Scientific Insights

Article

Using Reward Machines for Offline Reinforcement Learning With Non-Markovian Reward Functions

Yanze Wang

School for Engineering of Matter, Transport and Energy, Arizona State University, Arizona, AZ 85287, USA
Email: w593592469@gmail.com

Received: 21 March 2025; Accepted: 3 April 2025; Published: 27 April 2025

Abstract: We investigate the problem of offline reinforcement learning using non-Markovian reward functions, which allows for the incorporation of more realistic and intricate reward structures in the learning process. Offline reinforcement learning has shown promising potential in learning optimal policies when the agent has access to previously collected static datasets. Reward machines offer a way to encode the high-level structure of non-Markovian reward functions. We introduce C-QRM, an offline reinforcement learning approach that employs non-Markovian reward functions specified as reward machines to accomplish complex tasks and learn an optimal policy more efficiently by utilizing the offline dataset. Our objective is to learn a conservative Q-function that decomposes complex high-level reward machines and whose expected value of a policy under this Q-function provides a lower bound to its actual value. C-QRM learns these lower-bounded Q-values, mitigating overestimation bias and improving sampling efficiency. We evaluate the performance of the proposed C-QRM algorithm by comparing it to QRM as a baseline method. The results indicate that C-QRM outperforms QRM with fewer training steps and benefits from the offline dataset.

Keywords: offline reinforcement learning; reward machine; non-Markovian reward function

1. Introduction

Combining reinforcement learning (RL) with expressive deep neural networks has demonstrated great potential in a range of applications, from robotics [1] and strategy games [2] to recommendation systems [3]. In RL, an agent maximizes accumulative reward by learning from its interactions with the environment through trial and error. However, Real-world applications, such as autonomous driving and robotics, often require agents to learn from static datasets due to safety or cost constraints. The agent may not have access to a live environment for learning. Collecting online data can be costly and time-consuming in practice, so the agent may only have access to a previously collected static dataset. This is known as offline reinforcement learning (offline RL), which has become increasingly popular due to its practical applications. Offline RL involves learning effective policies from a static dataset without further environmental interaction. To learn the optimal policy, the agent must be provided with a user-specified reward function. However, this reward function is often programmed as a black box to the agent, making it difficult for the agent to access the structures or high-level ideas the programmer used in defining it [4]. Existing offline RL methods struggle with non-Markovian rewards (e.g., rewards dependent on historical states) and task complexity. To address this problem, we propose a solution where the high-level task specification is encoded by the reward machine in the offline reinforcement learning setting, enabling the agent to learn an optimal policy from the existing dataset while having access to the specification of the reward function. This work bridges reward machines—a formalism for encoding structured tasks—with conservative Q-learning to address these challenges.

Rewards machines can encode high-level tasks for RL agents [5] by allowing the agents to be awarded different rewards at different time steps, enabling the achievement of sub-goals in a temporally extended manner. Optimizing a policy that satisfies these goals, with the maximum expected discounted future reward from all MDP states, promotes the creation of more robust and interpretable policies better suited for generalization to new environments. Furthermore, reward machines are beneficial for capturing high-level objectives and safety requirements that cannot be expressed easily with simple scalar rewards. The reward machine facilitates their incorporation into standard Q-learning to encode non-Markovian reward functions. Nonetheless, applying standard Q-learning to real-world problems consistently poses practical challenges. To address this, we use offline data to train the Q-functions instead of interacting with the environment during training.

The distributional shift issue in offline RL arises when the learned policy deviates from the behavior policy that generated the dataset, leading to erroneous Q-value estimations for out-of-distribution actions [6]. To address this problem, we leverage the framework of Conservative Q-Learning (CQL) [7], which explicitly enforces a lower bound on Q-values through a regularization term. Specifically, CQL minimizes the expected Q-value under a learned policy while maximizing it under the behavior policy, thereby mitigating overestimation errors caused by distributional shifts.

This paper presents a novel offline reinforcement learning (RL) method, called C-QRM, that combines conservative Q-learning with reward machines to enhance sampling efficiency using unbiased or biased offline data. The conservative Q-learning technique ensures that the agent learns the optimal policy without deviating from the provided offline dataset. The effectiveness of C-QRM is evaluated on various benchmark environments and compared to existing Q-learning for reward machines (QRM) [4] methods. The experimental results demonstrate that C-QRM outperforms existing methods regarding sample efficiency, stability, and convergence guarantee, offering a promising approach for offline RL with complex and non-Markovian reward function specifications. To the best of our knowledge, this is the first work to investigate offline RL with non-Markovian reward functions using formal methods.

2. Related Work

Offline reinforcement learning: Offline RL, also known as batch reinforcement learning, is a powerful data-driven learning paradigm that learns the static pre-collected data via interaction with the environment [6,8]. In this setting, the offline policy is learned and improved over the behavior policy (i.e. the policy used to collect the offline dataset). The popularity of offline reinforcement learning has increased dramatically in recent years because of its potential to remove the barrier between online reinforcement learning methods and real-world applications, such as robotics [9–11], healthcare [12], and dialogue systems [13]. Even though the widespread offline reinforcement learning demonstrates such capability, it still suffers the unexpected behavior caused by the error of distributional shift. This is one of the core challenges of offline RL [6,8,14–16]. Several existing works have focused on solving the issue of distributional shifts. The authors proposed Batch-constrained deep Q-Learning (BCQ) in [17] to constrain the learned policy with minimal mismatch to the behavior policy concerning the dataset. This method relies on the idea of "staying close" to the dataset using the estimate of behavior policy [18]. However, the learned policy may still differ far from the behavior policy if the action moves far from the action in the offline data. A policy regulation approach was introduced in [19] to improve the robustness of policy selection. The work mentioned above mainly aims to prevent the learned policy from deviating from the behavior policy in the collected data. On the other hand, the Lyapunov stability in control theory is integrated with the density model to avoid distributional shifts in [20]. These methods are primarily based on estimating the density and the distribution of the provided static dataset. Besides solving the distributional issue, integrating the Trans-former [21] model with reinforcement learning has attracted more interest in recent years. The reinforcement learning problems can be unified under a sequence model [22], taking the entire state-action trajectory as input. Trajectory transformer is proposed in [22], closely related to Decision Transformer [23], but they have different purposes. Trajectory transformer output the sequence of

states, actions, and rewards as a planner in a long-horizon sparse-reward task. The decision transformer aims to output the reward-conditioned action only.

