Two-Step Offline Preference-Based Reinforcement Learning on Explicitly Constrained Policies

Anonymous authors
Paper under double-blind review

Abstract

Preference-based reinforcement learning (PBRL) in the offline setting has succeeded greatly in industrial applications such as chatbots. A two-step learning framework that learns a reward model from the dataset first and then optimizes a policy over the learned reward model through reinforcement learning has been widely adopted for the problem. However, such a method faces challenges from the risk of reward hacking and the complexity of reinforcement learning. To overcome the challenge, our insight is that both challenges come from the state-actions not supported in the dataset. Such state-actions are unreliable and increase the complexity of the reinforcement learning problem. Based on the insight, we develop a novel two-step learning method called PRC: preference-based reinforcement learning on explicitly constrained policies. The high-level idea is to limit the reinforcement learning agent to optimize over policies supported on an explicitly constrained action space that excludes the out-of-distribution state-actions. We empirically verify that our method has high learning efficiency on various datasets in robotic control environments.

1 Introduction

Deep reinforcement learning (DRL) is a fundamental learning paradigm for solving sequential decision-making problems and has rich real-world applications (Sutton & Barto, 2018; Luong et al., 2019; Haydari & Yılmaz, 2020). However, traditional DRL approaches require numerical reward feedback during learning, which might be hard to design or obtain in practice. In contrast, binary preference feedback is usually more accessible (Wirth & Fürnkranz, 2013; Wirth et al., 2017; Fürnkranz et al., 2012; Akrour et al., 2012). For example, to judge how well a robot dances, it is challenging to rate each move in a dance. However, it is relatively easier to tell if a piece of dance is better than another. As a result, preference-based reinforcement learning (PBRL) that only requires preference feedback has become a more realistic learning paradigm and is applied in real-world applications (Christiano et al., 2017). Furthermore, as a special instance of PBRL, reinforcement learning from human feedback (RLHF), which utilizes human preferences, has drawn considerable attention and achieved significant success in many NLP tasks (Ouyang et al., 2022; Rafailov et al., 2023). In this work, we focus on PBRL in the offline setting (Zhan et al., 2023; Zhu et al., 2023) where the agent trains on a pre-collected preference dataset on pairs of trajectories. Offline learning is considered to be more practical as it is usually safer and more convenient to access an offline dataset (Levine et al., 2020; Prudencio et al., 2023; Cheng et al., 2022). Particularly, we are interested in a popular learning framework that is widely applied in various PBRL literature and industrial applications, which we call 'two-step learning.'

Two-step learning framework: In a typical two-step learning framework (Ibarz et al., 2018; Christiano et al., 2017), the first learning step is reward modeling, where the learning agent approximates the reward model of the environment that can best explain the observations from the training preference dataset. The second learning step is policy optimization, where the learning agent optimizes a policy to achieve a high reward on the reward model learned in the previous step. This step is typically implemented by standard online reinforcement learning algorithms such as PPO (Schulman et al., 2017). There are multiple reasons behind the vast popularity of the two-step learning paradigm: 1. similar to the motivation of inverse reinforcement learning (Arora & Doshi, 2021), the learned reward model provides a succinct and transferable

definition of the task; 2. the method is modular in that each phase of learning can be implemented by existing well-developed methods. Despite its popularity, this method faces two main challenges for offline PBRL.

- **Reward over-optimization/Pessimistic learning:** Pessimistic learning has always been the key challenge in offline learning settings. Given a policy, due to the distribution mismatch between the trajectory distribution of the dataset and that induced by the policy, the agent cannot accurately evaluate the policy if the mismatch is large. Reward over-optimization can happen if an agent overestimates and outputs a policy that achieves a high score on the learned reward model but has poor performance in the real environment.

- **Reinforcement learning complexity:** Implementing Reinforcement learning efficiently is known to be challenging (Dulac-Arnold et al., 2021). In addition, for preference feedback, the reward model is learned from stochastic binary preference signals rather than exact scalar true rewards. This can influence the quality of the learned reward model and might increase the difficulty of policy optimization. This is empirically verified in our experiments from Section 4.

Most studies on offline PBRL seek other learning frameworks to circumvent the two challenges or increase the learning efficiency from aspects. For example, Hejna & Sadigh (2023); An et al. (2023) consider frameworks that do not learn a reward model. Kim et al. (2023); Zhang et al. (2023) considers using more advanced methods for reward modeling and trajectory generation, and the focus is not on solving the two challenges above. Standard two-step learning methods (Christiano et al., 2017) tackle the first challenge by applying KL regularization during the policy optimization step, but the second challenge from reinforcement learning difficulty remains. Due to the popularity and advantages of the two-step learning framework in practice and its advantages as mentioned before, this work aims at increasing the learning efficiency of the two-step learning framework by solving the two challenges above at the same time.

Our contributions. In this work, we propose a novel two-step learning PBRL algorithm called 'PRC' that simultaneously tackles the two main challenges above. Our key insight is that both challenges are induced by the same element: the state actions that are out of the dataset distribution. For pessimistic learning, the agent must avoid the policies that are more likely to visit such state-actions as the agent cannot evaluate their performance accurately. Meanwhile, these out-of-distribution state-actions contribute to the complexity of the learning problem. Therefore, our insight is to constrain the action space to exclude such out-of-distribution state-actions. In this way, our method not only follows the idea of pessimism but also reduces the complexity of the corresponding reinforcement learning problem. In the experiments, we apply our method to the continuous control problems, which are the standard RL benchmark. We construct offline PBRL datasets based on the offline RL benchmark D4RL (Fu et al., 2020). We empirically verify that, compared to most baseline methods, our algorithm always learns a policy of better performance. When compared to a recent state-of-the-art offline PBRL method 'flow to better' (FTB) (Zhang et al., 2023), PRC has similar performance. However, FTB requires access to an additional behavior dataset that is not used by PRC. More importantly, FTB uses the technique of diffusion (Yang et al., 2023), which is computationally expensive. On our benchmark, the PRC algorithm achieves a 7× speedup compared to FTB while achieving similar performance.

