D^\pi\|_1 &= \sum_{s'}\sum_{s}\left|\Delta(s'|s)d_D^\pi(s)\right| \\
&\leq \sum_{s,s'}\sum_{a}\left(P_D(s'|s,a) - P_M(s'|s,a)\right)\pi(a|s)d_D^\pi(s) \\
&\leq \sum_{s,a}\left\|P_D(\cdot|s,a) - P_M(\cdot|s,a)\right\|_1 \pi(a|s)d_D^\pi(s) \\
&\leq \sum_s d_D^\pi(s)\frac{C_{T,\delta}}{\sqrt{|D(s)|}}\sum_a \pi(a|s)\sqrt{\frac{\pi_\beta(a|s)}{\pi(a|s)}}.
\end{aligned}$$

As a result, the difference in returns can be depicted as a function of $\frac{\pi_\beta(a|s)}{\pi(a|s)}$, and this boundary can be represented using our penalty term D_{RCQL}. Define $\alpha(s,a)$ as $\frac{\pi(a|s)}{\sqrt{\pi_\beta(a|s)}}$. Then, $D_{RCQL}(\pi, \hat{\pi}_\beta)(s)$ is expressed as:

$$\begin{aligned}
D_{RCQL}(s) &= \sum_a \left(\frac{\pi(a|s)}{\pi_\beta(a|s)}\right)^2 - k(s,a) \\
\Rightarrow D_{RCQL}(s) + k(s,a) &= \sum_a \alpha(s,a)^2 \\
&\leq \left(\sum_a \alpha(s,a)\right)^2 \leq |A|(D_{RCQL}(s) + k(s,a)).
\end{aligned}$$

Merging these insights, we derive the upper limit on $|R(\pi, D) - R(\pi, M)|$:

$$|R(\pi, D) - R(\pi, M)| \leq \left(\frac{C_{r,\delta}}{1-\gamma} + \frac{\gamma R_{max} C_{T,\delta}}{(1-\gamma)^2}\right)\mathbb{E}_{s\sim d_D^\pi(s)}\left[\sqrt{\frac{|A|}{|D(s)|}}D_{RCQL}(\pi, \hat{\pi}_\beta)(s) + k(s,a)\right].$$

□

Therefore, we can see that if the dataset D is sufficiently large, we can utilize the proposed penalty to update the policy towards the true optimal policy.

3.2 Dataset Optimality

Offline Reinforcement Learning (RL) relies on a fixed dataset for learning, facing significant challenges due to data dependency. A major issue is the dataset's bias towards specific policies or environmental states, which limits the model's learning experiences. This limitation can lead to poor responses in unexpected real-world situations, as the model struggles to generalize effectively across different scenarios. Additionally, this bias may cause the model to learn a behavior distribution that diverges from real-world encounters, leading to unexpected decisions and performance degradation. However, the most critical problem in Offline RL is the risk of overfitting to the limited dataset. In such cases, the model performs well on training data but fails to generalize to real-world environments. This overfitting is a major obstacle, preventing the model from effectively adapting to new data or scenarios. Addressing these issues is essential for the advancement and practical application of Offline RL.

To mitigate these challenges, we adopt a strategy that goes beyond simply replicating the behavior policy as is. Instead, we aim to learn and utilize the behavior policy in proportion to the quality of the data. The previously proposed Implicit Q-Learning (IQL) [22] attempts to partially address these issues by using expectile regression to approximate the maximum Q value in a SARSA style target value. While this method enhances overall performance by leveraging the action with the highest value in the batch data for Q updates, it still relies solely on the dataset to learn the maximum batch action, not fully resolving the dependency on the behavior policy and resulting in limited performance. To overcome this, we propose a strategy to predict a more optimized behavior policy by evaluating the value in the batch data according to the policy's learning tendencies. This approach fosters a more effective and adaptable methodology in Offline RL.

In Soft Q learning, the Maximum Entropy Principle forms the basis for the assumption that expert policies are designed to maximize entropy [23–25]. Building on this premise, we utilize data clustering techniques at the outset to categorize states within batch data that exhibit similarities. Following this, we archive the Q values associated with this batch data. The next step involves leveraging exponential target Q values for pairs of states within each cluster. This process is crucial for updating the behavior policy, ensuring it aligns with a Boltzmann distribution. The revised behavior policy, which we denote as $\pi_{\beta'}$, is encapsulated in the subsequent equation:

$$\pi_{\beta'} \propto \frac{exp(Q(s,a))}{\mathbb{E}_{s,a\in C}[exp(Q(s,a))]}$$

Note that the updated behavior policy is dependent on the Q values updated towards the target policy. This dependency allows for the acquisition of a more optimal behavior policy compared to previously proposed methods. We will utilize the obtained updated behavior policy to perform Bellman updates and conservative Q-learning. Therefore, in the previously proposed objective equation, Equation 13, we

will use the updated behavior policy $\pi_{\beta'}$ instead of the original behavior policy π_β. Our final objective function is as follows:

$$\begin{aligned}\min_Q \mathbb{E}_{s,a\sim d_D^{\pi_{\beta'}},\pi_{\beta'}(a|s)}\left[\left(1-\frac{\pi_{\beta'}}{\rho}\right)_+^{\alpha} Q(s,a)\right] - \mathbb{E}_{s,a\sim d_D^{\pi_{\beta'}},\pi_{\beta'}(a|s)}\left[\left(1-\frac{\pi_{\beta'}}{\rho}\right)_+^{\alpha} Q(s,a)\right] \\ -\left(\mathbb{E}_{s\sim D,a\sim\pi(a|s)}\left[\left(1-\frac{\pi_{\beta'}}{\rho}\right)_+^{\alpha}\right] - \mathbb{E}_{s,a\sim d_D^{\pi_{\beta'}},\pi_{\beta'}(a|s)}\left[\left(1-\frac{\pi_{\beta'}}{\rho}\right)_+^{\alpha}\right]\right)\mathbb{E}_{s,a\sim d_D^{\pi_{\beta'}},\pi_{\beta'}(a|s)}[Q(s,a)] \\ +\frac{1}{2}\mathbb{E}_{s,as'\sim d_D^{\pi_{\beta'}}}\left[\left(Q(s,a)-\hat{B}^{\pi_k}\hat{Q}^k(s,a)\right)^2\right].\end{aligned} \tag{15}$$

Here, $d_D^{\pi_{\beta'}}$ signifies the discounted state visitation distribution for a policy $\pi_{\beta'}$, defined using $d_M^{\pi_{\beta'}}(s) := (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi_{\beta'})$, where $P(s_t = s|\pi_{\beta'})$ represents the likelihood of arriving at state s at time t when deploying $\pi_{\beta'}$ in the MDP M.

3.3 Relaxed CQL

Algorithm 1 RCQL: Relaxed Conservative Q-learning

Require: Offline dataset D, pre-trained behavior policy π_{β_ψ}, cluster information $\mathscr{C}$, policy π_ϕ, and critic Q_θ

1: Optionally initialize the critic Q_θ and policy π_ϕ with the learned behavior policy π_{β_ψ} and Q^{π_β}.
2: Initialize and store Q values with Q^{π_β} for all samples in replay buffer.
3: **for** $i = 1,2,3,\cdots,$ **do**
4: Calculate the weight $w = \frac{exp(Q(s,a))}{\mathbb{E}_{a\sim C}[exp(Q(s,a))]}$ by comparing the stored Q values between each batch data and its cluster set $\mathscr{C}$.
5: Update the policy π_ϕ by executing gradient adjustments on ϕ, employing an entropy regularization method similar to that used in SAC:
$\phi_t := \phi_{t-1} + \eta_\pi \nabla_\phi \mathbb{E}_{s\sim D,a\sim\pi_\phi(\cdot|s)}[Q_\theta(s,a) - log\pi_\phi(a|s)]$
6: Enhance the critic Q_θ by taking gradient steps on θ, in line with Equation 17:
$\theta_t := \theta_{t-1} - \eta_Q \nabla_\theta RCQL(\theta)$
7: Advance the behavior policy π_{β_ψ} through gradient updates on ψ, utilizing weighted behavior cloning techniques:
$\psi_t := \psi_{t-1} + \eta_{\pi_\beta}\nabla_\psi \mathbb{E}_{s,a\sim D}[-w\cdot log\pi_{\beta_\psi}(a|s)]$
8: **if** i % update frequency == 0 **then**
9: Update the behavior policy π_{β_ψ}.
10: **end if**
11: **end for**

Existing offline reinforcement learning algorithms often rely on complex methodologies that augment traditional Q-learning with elaborate strategies, including ensembles of multiple Q-networks or model-based predictions for action uncertainty in subsequent states. Such approaches, while sophisticated, tend to suffer from instability, particularly as the volume of data scales up or when model precision declines. In light of these challenges, our objective is to propose a method that is both more

pragmatic and universally applicable. Our goal is to accomplish this through the use of a limited number of Q-networks, specifically just two as demonstrated in our experiments, while integrating model-free methods. This approach is intended to streamline the learning process while still ensuring strong and consistent performance in diverse scenarios. Our proposed algorithm, termed 'Relaxed CQL' and outlined in Algorithm 1, builds upon conservative Q-learning while integrating data characteristics for enhanced data efficiency compared to earlier models. This approach cycles between conservative policy evaluation and policy enhancement, following an Actor-Critic framework.