RL with non-Markovian reward functions: Our work is closely related to RL with non-Markovian reward functions. In real-world scenarios, obtaining or designing immediate rewards for a control action is often challenging, such as scoring each move of an autonomous vehicle at every instant, which can be time-and resource-intensive [24]. In contrast, evaluating the entire trajectory after the agent completes a complex task with a non-Markovian reward function is much easier. Several RL approaches exist where non-Markovian reward functions are expressed using reward machines or temporal logic specifications [5, 25–33]. For instance, in [5], the authors introduced Q-learning for reward machine (QRM) and demonstrated its ability to converge to an optimal policy in the tabular case with high certainty. QRM also outperforms Q-learning and hierarchical RL for tasks where reward machines can represent the reward functions. In [25], the authors conceptualized and developed a *Joint Inference of Reward machines and Policies* (JIRP) algorithm for RL. Specifically, the approach infers high-level knowledge in the form of reward machines during RL, and the inferred high-level knowledge can effectively guide the future explorations of the learning agent. These works have yet to utilize offline datasets to expedite the RL process.

3. Preliminaries

Reinforcement Learning (RL): In reinforcement learning, the goal is to learn a policy that maximizes the expected cumulative discounted reward of an agent interacting with an unknown environment, which is modeled in a Markov decision process (MDP). One of the most popular RL algorithms is Q-learning, which involves learning an optimal policy π^* maximizes the expected cumulative dis-counted reward in the MDP. However, in some cases, Q-learning can result in excessive exploration and suboptimal decision-making, leading to undesirable outcomes.

Definition 1. *A labeled Markov decision process (MDP) is defined by a tuple (S, A, T, r, γ, P, L), where S is a finite state space and A is a finite action space. $T(s_{t+1} \mid s, a)$ is the dynamic transition probability distribution. r is the reward function and $\gamma \in (0, 1]$ is the discounted factor. Lastly, P is a finite set of propositional variables, and $L : S \times A \times S \rightarrow 2^P$ is the labeling function representing the set of relevant high-level events the agent senses while interacting with the environment.*

In the MDPs considered in this paper, we adopt a slightly different setting where the reward function r is defined throughout history. In contrast, the transition function is still Markovian (i.e., MDPs with non-Markovian reward functions). Each transition of an MDP has an associated label defined by labeling function L and a set of propositions P, which is known to the agent. The labels are expert knowledge of successfully executing a task. The reward can be easily expressed.

Reward Machine: To specify the reward associated with a task in RL, we use the formalism of the reward machine. It is a finite-state machine describing each state's expected reward and action. The reward machine is constructed using a set of reward functions defined for each state-action pair. The reward machines enable the agent to learn the optimal policy by maximizing the expected reward while satisfying the constraints specified by the reward machines.

Definition 2. *Reward machine $A = (V, v_1, 2^P, \mathbb{R}, \delta, \sigma)$ consists of a finite, nonempty set V of reward machine states, an initial reward machine state $v_1 \in V$, an input alphabet 2^P where P is a finite set of propositional variables, an output alphabet $\mathbb{R}$, a (deterministic) transition function $\delta : V \times 2^P \rightarrow V$, and an output function $\sigma : V \times 2^P \rightarrow \mathbb{R}$.*

Offline Reinforcement Learning: The offline RL problem can be considered as a data-driven method in RL settings where the agent has access to a dataset which consists of pre-collected experiences. The agent uses the pre-collected static dataset, $\mathbb{D} = \{(s_i, a_i, r_i, s_i')\}$, to learn the optimal policy $\pi(a \mid s, v)$ to obtain the maximized cumulative reward. For the offline RL, the behavior policy π_β is often generated by the expert to guide the agent in interacting with the environment. 'Here we use the empirical behavior policy $\hat{\pi}_\beta(a \mid s, v) := \frac{\sum_{s',v',a' \in \mathbb{D}} \mathbf{1}[s=s', v=v', a=a']}{\sum_{s',v' \in \mathbb{D}} 1[s=s', v=v']}$ to collect the dataset $\mathbb{D}$. Q-learning is a model-free reinforcement learning algorithm that involves learning an optimal policy by iteratively

updating the Q-values for each state-action pair. Conservative Q-learning (CQL) [7] is a variant of Q-learning or actor-critic algorithm in the conservative Q-function lower bounds of the true Q-value. The Soft Actor-Critic (SAC) algorithm constructs the backbone of the CQL algorithm. The CQL algorithm involves a parametric Q-function $Q_\theta(s, v, a)$ and parametric policy $\pi_\phi(a \mid s, v)$. $\hat{Q}(s, v, a)$ is the learned update that lower bounds the true Q-value. It shows that the $\hat{Q}$ lower bounds the Q. The Bellman operator is defined as $\mathbb{B}Q(s, v, a) = r(s, v, a) + \gamma \mathbb{E}\left[\max_a Q(s', v')\right]$. The parameters of the Q-function can be trained to minimize the residual error of $J_Q(\theta) = \mathbb{E}_{s\sim\mathbb{D}, a\sim\mathbb{D}}\left[\frac{1}{2}\left(Q_\theta(s, v, a) - \mathbb{B}\hat{Q}_\theta(s, v, a)\right)^{\frac{1}{2}}\right]$. The goal is to approximate the value function $V(s, v)$ of the policy π_θ given the offline dataset $\mathbb{D}$. We define the policy value under $\hat{Q}$ as $\hat{V}$. In the policy improvement stage, where $\hat{\pi} = \arg\max_\pi \mathbb{E}_{s\sim\mathbb{D}, a\sim\pi(a|s,v)}\left[\hat{Q}(s, v, a)\right]$.