2 Preliminary

2.1 Offline Preference Reinforcement Learning Problem

We consider preference-based reinforcement learning in a standard offline setting (Zhan et al., 2023). An environment is characterized by an incomplete Markov Decision Process (MDP) without a reward function $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and $\mathcal{P}$ is the state transition function. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ maps a state s to an action stochastically drawn from $\pi(\cdot|s)$. We denote $\Pi^{\mathcal{S},\mathcal{A}}$ as the class of policies supported on the state space $\mathcal{S}$ and action space $\mathcal{A}$. A trajectory of length t is a sequence of states and actions $(s_1, a_1, \ldots, s_t, a_t, s_{t+1})$ where $a_i \in \mathcal{A}, s_{i+1} \sim P(\cdot|s_i, a_i), \forall i \in [t]$. A preference model F takes a pair of trajectories τ_1, τ_2 of the same length as input, and outputs a preference $\sigma \in \{\succ, \prec\}$,

indicating its preference over the two trajectories, i.e. $\tau_1 \succ \tau_2$ or $\tau_1 \prec \tau_2$. In this work, we assume the standard conventions and consider the Bradley-Terry (BT) family of preference models (Bradley & Terry, 1952). Specifically, there exists a reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, such that the probability distribution of the output preference of F given τ_1, τ_2 satisfies:

$$F^R(\succ | \tau_1, \tau_2) = \frac{\exp(\sum_{(s,a)\in\tau_1} R(s,a))}{\exp(\sum_{(s,a)\in\tau_1} R(s,a)) + \exp(\sum_{(s,a)\in\tau_2} R(s,a))} \tag{1}$$

We define the performance of a policy π on a reward function R as the expected cumulative reward of a trajectory generated by the policy. Formally, we abuse the notation and denote the policy performance as $R(\pi) := \mathbb{E}_{a^i\sim\pi(s^i), s^{i+1}\sim P(\cdot|s^i,a_i)}[\sum_i \gamma^i \cdot R(s^i, a^i)]$, where $\gamma \in [0, 1]$ is a discount factor. An offline preference dataset $D = \{(\tau_1^1, \tau_2^1, \sigma^1), \ldots, (\tau_1^N, \tau_2^N, \sigma^N)\}$ consists of multiple preference data over different trajectory pairs. We use the term 'behavior policy' to represent how the trajectories are generated in the dataset. The action distribution given by the behavior policy is the same as that of the dataset (Wu et al., 2019). A learning agent is given access to the incomplete MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$ of the environment and a dataset D whose preferences are generated by a preference model F. The learning goal is to learn a policy that has high performance on the reward function R of the preference model F. We say a learning algorithm is efficient if it can learn such a high-performing policy.

2.2 Model-Based Two-Step Learning Framework

Here, we introduce a prevalent model-based two-step learning framework widely applied for solving PBRL problems (Christiano et al., 2017) and is the framework for our algorithm in this work. The general algorithm consists of the two learning steps as follows.

1. **Step 1: Reward Modeling.** The agent learns the optimal reward model $\hat{R}$ in $\mathcal{R}$ that minimizes the prediction loss on the preference dataset: $\hat{R} \leftarrow \arg\min_R \mathcal{L}_D^p(R)$.

2. **Step 2: Policy Optimization.** The agent learns the optimal policy on the learned reward model in a constrained policy class: $\hat{\pi} \leftarrow \max_{\pi\in\Pi_C} \hat{R}(\pi)$.

Here, at the first step, $\mathcal{L}_D^p(\cdot, \cdot)$ is a function for prediction loss that evaluates how well a reward model reproduces the preference labels in the dataset. A standard choice is the sum of the negative log probability of the reward model predicting the correct preference: $\mathcal{L}_D^p(R) := \sum_{(\tau_1,\tau_2,\sigma)\sim D} -\log F^R(\sigma|\tau_1, \tau_2)$. At the second step, the agent specifies a constrained policy class $\Pi_C \subseteq \Pi^{\mathcal{S},\mathcal{A}}$ to avoid the risk of reward hacking. A standard choice is policies with limited KL divergence to the behavior policy from which the dataset's trajectories are generated (Christiano et al., 2017). More specifically, the agent specifies an imitation loss function $\mathcal{L}_D^i(\pi)$ that describes how likely an imitation policy π generates the trajectories in the dataset. Then, the agent learns an imitation policy that minimizes the imitation loss $\pi_i \in \arg\min_\pi \mathcal{L}_D^i(\pi)$. This is often achieved by a supervised learning process. Let $\mathrm{KL}(\pi_1, \pi_2) := \mathbb{E}_{\tau\sim\pi_1} \sum_{(s,a)\in\tau} \log \frac{\pi_1(a|s)}{\pi_2(a|s)}$ be the definition of the KL divergence between two policies. Then the constrained policy class can be defined as $\Pi_C := \{\pi : \mathrm{KL}(\pi, \pi^i) \leq \beta\}$, where $\beta > 0$ is a constraint parameter.

In practice, it is hard to directly search for the optimal policy on Π_C specified by the KL regularization constraint above. A practical way to solve the problem is to introduce a Lagrange multiplier to the optimization goal and search for the optimal policy in the entire policy space: $\max_\pi \hat{R}(\pi) - \alpha \cdot \mathrm{KL}(\pi, \pi_i)$ (Zhan et al., 2023), which can be solved by a standard RL algorithm such as PPO (Schulman et al., 2017). Formally, Algorithm 1 below shows a popular practical implementation of two-step learning. The reward model $\hat{R}$ and the imitation policy π_i can be learned by standard supervised fine-tuning methods.

Algorithm 1: Two-step training with KL-regularization and PPO algorithm

Inputs : Environment $(\mathcal{S}, \mathcal{A}, \mathcal{P})$, Dataset D, prediction loss function $\mathcal{L}_D^p$, imitation loss function $\mathcal{L}_D^i$

1. Train a reward model $\hat{R} \leftarrow \arg\min_{R \in \mathcal{R}} \mathcal{L}_D^p(R)$
2. Train an imitation policy $\pi_i \leftarrow \arg\min_{\pi \in \Pi} \mathcal{L}_D^i(\pi)$
3. Run PPO algorithm to solve the optimization problem $\hat{\pi} \leftarrow \arg\max_{\pi \in \Pi^{\mathcal{S},\mathcal{A}}} R(\pi) - \alpha \cdot \mathrm{KL}(\pi, \pi_i)$
4. Output π^*