Policy Improvement with Maximum Entropy

Our algorithm represents an innovative variation of the Soft Actor-Critic (SAC) [12] framework, faithfully incorporating the maximization of entropy in the policy update process as originally advocated by SAC. This strategic adoption ensures a robust and exploratory policy by balancing exploitation with the necessary exploration. With a conservative critic $\hat{Q}^{\pi}$, we improve the policy as:

$$\pi_{\phi}^{k+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{s\sim D, a\sim\pi^{k}(\cdot|s)} \left[\hat{Q}_{\theta}^{k}(s,a) - log\pi_{\phi}^{k}(a|s) \right] \tag{16}$$

We can predict the argmax by utilizing gradient descent through a neural network. Moreover, as a modified form of the CQL algorithm, the trained policy can provide Improvement Guarantees with only a modification of the penalty as proven in Theorem 3.6 of Conservative Q-Learning (CQL) [1].

Conservative Policy Evaluation

In this phase, we address the need to update the improved behavior policy, which necessitates a careful consideration of transition probabilities. However, our approach involves conducting offline reinforcement learning without a model, presenting a unique challenge that requires an innovative approximation method. To overcome this, we have adeptly employed the technique of importance sampling. This strategic move allows us to perform conservative Q learning in a model-free environment. Specifically, we update the batch sample during the conservative Q learning process and subsequently amplify its impact by applying the importance sampling weight. This methodological innovation enables us to effectively conduct conservative Q learning without relying on a model. The detailed recursion process for Q learning is outlined as follows:

$$\begin{aligned} \min_{Q} \; & w \cdot \left(\mathbb{E}_{s\sim D, a\sim\pi(a|s)} \left[\left(1 - \frac{\pi_{\beta'}}{\rho} \right)_{+}^{\alpha} Q(s,a) \right] - \mathbb{E}_{s,a\sim D} \left[\left(1 - \frac{\pi_{\beta'}}{\rho} \right)_{+}^{\alpha} Q(s,a) \right] \right) \\ & - w \cdot \left(\mathbb{E}_{s\sim D, a\sim\pi(a|s)} \left[\left(1 - \frac{\pi_{\beta'}}{\rho} \right)_{+}^{\alpha} \right] - \mathbb{E}_{s,a\sim D} \left[\left(1 - \frac{\pi_{\beta'}}{\rho} \right)_{+}^{\alpha} \right] \right) \mathbb{E}_{s,a\sim D}[Q(s,a)] \\ & + \frac{1}{2} \mathbb{E}_{s,as'\sim D} \left[w \cdot \left(Q(s,a) - \hat{B}^{\pi_k} \hat{Q}^{k}(s,a) \right)^2 \right]. \end{aligned} \tag{17}$$

w is calculated exponentially by comparing the stored Q values between each batch data and its cluster set C, using the temperature parameter τ to adjust the magnitude of the Q values.

$$w = \frac{exp(Q(s,a)/\tau)}{\mathbb{E}_{s,a\in C}[exp(s,a)/\tau]}$$

Consequently, our approach innovatively employs clustering techniques in offline settings, offering an effective solution to alleviate the issues associated with suboptimal batch data. By strategically assigning weights to the batch data, we selectively train on the most valuable batch datasets, rather than indiscriminately using all available batches. This targeted method of training, which prioritizes high-value data, is poised to yield superior generalization capabilities when compared to traditional offline reinforcement learning methodologies.

Behavior Policy Improvement Using a Target Value

In our latest algorithm, we've introduced an innovative way of updating the behavior policy, which is a critical component for guiding how actions are chosen. An important aspect of this process is the weighted behavior cloning. This approach is grounded in the principle of weighted maximum likelihood estimation. Essentially, we adjust the behavior policy using specific weights, and this is expressed in the following formula:

$$\pi_{\beta_\psi}^{k+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{s,a\sim D}\left[-w \cdot log\pi_{\beta_\psi}^{k}(a|s)\right]. \tag{18}$$

By using batch data, this method avoids extrapolation concerns and significantly reduces interpolation errors, commonly arising when estimating values between data points, through learning based on the value of existing batch data alone. Certain challenges accompany this approach, notably when applying importance sampling to the behavior policy, which may introduce unpredictability into the learning algorithm. To enhance stability, we have adjusted the update interval so that the behavior policy is updated every 1000 steps. This approach aims to optimize the learning trajectory for maximum smoothness and effectiveness. Additionally, our algorithm includes a mechanism for systematically storing target Q values in the replay buffer in batches, thereby improving both the control and efficiency of the learning process.

Practical Implementation Details

In our strategy for clustering the batch dataset, we considered a variety of clustering methods and decided to use the Nearest Neighborhood (NN) method for its practicality. The NN method is particularly efficient as it utilizes distance information from the closest n data points, providing a more computationally efficient solution compared to other clustering techniques. Additionally, we applied a nuanced secondary filtering process based on the average distance within each cluster. This careful approach ensures that each cluster contains a maximum of 50 data points, achieving a balance between clustering accuracy and data efficiency.