4. Problem Statement

In many real-world scenarios, assuming a Markovian reward is unrealistic because historical states and actions may significantly influence the current reward. For example, controlling autonomous vehicles depends on the vehicle's historical actions, not just the current state. In Healthcare applications, treatment efficacy may rely on cumulative drug doses over time, also violating Markovian assumptions. Thus, in this paper, we consider the reward function to be non-Markovian. We use a reward machine to encode the non-Markovian reward functions while taking the previous states and actions into consideration. Moreover, the reward machine allows the agent to comprehend the high-level idea behind the user-defined reward structure. Our goal is to enable the agent to learn an optimal policy from the offline dataset and decompose the high-level complex task specification. Task complexity is evaluated through two dimensions: (1) the hierarchical structure of the reward machine, where states and transitions encode temporal dependencies (e.g., sequential subtasks in the office world), and (2) the non-Markovian reward function, which requires the agent to infer historical state-action pairs to compute rewards (e.g., traffic rule compliance).

Q-learning methods often fail to learn on static, off-policy data. Since the Q-function is trained only on actions according to the dataset that is sampled from the behavior policy, and the target value of the Bellman equation uses the action values based on the learned policy, offline reinforcement learning algorithms that are based on behavior policy suffer from this distributional shift during the training process. Furthermore, the learned policy is trained to maximize the Q-value in the Bellman update, which can lead to a bias towards out-of-distribution actions that produce high Q-values. Therefore, this Q-value is erroneously high, which may cause the Q-function converges with an error. Because the dataset is static and interacting with the environment is not available, correcting this Q-value and the out-of-distribution actions in offline reinforcement learning is challenging. Therefore, in this paper, we use conservative Q-learning to solve this erroneously Q-value updates and out-of-distribution actions issue.

Problem 1. Given a static offline dataset $\mathbb{D} = \{s, a, s', r\}$, a labeled $MDP = (S, A, T, r, \gamma, P, L)$, where the non-Markovian reward function r is encoded by the reward machine $A = (V, v_1, 2^P, R, \delta, \sigma)$. Learn the optimal policy of the agent for maximizing the cumulative reward.

Learning from the static offline data for the Q-learning method is challenging and rarely successful due to the out-of-distribution actions. Simply increasing the volume of the dataset is not helpful. In the Bellman update, maximizing the learned Q-values using the new action at the new state give the target Q-value. However, the target Q-value may contain an error if the Q-function was naively maximized since the Q approximation is produced based on the actions that are far outliers of the training dataset. Particularly, we denote $\eta_i(s, v, a) = Q_i(s, v, a) - Q^*(s, v, a)$ as the total error at each iteration i of Q-learning, and denote $e_i(s, v, a) = Q_i(s, v, a) - \mathbb{B}Q_{i-1}(s, v, a)$ as the current Bellman

error, where $\mathbb{B}Q(s, v, a) = r(s, v, a) + \gamma\mathbb{E}[\max Q(s', v', a')]$. Then the total error $\eta_i(s, v, a) \leq \delta_i(s, v, a) + \gamma \max_{a} \mathbb{E}_{s'}[\eta_{i-1}(s', v', a')]$. It suggests that the current Bellman error is the sum of the discounted error from (s', v', a') and the new error $\delta_i(s, v, a)$ from the current iteration in Q-learning. This paper uses the strategy to lower bound the Q-values to avoid these out-of-distribution action errors.

5. Conservative Q-learning for Reward Machines

This section presents the Conservative Q-learning for Reward Machine (C-QRM) algorithm. We begin by collecting the dataset using the QRM algorithm as the empirical behavior policy within the designed environment. Next, this dataset is then used to train the CQL network. Finally, to update the Q-value, we apply Conservative Q-learning, which involves adding a regularization term to the Q-function as a penalty. This regularization term ensures that the expected value of a policy under this Q-function is lower-bounded by its true value, thereby avoiding the distribution shift problem.

Algorithm 1 presents the pseudocode for C-QRM. The Q-function estimator and the policy are parametrized in this algorithm with a neural network parameter θ and ϕ as shown in the hyperparameter section. The input to the neural network consists of state and action data dimensions and high-level re-ward machine states. The reward machine A and the offline dataset $\mathbb{D}$are given as the input. We use empirical behavior policy to generate the offline dataset by observing the QRM in the environments In line 1. The parametrized Q-function Q_θ is first initialized. There are three *for* loops in the main body of the C-QRM Algorithm. In the first *for* loop, we iterate over the episode of RL. In each episode, the low-level MDP state s and the high-level reward machine state v are initialized in lines 3–4. In the second *for* loop (lines 5 to 10), the training step is iterated from 0 to *eplength*. The action is first sampled from the $\pi_\phi(a \mid s, v)$ distribution. The next state s' is then obtained by executing the selected action at the old state (line7). The next high-level reward machine state and new reward are obtained based on the previous reward machine state v and the labeling function constructed using a tuple (s, a, s') in line 8. The reward is obtained from the output function on the reward machines in line 9. Then, the Q-function is updated using the CQL objective function in line 10. The $\hat{Q}_\theta^{t+1}$ is the Q-value that lower bounds the true Q-value. In this line, the expectation of the Q-value is based on the action sampled from a particular distribution of state-action pair $\mu(a \mid s, v)$. The additional maximization expectation Q-value under the empirical behavior policy $\hat{\pi}_\beta(a \mid s, v)$ substantially improves the bound so that the action selection is biased toward the dataset. Then the last term is the standard Bellman error, where the $\mathbb{B}^\pi$ is the Bellman operator and the $\hat{Q}_\theta^t$ is the parametrized target Q-value. The off-policy imp ovement is evaluated in line 11 using gradient descent. We then have $\mathbb{E}_{\pi(a|s,v)}\left[\hat{Q}^\pi(s, v, a)\right] \leq V^\pi(s, v)$ when $\mu(a \mid s, v) = \pi(a \mid s, v)$.