3 Preference Based Reinforcement Learning on an Explicit Constrained Policies (PRC)

3.1 Sub-Optimality of Policy Optimization on Regularized Learned Reward Model

In this work, we focus on the choice of the constrained action space Π_C in the policy optimization step. Let $\hat{R}$ be the reward function learned in the reward modeling step, and R^* be the true reward function. We define $\Gamma_{\hat{R}}(\pi, \pi_{\text{ref}}) := |[R^*(\pi) - R^*(\pi_{\text{ref}})] - [\hat{R}(\pi) - \hat{R}(\pi_{\text{ref}})]|$ as the estimation error on a policy π given by the reward model $\hat{R}$ compared to a reference policy $\pi_{\text{ref}} \in \Pi^{\mathcal{S},\mathcal{A}}$. We focus on relative rewards because the preference model in Eq. 1 only depends on relative rewards. Intuitively, a policy will have low estimation error if it is likely to visit the state actions covered by the dataset. A rigorous analysis of $\Gamma_{\hat{R}}(\cdot)$ for a reward model $\hat{R}$ learned by deep neural networks based function approximation remains an open question. Based on the definition of $\Gamma_{\hat{R}}(\cdot)$, we can give a theoretical bound on the performance of the policy $\hat{\pi}$ learned at the second step in Proposition 3.1.

Proposition 3.1. *Let R^* be the true reward function, $\hat{R}$ be the learned reward function from the dataset, and $\Gamma_{\hat{R}}(\cdot)$ be the relative estimation error of the learned reward $\hat{R}$. Let $\hat{\pi}$ be the learned policy to approximate the optimal policy in $\hat{R}$, and $\epsilon_{opt} := \max_{\pi' \in \Pi_C} \hat{R}(\pi') - \hat{R}(\hat{\pi})$ be the corresponding optimization error. Then, the performance of the learned policy is at least*

$$R^*(\hat{\pi}) \geq \max_{\pi \in \Pi_C, \pi_{ref} \in \Pi^{\mathcal{S},\mathcal{A}}} \left(R^*(\pi) - \Gamma_{\hat{R}}(\pi, \pi_{ref}) - \max_{\pi' \in \Pi_C} \Gamma_{\hat{R}}(\pi', \pi_{ref}) \right) - \epsilon_{opt}.$$

The proof can be found in the appendix. By Proposition 3.1, we can better understand the difficulties of reward hacking and the complexity of reinforcement learning in the two-step learning framework. Here, the last term $\epsilon_{\text{opt}} := \max_{\pi' \in \Pi_C} \hat{R}(\pi') - \hat{R}(\hat{\pi})$ represents the optimization error during the policy optimization process. Intuitively, the optimization error would be large if the policy optimization problem is hard to solve. The first term $\max_{\pi \in \Pi_C, \pi_{\text{ref}} \in \Pi^{\mathcal{S},\mathcal{A}}} \left(R^*(\pi) - \Gamma_{\hat{R}}(\pi, \pi_{\text{ref}}) - \max_{\pi' \in \Pi_C} \Gamma_{\hat{R}}(\pi', \pi_{\text{ref}}) \right)$ implies a trade-off for the choice of the constrained policy class Π_C. Enlarging the constrained policy space can include policies of higher performance $R^*(\pi)$. However, the maximal estimation error in the constrained policy space $\min_{\pi_{\text{ref}}} \max_{\pi' \in \Pi_C} \Gamma_{\hat{R}}(\pi', \pi_{\text{ref}})$ also increases, which can cause reward hacking. To balance the trade-off, the constrained policy space should be limited to a subset of policies with low estimation error. This idea is closely related to the concept of 'pessimism,' which requires the agent to focus on the policies covered by the dataset.

As shown in Alg 1, an intuitive and popular way to balance the trade-off is to use KL regularization to specify the constrained policy space (Christiano et al., 2017). However, this approach could suffer from the difficulty of RL training for finding the optimal policy on the regularized reward among the full policy class $\pi \in \Pi^{\mathcal{S},\mathcal{A}}$, resulting in significant optimization errors. This is also observed in our experiments in Section 4. Next, we introduce our approach to balance the estimation-constraint error while reducing the optimization error.

3.2 Policy Optimization on An Explicitly Constrained Policy Space

A straightforward way to reduce the optimization error is to reduce the complexity of the optimization problem. Our high-level idea is to ensure that the agent can directly search for the optimal policy in the

constrained policy class instead of searching among all possible policies. This can be achieved by explicitly choosing specific policies for the constrained policy class. In order to avoid the problem of reward over-optimization, such policies should mostly visit the state-actions covered by the dataset. Therefore, we define a special constrained policy space with an imitation policy π^i and a pessimistic coefficient p as below:

$$\Pi_C := \{\pi : \forall(s, a) \text{ s.t. } \pi^i(a|s) < p, \pi(a|s) = 0\}$$

Here, we abuse the notation of $\pi(a|s)$. If the action space is discrete, then $\pi(a|s)$ represents the probability of the policy to generate the response a at the state s. If the action space is continuous, the $\pi(a|s)$ becomes the probability density. The constrained policy class Π_C consists of policies supported on a constrained action space. Given a value of p, the constrained action space at a state $\mathcal{A}'|s$ includes the actions that satisfy $\pi^i(a|s) \geq p$. This choice tackles the problem of reward hacking in a way similar to the KL regularization approach. If the imitation policy π^i can represent the action distribution in the dataset, then the constrained policy class here explicitly restricts the agent to consider policies that only take the actions likely covered by the dataset. More importantly, the complexity of finding the optimal policy is reduced when the agent only focuses on a constrained policy space instead of the full policy space. Here, the agent only needs to search among the policies supported on a constrained action space that is much smaller than the original action space. This can make the reinforcement learning process easier and reduce the optimization loss. We use an intuitive example to show how much optimization loss can be reduced. Consider the case of the environment being a tabular MDP where mature theoretical understandings have been developed. For the popular and efficient UCBVI algorithm (Azar et al., 2017), the dependency of optimization loss on the size of action space $|A|$ is $O(\sqrt{|A|})$. In the case of learning on a clipped action space, if the clipped action space has a size of $|A|/N$, then the optimization loss is reduced by $1/\sqrt{N}$ times. In our practical implementations in Section 4, the size of the clipped action is 10^{-d} times smaller than that of the full action space, where d is the number of action space dimensions. Intuitively, learning on such a small action space can significantly reduce the optimization error, and we further empirically verified this in Section 4.