IV Experiment

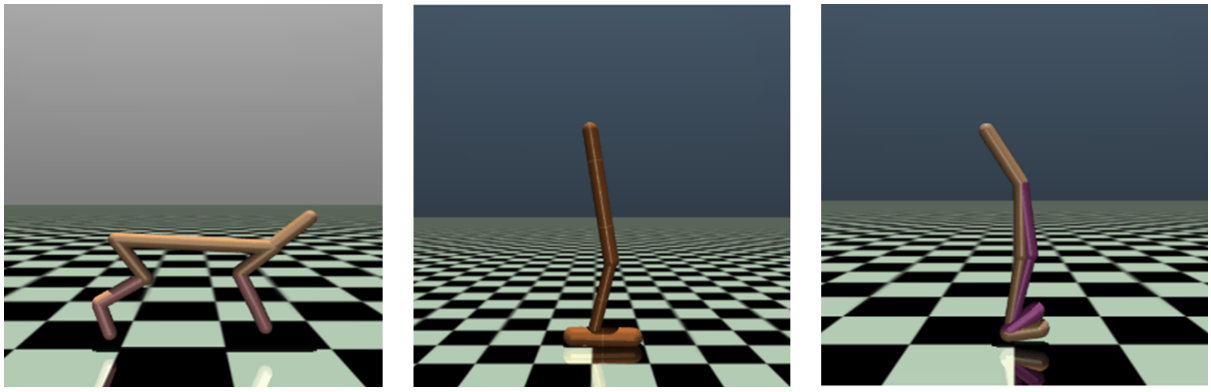


Figure 4: Mujoco locomotion tasks involve three distinct two-dimensional robotic figures, each with specific body parts and joint connections [2]. (a) The HalfCheetah is a 2D robot with 9 body parts and 8 joint connections. (b) The hopper is a two-dimensional, single-legged figure consisting of four main body parts. (c) The walker is a two-dimensional, double-legged figure with seven main body parts. In Mujoco locomotion tasks, the objective is to move forward (to the right) by applying torques to the joints that connect the body parts.

In this study, we utilize the continuous control datasets from the Gym-MuJoCo environments within the D4RL benchmark [2]. D4RL provides a collection of pre-collected interaction data from various environments and standardized evaluation metrics to objectively assess algorithm performance. The dataset includes data collected under a wide range of policy complexities. These are categorized as "-random," "-medium," and "-expert," each label indicating the sophistication level of the policy used during data collection. For the "random" datasets, the approach is straightforward. They are generated by deploying a randomly initialized policy across these environments, capturing the outcomes of this untrained approach. The "medium" dataset is generated in a unique way. It involves initially training a policy online using the Soft Actor-Critic method [12]. However, the training is stopped early, and 1 million samples are collected from this partially trained policy. The "medium-replay" or "full-replay" dataset records all samples observed in the replay buffer during the training phase, up until the point where the policy achieves a "medium" or "expert" level of performance.

4.1 Evaluation Results

We conducted an initial evaluation of our proposed techniques using the MuJoCo with D4RL offline dataset. As shown in the experimental results presented in Table 1, our methods exhibit comparable or superior performance compared to both the baseline algorithm, Conservative Q-Learning (CQL), and the state-of-the-art method EDAC. The difference in performance is particularly notable in environments with comparatively lower data quality, such as random and medium. This suggests that the traditional CQL relied excessively on batch data for performance, and our methods effectively mitigate this issue. Notably, compared to EDAC, which utilizes multiple Q prediction functions for the effect of a Q

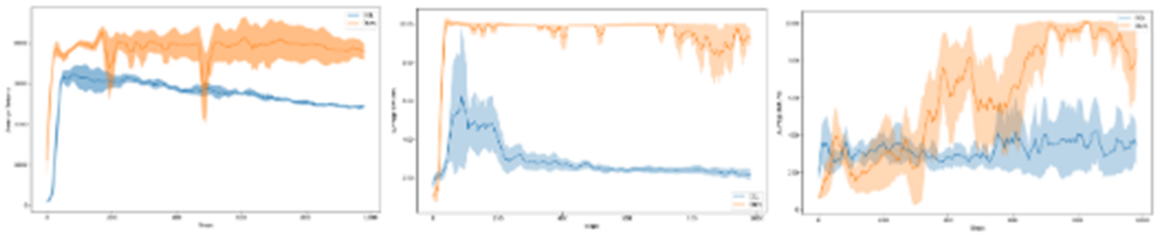
ensemble, our approach achieves comparable performance with just two Q networks. This indicates that our proposed methods enable more efficient learning without the need for uncertainty prediction by designing a more refined penalty compared to the existing CQL.