In the third loop, we loop through all other reward machine states that have not been experienced, denoted as $\tilde{v}$ from V. We obtain the next reward machine state $\tilde{v}'$ similarly by transition function δ and the reward $\tilde{r}$ by output function σ. First, the target value is updated using the reward $\tilde{r}$. Then the lower bounded is updated in a similar way as line 10. Next, the policy π_ϕ is updated similarly to line 11. Finally, the MDP state s moves to the next state s', and the previous reward machine state v move to the next reward machine v'.

Under mild conditions, Theorem 1 below provides the theoretical guarantee that the proposed method learns Q-value approximations that provide a lower bound to the actual Q-function [7].

Algorithm 1: Conservative Q-learning for reward machine

Hyperparameter: episode length eplength, tradeoff factor α, learningrate γ, parametric Q-function parameter θ, parametric policy parameter ϕ

Input: A reward machine $A = (V, v_I, 2^P, \mathbb{R}, \delta, \sigma)$, an offline dataset $\mathbb{D} = \{(s_i, a_i, s'_i, r_i)\}$, for each trajectory i

1 $Q_\theta \leftarrow initialQFunctions()$

2 for $episode = 1, 2, \cdots$ **do**

3 $s \leftarrow InitialState()$

4 $v \leftarrow InitialRMState()$

5 for $0 \le t < eplength$ **do**

6 $a \sim \hat{\pi}_\phi(a \mid s, v)$

7 $s' \leftarrow ExecuteAction(s,a)$

8 $v' \leftarrow \delta(v, L(s, a, s'))$

9 $r \leftarrow \sigma(v, L(s, a, s'))$

10 $$\hat{Q}_\theta^{t+1}(s,v,a) \leftarrow \arg\min_Q \alpha\left[\mathbb{E}_{s\sim\mathbb{D},a\sim\mu(a|s,v)} Q_\theta(s,v,a) - \mathbb{E}_{a\sim\mathbb{D},a\sim\hat{\pi}_\beta(a|s,v)} Q_\theta(s,v,a)\right] + \frac{1}{2}\mathbb{E}_{s,a,s'\sim\mathbb{D}}\left[(Q_\theta(s,v,a) - \mathbb{B}^\pi \hat{Q}_\theta^t(s,v,a))^2\right]$$

11 $$\phi_t \leftarrow \phi_{t-1} - \gamma_\pi \mathbb{E}_{s\sim\mathbb{D},a\sim\hat{\pi}_{\hat{Q}(\cdot|s,v)}}\left[\hat{Q}_\theta(s,v,a) - \log \hat{\pi}_\phi(a \mid s, v)\right]$$

12 for $\tilde{v} \in V \setminus \{v\}$ // excludingv

13 do

14 $\tilde{v}' \leftarrow \delta\left(\tilde{v}, L(s,a,s')\right)$

15 $\tilde{r} \leftarrow \sigma\left(\tilde{v}, L(s,a,s')\right)$

16 update $\mathbb{B}\hat{Q}_\theta^t(s,v,a)$ using reward $\tilde{r}$

17 update $\hat{Q}_\theta^{t+1}(s,\tilde{v},a)$ in a similar way as line 10

18 improve the policy $\hat{\pi}_\phi$ in a similar way as line 11

19 $s \leftarrow s'$

20 $v \leftarrow v'$

21 end for

22 return $\hat{Q}_\theta$

Theorem 1. *[7](CQL learns lower-bounded Q-values) Let* $\pi_{\hat{Q}^t}(a \mid s, v) \propto exp\left(\hat{Q}^t(s,v,a)\right)$ *where* $\pi_{\hat{Q}^t}$ *is the learned policy with the lower-bounded Q-function* $\hat{Q}^t$ *at the t-th iteration, and assume that* $D\left(\hat{\pi}^{t+1}, \pi_{\hat{Q}^t}\right) \le \epsilon$, *where* $D\left(\hat{\pi}^{t+1}, \pi_{\hat{Q}^t}\right)$ *is the total variation distance between the two policies* $\hat{\pi}^{t+1}$ *and* $\pi_{\hat{Q}^t}$. *Then the policy value under* $\hat{Q}^t$ *lower bounds the actual policy value, i.e.,* $\hat{V}^{t+1}(s,v) \le V^{t+1}(s,v)$ *for every s and v, where* $\hat{V}^{t+1}(s,v)$ *is the lower-bounded expected value function, if*

$$\mathbb{E}_{\pi_{\hat{Q}^t}(a|s,v)}\left[\frac{\pi_{\hat{Q}^t}(a \mid s,v)}{\hat{\pi}_\beta(a \mid s.v)} - 1\right] \ge \sum_a \left[\frac{\pi_{\hat{Q}^t}(a \mid s,v)}{\hat{\pi}_\beta(a \mid s.v)}\right] \cdot \epsilon \quad (1)$$

Proof of Theorem 1. Following the proof of Theorem 3.3 in [7], We calculate the change in the policy value, $\hat{V}^{t+1}$, that results from using the updated Q-value, $\hat{Q}^{t+1}$, with respect to the previous iterate $\mathbb{B}^\pi \hat{Q}^t$.