Formally, we show the general form of the PRC algorithm using the constrained policy class above in Alg 2. In the first step, the learning agent learns a behavior policy that can best imitate the behavior demonstrated in the dataset and then learns a reward model that best interprets the dataset's preference label. In the second learning step, the agent learns the policy supported on a constrained action space $\mathcal{A}'$ that maximizes the cumulative return on the learned reward model.

Algorithm 2: Preference Based Reinforcement Learning on a Constrained Action Space

Inputs **:** Environment $(\mathcal{S}, \mathcal{A}, \mathcal{P})$, Dataset D

1. Train a utility function $\hat{u}$ on D that minimize $\mathcal{L}_u(\cdot, D)$ to approximate the preference model in D.
2. Train a policy π_b from a class of policy Π_b that minimizes $\mathcal{L}_\pi(\pi, D)$ to approximate the behavior policy behind D.
3. Construct a clipped action space $\mathcal{A}'$ such that at a state s, the clipped action space is $\mathcal{A}'|s = \{a : \pi_0(a|s) \geq p\}$.
4. Find the policy π^* supported on the clipped action space that optimizes $\hat{\pi} = \arg\max_{\pi \in \Pi_{\mathcal{A}'}} \hat{R}(\pi)$
4. Output π^*

3.3 Practical Implementation of PRC

Finally, we present a practical implementation of the PRC algorithm for our experiments on continuous control problems. In this case, the action space is continuous and has multiple dimensions. Our goal is to determine the clipped action space that is easy to represent in practice. To achieve this, we train a deterministic imitation policy π^i that generates actions close to the ones in the dataset and then specifies the constrained action space centered at the actions given by the policy. We use $\pi^i(s)$ to denote the action given by the deterministic policy at state s. The action given by the deterministic policy can be thought of as an empirical estimation of the center of the distribution of the actual underlying behavior policy of the dataset. Since we have no prior knowledge about the shape of the distribution of the behavior policy, we

adopt a reasonable assumption that an action is more likely to be sampled by the behavior policy if it is closer to the center of its distribution. Then, we can approximate the space of $\mathcal{A}'$ as a box whose center is at the deterministic policy, and the radius of the action space is a hyperparameter related to the degree of pessimism.

Formally, in Alg 3, we show our practical implementation for the PRC algorithm. Lines 1-2 learn a reward model and an imitation policy, which a standard supervised learning approach can achieve. The loss functions are set in Eq. 2. Here, $d(\cdot,\cdot)$ measures the distance between two actions. In our experiments, we consider the standard continuous control problems, and the distance measure is defined as the L2 distance between the two actions on the action space. Lines 3-5 implement an RL environment with a constrained action space. For a policy π_{clip} supported on $\mathcal{S}$, $\mathcal{A}'$, if it outputs an action on the constrained action space $a' \in \mathcal{A}'$, then its actual output action in the original action space satisfies $a = f(s, a')$, and the state transition follows $\mathcal{P}(\cdot|s, f(s, a')) = \mathcal{P}'(\cdot|s, a')$, where $\mathcal{P}'$ is the state transition in Line 5. Therefore, we can search for the optimal clipped policy π^* from $\Pi^{\mathcal{S},\mathcal{A}'}$ on the MDP with state space $\mathcal{S}$, action space $\mathcal{A}'$, state transition $\mathcal{P}'$, and output the corresponding actual policy of π^*. In practice, we can solve this by applying the SAC or PPO algorithm on the MDP $\mathcal{M}' = (\mathcal{S}, \mathcal{A}', \mathcal{P}', \hat{R})$.

$$
\begin{aligned}
\mathcal{L}_D^p(R) &= -\sum_{(\tau_1,\tau_2)\in D} \log \frac{\exp(\sum_{(s,a)\in\tau_1} R(s,a))}{\exp(\sum_{(s,a)\in\tau_1} R(s,a)) + \exp(\sum_{(s,a)\in\tau_2} R(s,a))} \\
\mathcal{L}_D^i(\pi) &= \sum_{(s,a)\sim D} d(\pi(s), a)
\end{aligned}
\tag{2}
$$

Algorithm 3: PRC practical implementation

Inputs : Environment $(\mathcal{S}, \mathcal{A}, \mathcal{P})$, Dataset D
Parameters: Positive real number r
1. Train a reward function $\hat{R} \leftarrow \arg\min_R \mathcal{L}_D^p(R)$
2. Train a deterministic behavior clone policy $\pi^i \leftarrow \mathcal{L}_D^i(\pi)$.
3. Construct a constrained action space $\mathcal{A}' = \mathbb{R}^N$ where N is the dimensions of $\mathcal{A}$. Each dimension is constrained on $[-r, r]$.
4. Construct a mapping $f : \mathcal{S} \times \mathcal{A}' \to \mathcal{A}$ as $f(s, a') = \text{Proj}_{\mathcal{A}}(\pi^i(s) + a')$
5. Construct a state transition $\mathcal{P}'$ as $\mathcal{P}'(\cdot|s, a') = \mathcal{P}(\cdot|s, f(s, a'))$
6. Find the optimal policy $\hat{\pi}$ that optimizes $\hat{\pi}' \leftarrow \arg\max_{\pi\in\Pi^{\mathcal{S},\mathcal{A}'}} \hat{R}(\pi)]$
7. Output $\hat{\pi} : \hat{\pi}(f(s, a)|s) = \hat{\pi}'(a|s)$

4 Experiments

4.1 Setup

Dataset: Following previous studies (Hejna & Sadigh, 2023; Zhang et al., 2023), we construct our offline preference dataset from D4RL benchmark (Fu et al., 2020) labeled by synthetic preference following the preference model in Section 2. The details of our dataset construction can be found in the Appendix. Note that we consider the offline setting where the agent can only access a dataset of preference labels. We consider various datasets from D4RL to represent the general learning scenarios. The robot control environments we choose include 'Hopper', 'HalfCheetah', and 'Walker'. The types of trajectories we choose include 'Medium', 'Medium-Replay', and 'Medium-Expert'. To make the number of state-action in the dataset aligned with that of the D4RL benchmark, the total number of trajectory pairs with a preference label is 1×10^6.