Task name	**BC**	**CQL paper**	**CQL Reprod.**	**Onestep**	**EDAC**	**Ours**
halfcheetah-random	2.2	35.4	31.3	6.9	28.4	35.6
hopper-random	3.7	10.8	5.3	6.1	25.3	31.3
walker2d-random	1.3	7.0	5.4	7.8	16.6	21.8
halfcheetah-medium	43.2	44.4	46.9	55.6	65.9	66.7
hopper-medium	54.1	86.6	61.9	86.9	101.6	100.3
walker2d-medium	70.9	74.5	79.5	83.3	92.5	90.6
halfcheetah-medium-expert	44.0	62.4	95.0	93.5	106.6	95.7
hopper-medium-expert	53.9	111.0	96.9	112.9	110.7	112.8
walker2d-medium-expert	90.1	98.7	109.1	102.1	114.7	112.2
halfcheetah-expert	91.8	104.8	97.3	-	106.8	106.6
hopper-expert	107.7	109.9	106.5	-	110.1	113.2
walker2d-expert	108.7	121.6	109.3	-	115.1	114.4
halfcheetah-medium-replay	37.6	46.2	45.3	42.4	61.3	54.4
hopper-medium-replay	16.6	48.6	86.6	71.6	101.0	99.9
walker2d-medium-replay	20.3	32.6	76.8	94.1	87.1	89.7
halfcheetah-full-replay	69.9	-	76.9	-	84.6	77.8
hopper-full-replay	19.9	-	101.9	-	105.4	105.5
walker2d-full-replay	68.8	-	94.2	-	99.8	98.2

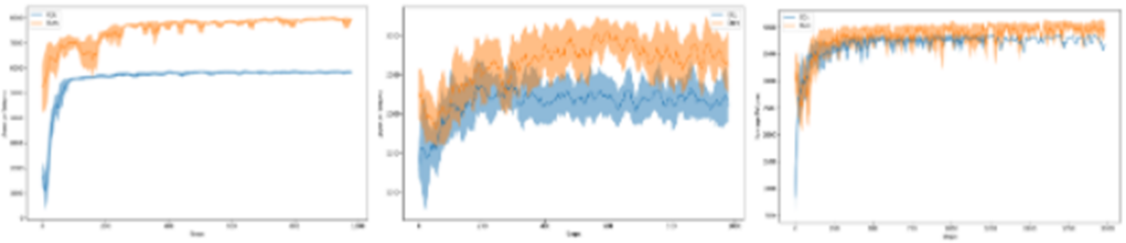
Table 1: Our evaluation of Relaxed CQL and the baseline algorithms in the D4RL gym domains, using an average return metric across three seeds, shows that CQL performs similarly to or even better than previous methods. Notably, CQL outperforms earlier methods, especially in datasets with random data distributions ('–random').

4.2 Ablation Study

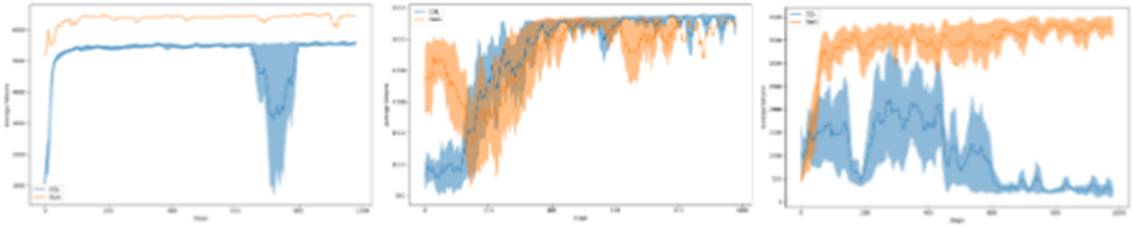
To investigate the impact of each proposed component, we conducted an ablation study on each component. Each experiment was performed by removing a single component from the setting that had the most optimal parameters according to our experimental results.



(a) Halfcheetah-random-v2, (b) Hopper-random-v2, (c) Walker2d-random-v2

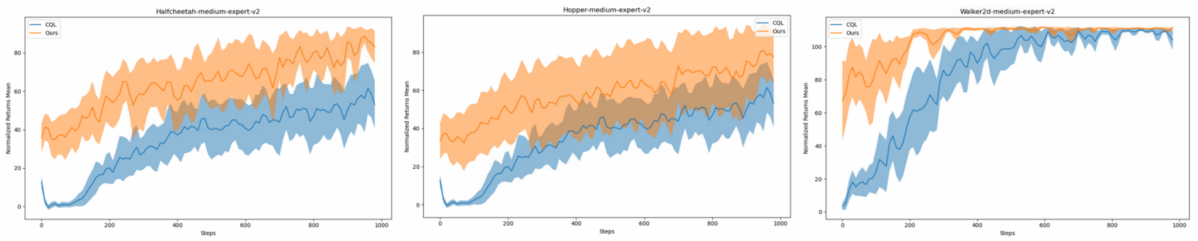


(d) Halfcheetah-medium-v2, (e) Hopper-medium-v2, (f) Walker2d-medium-v2

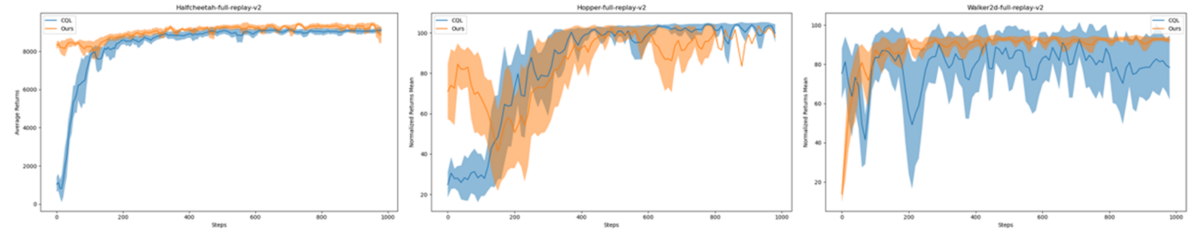


(g) Halfcheetah-medium-replay-v2, (h) Hopper-medium-replay-v2, (i) Walker2d-medium-replay-v2