$$\mathbb{E}_{\hat{\pi}^{t+1}(a|s,v)}\left[\hat{Q}^{t+1}(s,v,a)\right] = \mathbb{E}_{\hat{\pi}^{t+1}(a|s,v)}\left[\mathbb{B}^{\pi}\hat{Q}^{t}(s,v,a)\right] - \mathbb{E}_{\hat{\pi}^{t+1}(a|s,v)}\left[\frac{\hat{\pi}_{\hat{Q}^t}(a \mid s,v)}{\hat{\pi}_{\beta}(a \mid s,v)} - 1\right]$$

$$= \mathbb{E}_{\hat{\pi}^{t+1}(a|s,v)}\left[\mathbb{B}^{\pi}\hat{Q}^{t}(s,v,a)\right] - \mathbb{E}_{\hat{\pi}_{\hat{Q}^t}(a|s,v)}\left[\frac{\hat{\pi}_{\hat{Q}^t}(a \mid s,v)}{\hat{\pi}_{\beta}(a \mid s,v)} - 1\right]$$

$$+ \sum_{a}\left(\hat{\pi}_{\hat{Q}^t}(a \mid s,v) - \hat{\pi}^{t+1}(a \mid s,v)\right)\frac{\hat{\pi}_{\hat{Q}^t}(a \mid s,v)}{\hat{\pi}_{\beta}(a \mid s,v)}$$

If $\mathbb{E}_{\hat{\pi}_{\hat{Q}^t}(a|s,v)}\left[\frac{\hat{\pi}_{\hat{Q}^t}(a|s,v)}{\hat{\pi}_{\beta}(a|s.v)} - 1\right] \geq \sum_{a}\left[\frac{\hat{\pi}_{\hat{Q}^t}(a|s,v)}{\hat{\pi}_{\beta}(a|s.v)}\right] \cdot \epsilon$, as for any a

$$D\left(\hat{\pi}^{t+1}, \hat{\pi}_{\hat{Q}^t}\right) = \max_{a}\left[\hat{\pi}_{\hat{Q}^t}(a \mid s,v) - \hat{\pi}^{t+1}(a \mid s,v)\right] \leq \epsilon,$$

We have $\mathbb{E}_{\hat{\pi}_{\hat{Q}^t}(a|s,v)}\left[\frac{\hat{\pi}_{\hat{Q}^t}(a|s,v)}{\pi_{\beta}(a|s,v)} - 1\right] \geq \sum_{a}\left(\hat{\pi}_{\hat{Q}^t}(a \mid s,v) - \hat{\pi}^{t+1}(a \mid s,v)\right)\frac{\hat{\pi}_{\hat{Q}^t}(a|s,v)}{\pi_{\beta}(a|s,v)}$. Therefore, $\mathbb{E}_{\hat{\pi}^{t+1}(a|s,v)}\left[\hat{Q}^{t+1}(s,v,a)\right] \leq \mathbb{E}_{\hat{\pi}^{t+1}(a|s,v)}\left[\mathbb{B}^{\pi}\hat{Q}^{t}(s,v,a)\right]$, and through recursive reasoning, we can conclude that the calculated Q-value underestimates the optimal Q-value.

6. Simulation

In this section, we conducted two case studies in the traffic world and office world environment, respectively, to evaluate the effectiveness of the proposed C-QRM algorithm. In both cases, we compare C-QRM and QRM with unbiased and biased data accordingly.

- C-QRM: Our proposed algorithm in Section 5. It integrates the CQL algorithm with QRM to estimate the expected value of a policy under the learned Q-function to lower bound the true Q-value in a complex world where the high-level task is specified in reward machines.
- QRM (Q-learning for Reward Machines): We use the QRM algorithm, i.e., Algorithm 1 from [4].

6.1. Data Collection

Biased offline data collection: Using the QRM as the baseline method in both the traffic world and office world environments, we first collect the results of using QRM to learn an optimal policy which includes the trajectories of the composition of the state, next state, action, and reward for each episode. For the dataset preparation, each state and the corresponding actions are collected. By generating the action distribution according to different states, we determine the best action at a particular state with the highest probability. Since the result contains the entire training process, the data is biased since it involves the data produced by the suboptimal policy and causes noise in the whole dataset. We then train the C-QRM algorithm to learn an optimal policy from this biased offline dataset.

Unbiased offline data collection: We generate a separate dataset following an optimal policy. In other words, this dataset contains the trajectories after the algorithm converges to an optimal policy. We collect the data after different episodes for different case studies until the cumulative reward converges under an optimal policy. This dataset is unbiased to select the best actions in certain states. This dataset helps the C-QRM algorithm to converge faster to an optimal policy.

In the following two subsections, we compare the results with unbiased and biased data for the traffic world and office world environments, respectively.

6.2. Traffic World Environment

We apply C-QRM to an autonomous vehicle scenario as introduced in [25]. We consider an autonomous vehicle that needs to drive from position "A" to position "B" on a map (see Figure 1),

following traffic rules. To simplify things, we will only focus on the right-of-way rules and how the vehicle behaves at intersections concerning the traffic from the intersecting roads. We will also make two assumptions: (1) the vehicle correctly senses if it is on a priority road, and (2) the vehicle always drives straight forward and stays on the road when not at intersections.

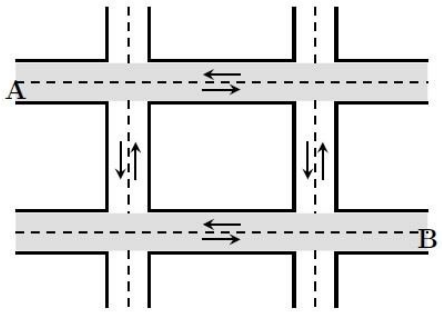

Figure 1. Map of a residential area in the traffic world environment.

The vehicle complies with the traffic rules if and only if it travels on an ordinary road and stops for a one-time unit at the intersections; or it travels on a priority road and does not stop at the intersections. We aim to accomplish this task using episodic reinforcement learning with an offline dataset. Specifically, after each episode of 100 time units, the vehicle will receive a reward of 1 if it reaches position B while obeying the traffic rules. Otherwise, it will receive a reward of 0. It is important to note that the reward function is non-Markovian because the reward depends on the current state and past states, which determines whether the traffic rules were followed.