Baseline Algorithms: Here, we consider multiple two-step learning algorithms and state-of-the-art attacks in the related offline PBRL setting as baselines, which are listed below:

Dataset	Oracle	PRC	Naive two-step	KL two-step	IPL	RM	FTB
HalfCheetah-Medium	47.3 ± 0.2	**47.0 ± 0.5**	40.1 ± 0.5	41.9 ± 0.1	42.7 ± 0.1	43.2 ± 0.1	43.5 ± 0.3
HalfCheetah-Medium-Replay	46.1 ± 0.1	**43.3 ± 0.2**	31.9 ± 0.8	32.9 ± 0.9	34.9 ± 3.1	40.1 ± 0.7	38.6 ± 1.1
HalfCheetah-Medium-Expert	92.1 ± 0.7	78.4 ± 2.9	40.9 ± 0.2	40.4 ± 0.5	41.7 ± 1.0	48.2 ± 0.8	**86.0 ± 0.5**
Hopper-Medium	76.1 ± 1.2	**71.0 ± 7.5**	67.5 ± 2.4	**73.5 ± 4.0**	**72.4 ± 7.4**	67.2 ± 0.3	**66.8 ± 5.7**
Hopper-Medium-Replay	76.7 ± 5.3	47.0 ± 19.5	49.4 ± 7.4	**53.1 ± 2.2**	39.5 ± 17.5	32.2 ± 0.4	**62.9 ± 9.2**
Hopper-Medium-Expert	113.1 ± 0.4	100.2 ± 9.5	67.3 ± 7.1	71.4 ± 10.9	76.9 ± 8.1	97.1 ± 5.2	**109 ± 1.7**
Walker2d-Medium	84.5 ± 0.3	**84.4 ± 0.8**	81 ± 0.8	79.5 ± 2.1	80.8 ± 1.6	81.9 ± 2.5	80.5 ± 0.6
Walker2d-Medium-Replay	83.1 ± 2.3	**87.9 ± 6.1**	49.8 ± 7.3	45.4 ± 0.1	49.3 ± 3.5	71.8 ± 6.4	79.3 ± 1.9
Walker2d-Medium-Expert	111.5 ± 0.4	**110.4 ± 0.6**	93.4 ± 4.9	92.1 ± 2.3	**107.5 ± 3.0**	105.7 ± 6.9	109 ± 0.3

Table 1: Comparison between the performance of different learning methods. Oracle is trained on the true reward signals instead of preference signals. FTB requires much more training time than other methods. The details can be found in the Appendix.

1. Two-step learning with KL-regularization and no regularization: We adopt the traditional two-step learning baseline with and without KL-regularization as introduced in Section 2. To make the comparison fair, the algorithms initialize their policies as the behavior clone policy.

2. Reward modeling followed by standard offline RL algorithms: Another simple yet efficient two-step learning approach for offline PBRL problems is to combine reward modeling with a standard reward-based offline RL algorithm. First, a reward model is learned from the preference dataset. Then, the reward model is used to provide scalar reward labels to the state-action in the dataset. Finally, we apply a standard offline RL algorithm IQL (Kostrikov et al., 2021) training on the dataset with scalar reward labels.

3. Inverse preference learning (IPL) (Hejna & Sadigh, 2023) and flow to better (FTB) (Zhang et al., 2023): IPL and FTB are the state-of-the-art learning algorithms for a related offline PBRL setting that requires a preference dataset and a behavior dataset. To avoid underestimating the performance of them, these algorithms have access to the full D4RL dataset during training. Note that our algorithm has no access to the extra behavior dataset.

4. Oracle: The oracle is trained with true rewards instead of preference labels. Here, we apply IQL training on the base D4RL dataset with true reward signals. Note that the information from the reward signals is strictly more than that from the preference signals in this case. The oracle should be considered as an upper bound on the performance of an offline PBRL algorithm.

To ensure a fair comparison, all the methods that require reward modeling share the same learned reward model during training. Additional detailed training setups can be found in the Appendix.

4.2 Learning Efficiency Evaluation

Here, we compare the efficiency of PRC against other baseline algorithms on different datasets. To straightforwardly compare the performance of different learning algorithms, we show the standard D4RL score (normalized performance) of the policy learned by a method during training.

The results in Table 1 show that the PRC algorithm has high learning efficiency. It is generally more efficient than most baselines and sometimes even competes with the oracle. Our algorithm is a more efficient two-step learning method as it performs much better than other two-step learning baselines initialized from the behavior clone policy. This observation indicates that setting the initial policy as the behavior clone policy is not enough to achieve high learning efficiency, and training on the constrained action space is the key to high learning efficiency for PRC. Even with additional access to the behaviors in the D4RL dataset, PRC algorithm outperforms the IPL algorithm. Although PRC and FTB have similar performance, we notice that FTB uses the diffusion technique (Yang et al., 2023) during training, which is very computationally expensive. In our testing, the training time required by the FTB is about 7 times as long as that required by PRC. The details can be found in the Appendix. In conclusion, the PRC algorithm has high learning efficiency while being computationally efficient.

Next, we empirically analyze why PRC should be an efficient learning algorithm from the aspect of pessimistic learning and reduced reinforcement learning complexity.

4.3 Pessimism Effectiveness

Here, we empirically verify that by training on a constrained action space, the PRC algorithm achieves effective pessimistic learning. In Figure 1, we show some representative examples. For the PRC algorithm, during training, the trend of the learned policies' performance measured on the learned reward model is aligned with that measured on the true reward. This suggests that the pessimistic learning in the PRC algorithm is effective: it mainly considers the policies that are supported by the dataset so that the agent can evaluate their relative performance accurately. In contrast, for naive two-step learning with no regularization and KL-regularized two-step learning, we find multiple cases where the trends can even be opposite when evaluated on the reward model and on the true reward. These results suggest pessimistic learning is not efficient enough in the baseline methods compared to PRC.