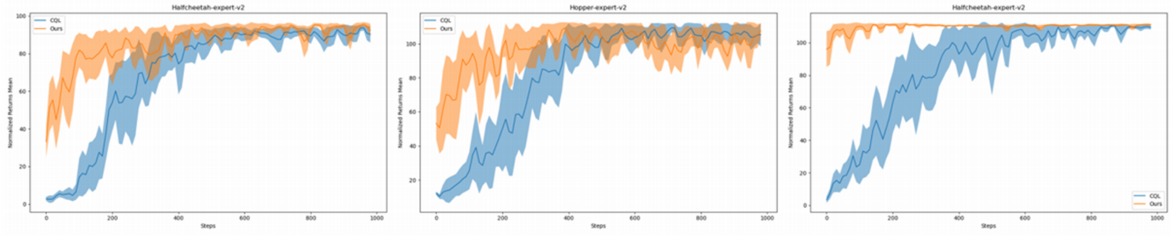
Figure 5: Performance in Random to Medium Level Environments



(j) Halfcheetah-medium-expert-v2 (k) Hopper-medium-expert-v2 (l) Walker2d-medium-expert-v2



(m) Halfcheetah-full-replay-v2, (n) Hopper-full-replay-v2, (o) Walker2d-full-replay-v2



(p) Halfcheetah-expert-v2, (q) Hopper-expert-v2, (r) Walker2d-expert-v2

Figure 6: Performance in Expert Level Environments

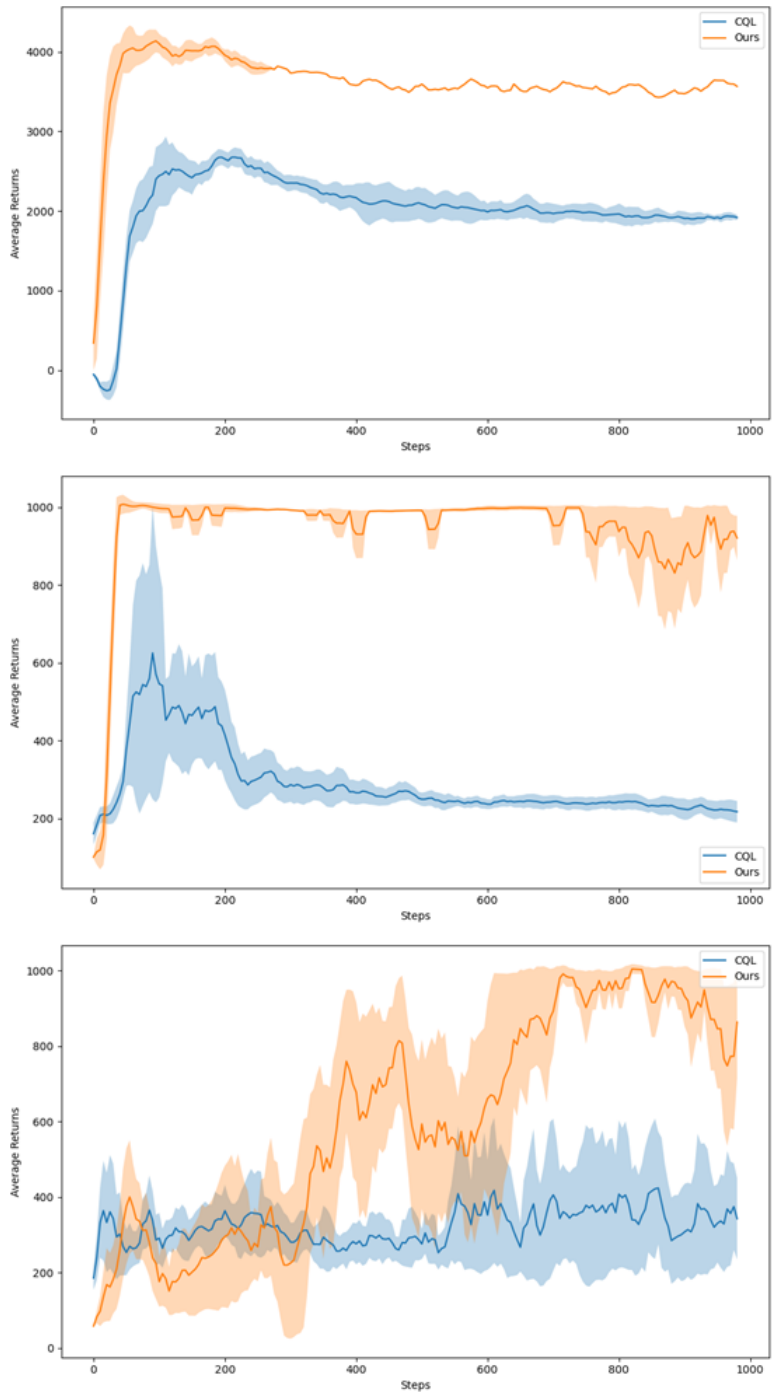


Figure 7: Ablation study to verify the effect of penalty mitigation. To assess the impact of penalty mitigation, we conducted experiments in the '-random' environment using two seeds. The key aspect of our experiment was to keep the behavior policy constant, focusing solely on adjusting the temperature parameter of penalty mitigation during the learning process.

4.2.1 Penalty

First, we conducted Q-learning using the existing behavior policy to assess the effectiveness of the penalty term we proposed. The experimental results showed that the mitigated penalty works very effectively in the '-random' environment. This effectiveness can be attributed to the excessive reliance of the conventional penalty function on batch data. Since CQL imposes the same penalty on all target policies, the target policy increasingly follows the behavior policy as learning progresses. Therefore, such a penalty can be particularly detrimental in the '-random' environment, which is composed of randomly collected data. Indeed, as shown in Figure 8, for the Hopper-random case, high return values were obtained in the initial stages of learning, but a decrease in performance was observed as learning progressed. This result indicates that the penalty mitigation technique we proposed is effectively operational for the CQL algorithm. Conversely, in situations where batch data is very limited (e.g., expert), we can impose a stronger penalty on target policies deviating from the behavior policy by increasing the constant value for penalty mitigation.