The reward machine specifying the high-level task in the traffic world is shown in Figure 2. The set of actions is A = {straight, left, right, stay}, corresponding to going straight, turning left, turning right, and staying in place. For simplicity, we assume that the labeled MDP is deterministic (i.e., the slip rate is zero for each action). We set $eplength = 2000$, $N = 100$. Figure 3 shows the rewards attained with the C-QRM and QRM methods in the autonomous vehicle scenario trained with the unbiased dataset. C-QRM converges to an optimal policy within 15,000 training steps, while QRM converges to optimal policies within 27,000 training steps.

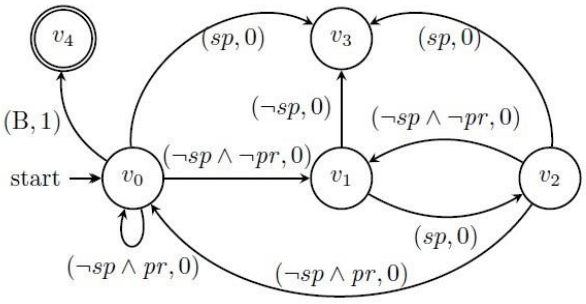

Figure 2. Reward machine for the traffic world environment. *sp*: stop at an inter-section; ¬*sp*: not stop at an intersection; *pr*: end in a priority road; ¬*pr*: end in an ordinary road. An edge (*sp*, 0) between v_0 and v_3 means that the reward machine will transition from v_0 to v_3 if the proposition (label) *sp* becomes true and output a reward equal to zero.

We also show the performance results of comparing C-QRM and QRM with biased data in Figure 4. As it shows, even with the biased dataset that contains suboptimal policy data, the C-QRM can still converge to optimal policy faster than the original QRM method. Specifically, with the biased data, the C-QRM method converges to the optimal policy after 20,000 training steps.

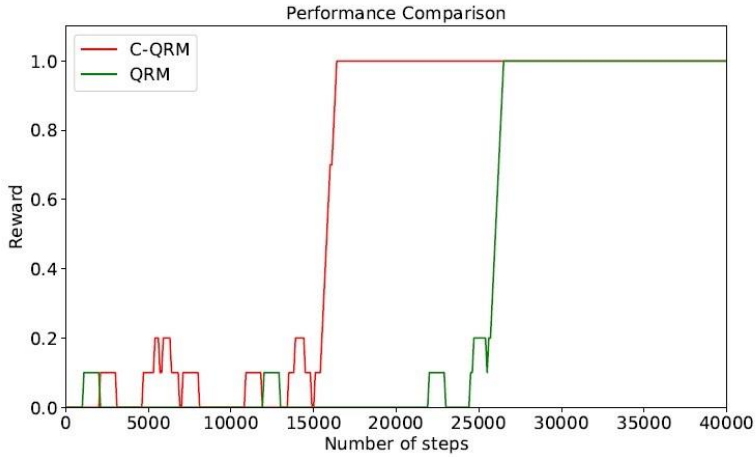

Figure 3. Performance of C-QRM in the traffic world environment with unbiased offline data.

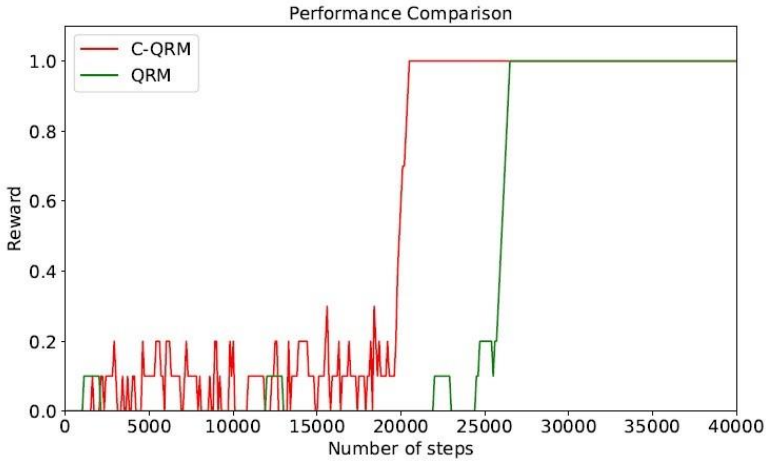

Figure 4. Performance of C-QRM in the traffic world environment with biased offline data.

6.3. Office World Environment

For this case study, we apply the C-QRM algorithm to the office world scenario in the 9 × 12 grid world introduced in [4]. The reward machine specifying the high-level task in the office world is shown in Figure 5. The agent has four possible actions at each time step: move north, move south, move east, and move west. After each action, the robot may slip to each of the two adjacent locations with a probability of 0.05. We use three different high-level subtasks: getting coffee, getting mail, and going to the office. The same hyperparameters are applied in the office environment as the traffic scenario. Figure 6 shows the cumulative rewards with two different methods and compares the performance in the office world environment using an unbiased dataset. C-QRM converges to the optimal policy at about 100,000 training steps, while QRM converges to the optimal policy after 250,000 training steps. With the biased data, the proposed C-QRM algorithm converges to the optimal policy at about 200,000 training steps, as shown in Figure 7.

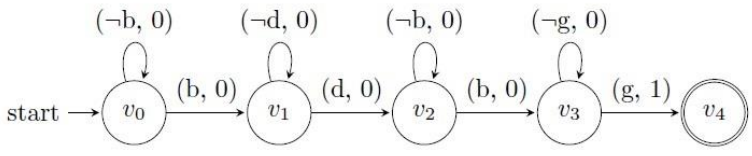

Figure 5. Reward machine for the office world environment. The task consists in visiting locations on the map in order, here b, then d, then b again, and finally.