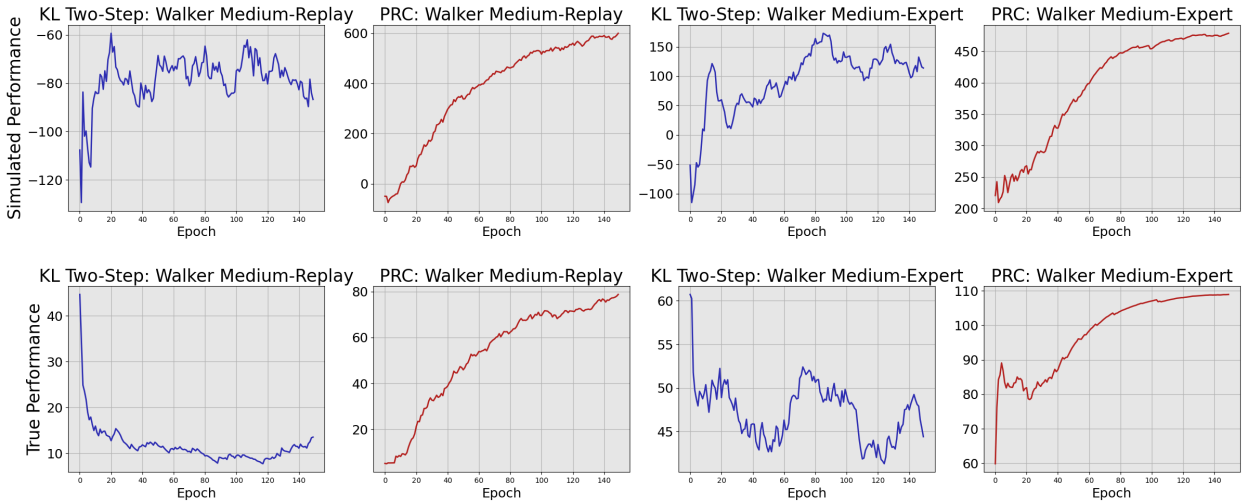

Figure 1: Comparison between the trend of the performance of the learned policies on the learned (simulated performance) and true reward models (true performance) during training. An algorithm is not pessimistic enough if the two trends are not aligned.

4.4 Reinforcement Learning Efficiency on Constrained Action Space

Here, we empirically verify that reinforcement learning is much easier on the constrained action space compared to learning on the full action space. We observe that in most cases, the simulated performance of the learned policies in the PRC method that optimizes over a constrained action space is much higher than in the naive two-step learning methods that optimize over the full action space. In Figure 2, we show some representative examples. It is clear that while the baseline method struggles with low-performing policies, the PRC method learns policies of much higher performance. Note that the simulated performances of the policies are measured over the same reward model. Considering that the optimal policy supported on a constrained action space is no higher than that supported on the full action space, the empirical results show that the optimization error of reinforcement learning on a constrained action space is much less than that on the full action space. Therefore, the empirical observation verifies our statement that the PRC method has less optimization error during reinforcement learning.

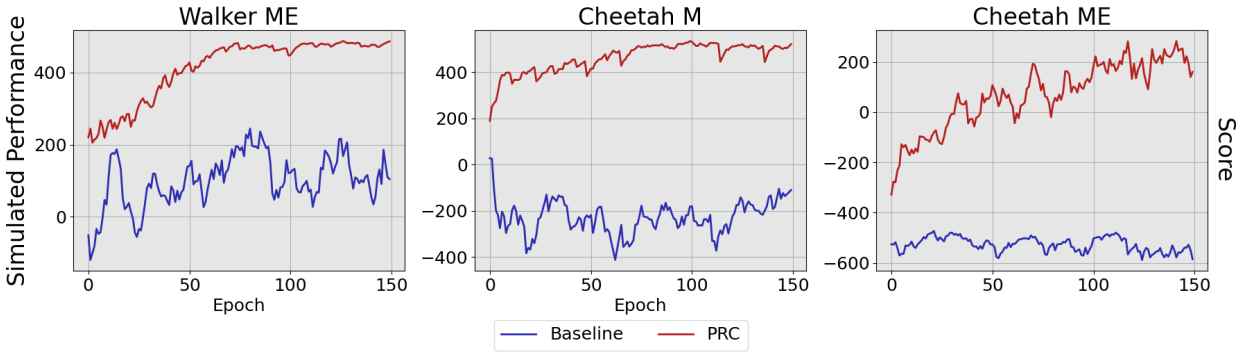

Figure 2: Comparison between the performance of the learned policies on the learned reward models during training. The reinforcement learning complexity is less in a setting if the simulated performance is high.

4.5 Reinforcement Learning Complexity in PBRL

Here, we empirically show that training on a reward model learned from preference signals is harder than that from true reward signals. Given the same D4RL dataset, we label the dataset with true reward signals and synthetic preference labels, and then train two reward models based on the two kinds of signals. Finally, we train an efficient RL algorithm on the two reward models and compare the learning efficiency in the two cases. The results in Figure 3 show that when learning on the reward model trained from true rewards, the RL agent can quickly learn some policies that have high performance on the reward model. In comparison, it can take much longer for the agent to find a good policy on the reward model from preference signals, and sometimes, the agent may not be able to find a decent policy after a lot of training epochs. This indicates that it is harder to learn from a reward model that is trained on preference signals instead of true reward signals.

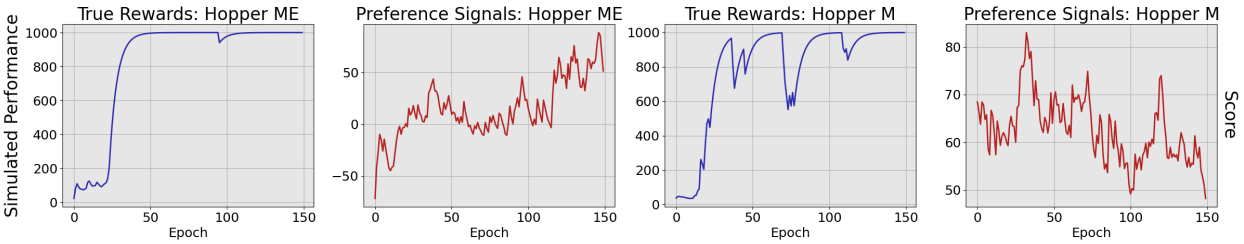

Figure 3: For each dataset, a pair of reward models are trained on the true rewards and preference signals. The same RL algorithm is applied to learn on both reward models. The learning difficulty on a reward model is less if the PPO algorithm can learn better policies according to the reward model.