4.2.2 Improved Optimality

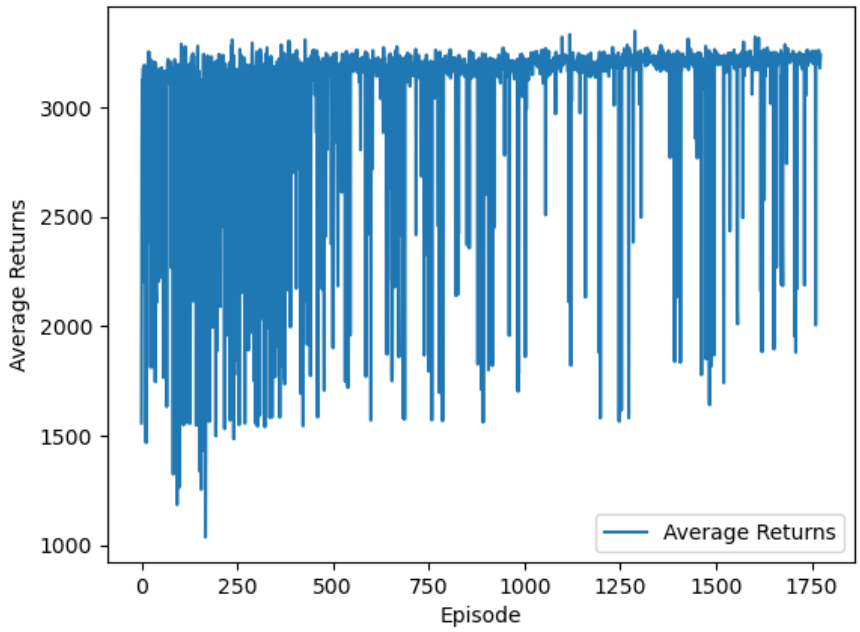


Figure 8: Ablation study on performing conservative Q-learning with an updated behavior policy. In the Hopper-medium environment, updating the behavior policy yielded very effective results. From these outcomes, we can discern that updating the behavior policy by leveraging the value of data can partially improve the sub-optimality of the data.

Another ablation study allowed us to analyze the impact of the beta prime, which involves adjusting the temperature parameter of the penalty we proposed. The experimental results showed that utilizing an improved behavior policy in the Hopper-medium environment was highly effective in enhancing performance. We can infer that this result stems from the diversity of actions inherent in the medium dataset. The medium dataset is collected using agents that were stopped early during training, hence the quality of the behavior policy varies considerably. Therefore, the behavior policy update method we proposed can reduce the reliance on excessive low-quality data.

V Conclusion

Offline reinforcement learning is an effective approach for achieving optimal performance in constrained environments. However, the challenge of incomplete value function prediction remains unresolved due to the lack of interaction with the real environment. Conventional offline reinforcement learning approaches face a trade-off between two main strategies: one strongly restricts the target policy to match the behavior policy to prevent deviations from the batch data, while the other addresses this issue by employing additional learning networks to predict uncertainty. The former strategy can be overly dependent on the dataset's quality, while the latter incurs extra computational costs.