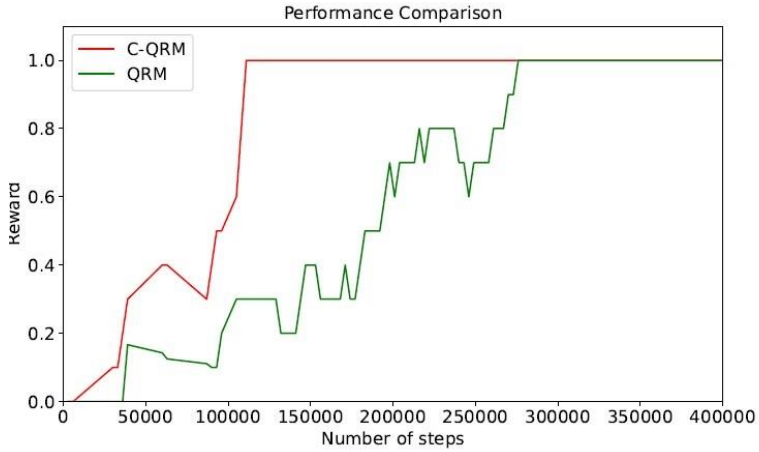

Figure 6. Performance of C-QRM in office world with unbiased offline data.

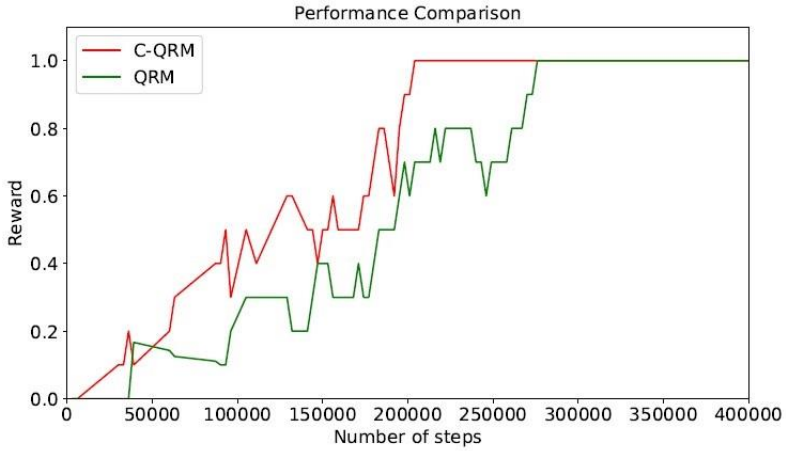

Figure 7. Performance of C-QRM in office world with biased offline data.

7. Conclusion

In this paper, we have addressed the problem of offline reinforcement learning with non-Markovian reward functions. We propose using reward machines to encode these functions and specify high-level tasks for the learning agent in the low-level MDP. Conservative Q-learning is integrated with reward machines to address real-world situations where learning while interacting with the environment is expensive and dangerous. Conservative Q-learning lower bounds the true value of the learned policy to avoid the distributional shift issue with static datasets. We have conducted experiments to demonstrate that our pro-posed method, C-QRM, outperforms QRM in terms of a faster convergence of rewards. Two case studies were conducted to evaluate the results. We hypothesized that integrating reward machines with conservative Q-learning (C-QRM) would improve sample efficiency and convergence speed compared to QRM in offline RL with non-Markovian rewards. Experimental results in Section 6 validate this hypothesis: C-QRM achieved 44% faster convergence in the traffic world (15k *vs*. 27k steps) and 60% faster in the office world (100k *vs*. 250k steps) with unbiased data. Even with biased data, C-QRM maintained superior performance, confirming its robustness to suboptimal datasets. Our work opens new research directions for formal methods with offline reinforcement learning algorithms. Future work could explore how reward machines can be integrated with other offline RL algorithms, such as Batch Constrained Q-learning [17] or the D4RL dataset. Furthermore, future work could focus on scaling to more complex tasks and environments, such as multi-agent scenarios or real-world robotics applications. We will also implement the proposed method on physical robots to complete complex tasks and learn the optimal policy using pre-collected offline datasets.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In Proceedings of 2022 2nd International Conference on Robotics and Control Engineering, Nanjing, China, 25–27 March 2022; pp. 651–673.
2. Vinyals, O.; Babuschkin, I.; Chung, J.; Mathieu, M.; Jaderberg, M.; Czarnecki, W.; Dudzik, A.; Huang, A.; Georgiev, P.; Powell, R.; et al. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. Available online: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/ (accessed on 20 March 2025).
3. Li, L.; Chu, W.; Langford, J.; Schapire, R.E. A contextual-bandit approach to personalized news article recommendation. In Proceedings of WWW '10: The 19th International World Wide Web Conference, Raleigh North, CA, USA, 26–30 April 2010, https://doi.org/10.1145/1772690.1772758.
4. Icarte, R.T.; Klassen, T.Q.; Valenzano, R.A.; Mcllraith, S.A. Using reward machines for high-level task specification and decomposition in reinforcement learning. In Proceedings of the International Conference on Machine Learning (ICML 2018), Stockholm, Sweden, 10–15 July 2018; pp. 2112–2121.
5. Toro Icarte, R.; Waldie, E.; Klassen, T.; Valenzano, R.; Castro, M.; McIlraith, S. Learning reward machines for partially observable reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 15497–15508.
6. Levine, S.; Kumar, A.; Tucker, G.; Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint* 2020, arXiv:2005.01643, https://doi.org/10.48550/arXiv.2005.01643.
7. Kumar, A.; Zhou, A.; Tucker, G.; Levine, S. Conservative q-learning for offline reinforcement learning. *Adv. Neural. Inf. Process. Syst.* **2020**, *33*, 1179–1191.
8. Prudencio, R.F.; Maximo MR, O.A.; Colombini, E.L. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *35*, 10237–10257, https://doi.org/10.1109/TNNLS.2023.3250269.
9. Sinha, S.; Mandlekar, A.; Garg, A. S4RL: Surprisingly Simple Self-supervision for Offline Reinforcement Learning in Robotics. In Proceedings of the 5th Conference on Robot Learning (CoRL), Auckland, New Zealand, 14–18 December 2022; pp. 907–917.
10. Levne, S.; Pastor, P.; Krizhevsky, A.; Ibarz, J.; Quillen, D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int. J. Robot. Res.* **2017**, *37*, 421–436, https://doi.org/10.1177/0278364917710318.
11. Rafailov, R.; Yu, T.H.; Rajeswaran, A.; Finn, C. Offline reinforcement learning from images with latent space models. In Proceedings of the 3rd Annual Conference on Learning for Dynamics and Control (L4DC), ETH Zurich, Zurich, Switzerland, 7–8 June 2021; pp. 1154–1168.
12. Nie, X.; Brunskill, E.; Wager, S. Learning when-to-treat policies. *J. Am. Stat. Assoc.* **2021**, *116*, 392-409, https://doi.org/10.1080/01621459.2020.1831925.
13. Jaques, N.; Ghandeharioun, A.; Shen, J.H.; Ferguson, C.; Lapedriza, A.; Jones, N.; Picard, R. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv* **2019**, arXiv:1907.00456, https://doi.org/10.48550/arXiv.1907.00456.
14. Yu, T.; Kumar, A.; Rafailov, R.; Rajeswaran, A.; Levine, S.; Finn, C. Combo: Conservative offline model-based policy optimization. *Adv Neural Inf Process Syst.* **2021**, *34*, 28954-28967.
15. Foster, D.J.; Krishnamurthy, A.; Simchi-Levi, D.; Xu, Y. Offline reinforcement learning: Fundamental barriers for value function approximation. *arXiv preprint* **2021**, arXiv:2111.10919, https://doi.org/10.48550/arXiv.2111.10919.
16. Kostrikov, I.; Nair, A.; Levine, S. Offline reinforcement learning with implicit q-learning. *arXiv preprint* **2021**, arXiv:2110.06169, https://doi.org/10.48550/arXiv.2110.06169.
17. Fujimoto, S.; Meger, D.; Precup, D. Off-policy deep reinforcement learning without exploration. In Proceedings of the 36th International Conference on Machine Learning (ICML 2019), Long Beach, CA, USA, 9–15 June 2019.
18. Ghasemipour, S.K.S.; Schuurmans, D.; Gu, S.S. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In Proceedings of the International Conference on Machine Learning (PMLR), Virtual, 18–24 July 2021; pp. 3682–3691.