5 Related Work

Offline Reinforcement learning: There have been many studies on the standard offline reinforcement learning setting with reward feedback (Levine et al., 2020; Cheng et al., 2022; Yu et al., 2021; Fujimoto & Gu, 2021; Bhardwaj et al., 2023; Lyu et al., 2024), in contrast to preference feedback considered in our setting. The key idea behind offline reinforcement learning is pessimism, which encourages the learning agent to focus on the policies or actions supported by the dataset (Cheng et al., 2022). However, the gap between learning from reward feedback and preference feedback is currently unclear.

Online PBRL: Preference-based reinforcement learning is popular in the online learning scenario (Christiano et al., 2017; Ye et al., 2024; Ibarz et al., 2018; Lee et al., 2021b). Various types of preference models and labels have been considered in the tabular case (Fürnkranz et al., 2012; Wirth et al., 2017). Similar to the works that consider the general case, in our work, we consider the preference models to be Bradley-Terry models and output binary preference labels (Ibarz et al., 2018; Lee et al., 2021a). The two-phase learning approach mentioned in Section 1 is prevalent in the online PBRL setting. Note that directly applying online methods to the offline scenario is also likely to be inefficient even when possible (Van Hasselt et al., 2018), as the online learning approaches do not need to be pessimistic.

Offline PBRL: Offline preference-based reinforcement learning has attracted much attention recently. Specifically, we find three types of offline PBRL settings attract most attention. 1. Zhan et al. (2023); Zhu et al. (2023) consider the offline learning setting where the training data only consists of preference labels. This setting is similar to the standard reward-based offline RL setting, where the training data only consists of reward labels. This work focuses on this setting. Zhan et al. (2023) proposes an optimization problem following the two-phase learning framework and theoretically proves that the solution to the problem is a near-optimal policy with respect to the training dataset. However, how to solve the optimization problem in practice remains unknown. Concurrently, Zhu et al. (2023) formulates a similar optimization problem under the linear assumption of the environment and proposes a way to solve the problem. Still, the results cannot be extended to the general case without the linear assumption. 2. Sadigh et al. (2017); Shin et al. (2021; 2023) study the PBRL problem in the active learning setting where the agent has access to an offline behavior dataset and can choose a subset of the dataset to be labeled with preference feedback. 3. An et al. (2023); Kim et al. (2023); Hejna & Sadigh (2023); Zhang et al. (2023); Tu et al. (2024); Gao et al. (2024); Kang & Oh (2025); Choi et al. (2024); Liu et al. (2024) consider a different offline learning setting where the agent can access both a dataset of preference labels and an additional trajectory dataset that only contains trajectories. The fundamental difference between this setting and the active learning setting is that, here, the preference labels are given to the trajectories before training. Note that reinforcement learning from human feedback (RLHF) for natural language processing (NLP) tasks is a special instance of PBRL where humans provide the preference labels and is often studied in the standard offline setting (Ouyang et al., 2022; Bai et al., 2022; Rafailov et al., 2023) which is the same as the setting considered in this work.

6 Conclusion and Limitation

In this work, we propose a novel two-step learning algorithm called PRC for the offline PBRL setting. We empirically verify that the PRC has high learning efficiency and provide evidence for why it is a more efficient two-step learning algorithm than others. Our framework is limited to the typical offline learning setting where the agent only has access to an offline preference dataset. Our experimental evaluation is limited to continuous control problems, the standard benchmark in RL studies.

References

Riad Akrour, Marc Schoenauer, and Michèle Sebag. April: Active preference learning-based reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II 23*, pp. 116–131. Springer, 2012.

Gaon An, Junhyeok Lee, Xingdong Zuo, Norio Kosaka, Kyung-Min Kim, and Hyun Oh Song. Direct preference-based policy optimization without reward modeling. *Advances in Neural Information Processing Systems*, 36:70247–70266, 2023.

Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *International conference on machine learning*, pp. 263–272. PMLR, 2017.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Mohak Bhardwaj, Tengyang Xie, Byron Boots, Nan Jiang, and Ching-An Cheng. Adversarial model for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:1245–1269, 2023.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal. Adversarially trained actor critic for offline reinforcement learning. In *International Conference on Machine Learning*, pp. 3852–3878. PMLR, 2022.

Heewoong Choi, Sangwon Jung, Hongjoon Ahn, and Taesup Moon. Listwise reward estimation for offline preference-based reinforcement learning. *arXiv preprint arXiv:2408.04190*, 2024.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Johannes Fürnkranz, Eyke Hüllermeier, Weiwei Cheng, and Sang-Hyeun Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine learning*, 89:123–156, 2012.

Chen-Xiao Gao, Shengjun Fang, Chenjun Xiao, Yang Yu, and Zongzhang Zhang. Hindsight preference learning for offline preference-based reinforcement learning. *arXiv preprint arXiv:2407.04451*, 2024.

Ammar Haydari and Yasin Yılmaz. Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):11–32, 2020.

Joey Hejna and Dorsa Sadigh. Inverse preference learning: Preference-based rl without a reward function. *arXiv preprint arXiv:2305.15363*, 2023.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31, 2018.

Hyungkyu Kang and Min-hwan Oh. Adversarial policy optimization for offline preference-based reinforcement learning. *arXiv preprint arXiv:2503.05306*, 2025.

Changyeon Kim, Jongjin Park, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. Preference transformer: Modeling human preferences using transformers for rl. *arXiv preprint arXiv:2303.00957*, 2023.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021a.

Kimin Lee, Laura Smith, Anca Dragan, and Pieter Abbeel. B-pref: Benchmarking preference-based reinforcement learning. *arXiv preprint arXiv:2111.03026*, 2021b.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Xiao-Yin Liu, Guotao Li, Xiao-Hu Zhou, and Zeng-Guang Hou. Lease: Offline preference-based reinforcement learning with high sample efficiency. *arXiv preprint arXiv:2412.21001*, 2024.

Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys & Tutorials*, 21(4):3133–3174, 2019.

Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. Mildly conservative q-learning for offline reinforcement learning, 2024. URL `https://arxiv.org/abs/2206.04745`.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. *Active preference-based learning of reward functions*. 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Daniel Shin, Daniel S Brown, and Anca D Dragan. Offline preference-based apprenticeship learning. *arXiv preprint arXiv:2107.09251*, 2021.