Our proposed Relaxed CQL addresses these issues by analyzing the characteristics of the dataset and incorporating this analysis into the penalty. This approach mitigates the excessive penalty imposition inherent in traditional CQL. The penalty in Relaxed CQL is designed to gradually decrease as the actions generated by the target policy diverge from the behavior policy, proposing a more nuanced method of penalty imposition. Furthermore, by updating the behavior policy according to the dataset's value function, we can improve the dependency on incomplete data. Experimental results show that Relaxed CQL can achieve performance comparable to existing algorithms with minimal value functions. Particularly in scenarios where collected data is random or has high potential value, our proposed methods prove to be effective.

References

[1] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[2] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[3] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International conference on machine learning*. PMLR, 2019, pp. 2052–2062.

[4] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[5] Y. Wu, G. Tucker, and O. Nachum, "Behavior regularized offline reinforcement learning," *arXiv preprint arXiv:1911.11361*, 2019.

[6] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[8] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[9] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[11] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[12] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[13] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[14] G. An, S. Moon, J.-H. Kim, and H. O. Song, "Uncertainty-based offline reinforcement learning with diversified q-ensemble," *Advances in neural information processing systems*, vol. 34, pp. 7436–7447, 2021.

[15] S. Emmons, B. Eysenbach, I. Kostrikov, and S. Levine, "Rvs: What is essential for offline rl via supervised learning?" *arXiv preprint arXiv:2112.10751*, 2021.

[16] D. Brandfonbrener, W. Whitney, R. Ranganath, and J. Bruna, "Offline rl without off-policy evaluation," *Advances in neural information processing systems*, vol. 34, pp. 4933–4946, 2021.

[17] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn, "COMBO: conservative offline model-based policy optimization," *CoRR*, vol. abs/2102.08363, 2021. [Online]. Available: https://arxiv.org/abs/2102.08363

[18] I. Osband and B. Van Roy, "Why is posterior sampling better than optimism for reinforcement learning?" in *International conference on machine learning*. PMLR, 2017, pp. 2701–2710.

[19] Z. Zhang, Y. Jiang, Y. Zhou, and X. Ji, "Near-optimal regret bounds for multi-batch reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 586–24 596, 2022.

[20] R. Laroche and P. Trichelair, "Safe policy improvement with baseline bootstrapping," *CoRR*, vol. abs/1712.06924, 2017. [Online]. Available: http://arxiv.org/abs/1712.06924

[21] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International conference on machine learning*. PMLR, 2017, pp. 22–31.

[22] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *CoRR*, vol. abs/2110.06169, 2021. [Online]. Available: https://arxiv.org/abs/2110.06169

[23] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *arXiv preprint arXiv:1805.00909*, 2018.

[24] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, "Modeling interaction via the principle of maximum causal entropy," 2010.

[25] S. Reddy, A. D. Dragan, and S. Levine, "Sqil: Imitation learning via reinforcement learning with sparse rewards," *arXiv preprint arXiv:1905.11108*, 2019.

Acknowledgements

Before concluding my thesis, I would first like to express my gratitude to my advisor, Professor Seungyul Han, who has led and guided my research throughout this journey. Entering graduate school, I was deeply introduced to reinforcement learning, an area where I had much to learn and where progress seemed very slow. However, I am thankful for your assistance in writing the thesis and completing my master's defense, and for your patience until the end. Researching offline reinforcement learning was not an easy task, but working with you, Professor, I learned the mindset to overcome difficulties. I will always remember this moment and strive to continually grow and develop in society.

I also extend my thanks to my research colleague, Yonghyeon Jo. Being in a new lab meant there was much to prepare and set up, a task I believe impossible to achieve alone. I am grateful for the sense of moving towards a common goal in the lab, as we asked each other about unknowns or shared our knowledge. As a younger brother, I might have had many shortcomings, but you always generously understood and taught me, and praised me for my good work, making difficult times feel much lighter. This has allowed me to realize the importance of cooperation and to mature as a person.

During my master's program, I have undertaken many tasks, and it was due to our lab members that I could carry them out smoothly. I appreciate your understanding and following me, even when there were agreeable and disagreeable aspects of working together. It's because of you that I could discuss many things I couldn't with the professor and find opportunities for growth. I am thankful for always listening to my opinions and providing feedback, even when it might have been annoying or sensitive. Teaching juniors and distributing work among seniors has made me reflect a lot on my attitude towards work and the process of handling tasks. I will strive to grow into a person who is recognized in society based on what I have learned now.

Finally, I would like to thank the administrative staff of our lab for handling the administrative tasks. There were many inexperienced aspects in handling administrative work, yet you always treated me kindly and patiently, allowing the lab to operate until now. Being new to handling administrative tasks in the lab, I made many mistakes and had numerous questions. You preemptively pointed out potential oversights and gave special attention despite managing many labs' administrative tasks. This experience has once again made me realize that accomplishing anything requires the help of many people around us, for which I am deeply grateful.