19. Kumar, A.; Fu, J.; Soh, M.; Tucker, G.; Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Adv Neural Inf Process Syst.* **2019**, *32*, 11761–11771.
20. Kang, K.; Gradu, P.; Choi, J.J.; Janner, M.; Tomlin, C.; Levine, S. Lyapunov density models: Constraining distribution shift in learning-based control. In International Conference on Machine Learning, Baltimore, MA, USA, 17–23 July 2022, pp. 10708-10733.
21. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All You Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 6000–6010.
22. Janner, M.; Li, Q.; Levine, S. Offline reinforcement learning as one big sequence modeling problem. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 1273–1286.
23. Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 15084–15097.
24. Xu, T.; Wang, Y.; Zou, S.; Liang, Y. Provably Efficient Offline Reinforcement Learning With Trajectory-Wise Reward. *IEEE Trans. Inf. Theory* **2024**, *70*, 6481–6518, https://doi.org/10.1109/tit.2024.3427141.
25. Xu, Z.; Gavran, I.; Ahmad, Y.; Majumdar, R.; Neider, D.; Topcu, U.; Wu, B. Joint inference of reward machines and policies for reinforcement learning. In Proceedings of the International Conference on Automated Planning and Scheduling, Virtual Event, 17–23 July 2022, pp. 590-598, https://doi.org/10.1609/icaps.v30i1.6756.
26. Rens, G.; Raskin, J.F. Learning non-Markovian reward models in MDPs. *arXiv preprint* **2020**, arXiv:2001.09293, https://doi.org/10.48550/arXiv.2001.09293.
27. Rens, G.; Raskin, J.F.; Reynouad, R.; Marra, G. Online learning of non-markovian reward models. *arXiv preprint* **2020**, arXiv:2009.12600, https://doi.org/10.48550/arXiv.2009.12600.
28. Furelos-Blanco, D.; Law, M.; Russo, A.; Broda, K.; Jonsson, A. Induction of subgoal automata for reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020, pp. 3890-3897, https://doi.org/10.1609/aaai.v34i04.5802.
29. Xu, Z.; Wu, B.; Ojha, A.; Neider, D.; Topcu, U. Active finite reward automaton inference and reinforcement learning using queries and counterexamples. In International Cross-Domain Conference, CD-MAKE 2021, Virtual Event, 17–20 August 2021, pp. 115-135, https://doi.org/10.1007/978-3-030-84060-0_8.
30. Hasanbeig, M.; Jeppu, N.Y.; Abate, A.; Melham, T.; Kroening, D. DeepSynth: Automata Synthesis for Automatic Task Segmentation in Deep Reinforcement Learning. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 7647–7656, https://doi.org/10.1609/aaai.v35i9.16935.
31. Neider, D.; Gaglione, J.-R.; Gavran, I.; Topcu, U.; Wu, B.; Xu, Z. Advice-Guided Reinforcement Learning in a non-Markovian Environment. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 9073–9080, https://doi.org/10.1609/aaai.v35i10.17096.
32. Memarian, F.; Xu, Z.; Wu, B.; Wen, M.; Topcu, U. Active task-inference-guided deep inverse reinforcement learning. In 2020 59th IEEE Conference on Decision and Control (CDC), Jeju Island, South Korea,14–18 December 2020, pp. 1932-1938, https://doi.org/10.1109/CDC42340.2020.9304190.
33. Xu, Z.; Topcu, U. Transfer of temporal logic formulas in reinforcement learning. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence. Macao, China, 10–16 August 2019, p. 4010, https://doi.org/10.24963/ijcai.2019/557.