Daniel Shin, Anca D Dragan, and Daniel S Brown. Benchmarks and algorithms for offline preference-based reward learning. *arXiv preprint arXiv:2301.01392*, 2023.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Songjun Tu, Jingbo Sun, Qichao Zhang, Yaocheng Zhang, Jia Liu, Ke Chen, and Dongbin Zhao. In-dataset trajectory return regularization for offline preference-based reinforcement learning. *arXiv preprint arXiv:2412.09104*, 2024.

Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Christian Wirth and Johannes Fürnkranz. Preference-based reinforcement learning: A preliminary survey. In *Proceedings of the ECML/PKDD-13 Workshop on Reinforcement Learning from Generalized Feedback: Beyond Numeric Rewards*. Citeseer, 2013.

Christian Wirth, Riad Akrour, Gerhard Neumann, Johannes Fürnkranz, et al. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46, 2017.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

Chenlu Ye, Wei Xiong, Yuheng Zhang, Hanze Dong, Nan Jiang, and Tong Zhang. Online iterative reinforcement learning from human feedback with general preference model. *Advances in Neural Information Processing Systems*, 37:81773–81807, 2024.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Wenhao Zhan, Masatoshi Uehara, Nathan Kallus, Jason D Lee, and Wen Sun. Provable offline reinforcement learning with human feedback. *arXiv preprint arXiv:2305.14816*, 2023.

Zhilong Zhang, Yihao Sun, Junyin Ye, Tian-Shuo Liu, Jiaji Zhang, and Yang Yu. Flow to better: Offline preference-based reinforcement learning via preferred trajectory generation. In *The Twelfth International Conference on Learning Representations*, 2023.

Banghua Zhu, Jiantao Jiao, and Michael I Jordan. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. *arXiv preprint arXiv:2301.11270*, 2023.

A Proof for Proposition 3.1

Proof. For any policy $\pi \in \Pi_C$ and any reference policy $\pi_{\text{ref}} \in \Pi^{\mathcal{S},\mathcal{A}}$, the performance gap between π and the learned policy $\hat{\pi}$ satisfies:

$$\begin{aligned}
R^*(\pi) - R^*(\hat{\pi}) &= \big(R^*(\pi) - R^*(\pi_{\text{ref}})\big) - \big(R^*(\hat{\pi}) - R^*(\pi_{\text{ref}})\big) \\
&= \{\big(R^*(\pi) - R^*(\pi_{\text{ref}})\big) - \big(\hat{R}(\pi) - \hat{R}(\pi_{\text{ref}})\big)\} + \big(\hat{R}(\pi) - \hat{R}(\pi_{\text{ref}})\big) - \\
&\quad \{\big(R^*(\hat{\pi}) - R^*(\pi_{\text{ref}})\big) - \big(\hat{R}(\hat{\pi}) - \hat{R}(\pi_{\text{ref}})\big)\} - \big(\hat{R}(\hat{\pi}) - \hat{R}(\pi_{\text{ref}})\big) \\
&\leq \Gamma_{\hat{R}}(\pi, \pi_{\text{ref}}) + \Gamma_{\hat{R}}(\hat{\pi}, \pi_{\text{ref}}) + \hat{R}(\pi) - \hat{R}(\hat{\pi}) \\
&\leq \Gamma_{\hat{R}}(\pi, \pi_{\text{ref}}) + \max_{\pi' \in \Pi_C} \Gamma_{\hat{R}}(\pi', \pi_{\text{ref}}) + \epsilon_{\text{opt}}
\end{aligned}$$

The first inequality follows the definition of the estimation error Γ. Since the bound on performance gap holds for any policy $\pi \in \Pi_C$ given any reference policy $\pi_{\text{ref}} \in \Pi^{\mathcal{S},\mathcal{A}}$, the performance of $\hat{\pi}$ can be lower bound by

$$R^*(\hat{\pi}) \geq \max_{\pi \in \Pi_C, \pi_{\text{ref}} \in \Pi^{\mathcal{S},\mathcal{A}}} \big(R^*(\pi) - \Gamma_{\hat{R}}(\pi, \pi_{\text{ref}}) - \max_{\pi' \in \Pi_C} \Gamma_{\hat{R}}(\pi', \pi_{\text{ref}})\big) - \epsilon_{\text{opt}}$$

□

B Additional Details for Experimental Setups

To learn a reward model, we follow a standard supervised learning framework. Specifically, we use a multilayer perceptron (MLP) structure for the neural network to approximate the utility model. The neural network consists of 3 hidden layers, and each has 64 neurons. We use a tanh function as the output activation so that the output is bound between $[-1, 1]$. To train a deterministic behavior clone policy, the neural network we use to represent the clone policy has the same structure as that for the utility model. To train a stochastic behavior clone policy, the network outputs the mean and standard deviation of a Gaussian distribution separately. The network is an MLP with 3 hidden layers each with 64 neurons. The last layer is split into two for the two outputs. For the mean output, we use a linear function for the last layer. For the standard deviation, we use a linear function and an exp activation for the last layer. In the reinforcement learning step, we use either the SAC algorithm or the PPO algorithm, depending on the dataset. The structure of the neural network for the actor is identical to that of the network used for training a stochastic behavior clone policy. The structure of the neural network for the critic is identical to that of the network used for training the reward model.

C Dataset Construction

The construction of our preference dataset is based on the reward-based datasets from the standard D4RL benchmark. We generate synthetic preference labels following the preference model introduced in Section 2 to ensure that the dataset construction is consistent with the problem setting. We formally show the process of data construction as follows.

1. Randomly sample pairs of trajectory clips from the D4RL dataset. Following previous studies (Christiano et al., 2017), the length of the clip is set to be 20 steps.
2. For each pair of trajectory clips, compute the probability of a trajectory to be preferred based on the reward signals. To ensure consistency between different datasets, the reward signals are regularized to be bound in $[-1, 1]$.
3. For each pair of trajectory clips, randomly generate a preference label through a Bernoulli trial with the probability computed above.
4. Return the preference dataset consisting of the trajectory clip pairs and the corresponding preference labels.

D PRC and FTB Runtime Comparison

For the computational resource, we use a single NVIDIA A100-PCI GPU with 40 GB RAM and an Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz with 64 GB RAM. The architecture for all neural networks used in the FTB and PRC algorithms is a multilayer perceptron with 3 hidden layers, and each has 64 neurons. On the datasets we construct, on average, the training time until convergence for the PRC algorithm is 5.14 ± 1.50 hours, while for the FTB algorithm it is 34.64 ± 2.96 hours.