POPO: Pessimistic Offline Policy Optimization

Qiang He[1,2]**, Xinwen Hou**[1]**, Yu Liu**[1]
[1]Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences

Abstract

Offline reinforcement learning (RL), also known as batch RL, aims to optimize policy from a large pre-recorded dataset without interaction with the environment. This setting offers the promise of utilizing diverse, pre-collected datasets to obtain policies without costly, risky, active exploration. However, commonly used off-policy algorithms based on Q-learning or actor-critic perform poorly when learning from a static dataset. In this work, we study why off-policy RL methods fail to learn in offline setting from the value function view, and we propose a novel offline RL algorithm that we call Pessimistic Offline Policy Optimization (POPO), which learns a pessimistic value function to get a strong policy. We find that POPO performs surprisingly well and scales to tasks with high-dimensional state and action space, comparing or outperforming several state-of-the-art offline RL algorithms on benchmark tasks.

1 Introduction

One of the main driving factors for the success of the mainstream machine learning paradigm in open-world perception environments (such as computer vision, natural language processing) is the ability of high-capacity function approximators (such as deep neural networks) to learn inductive models from large amounts of data [1] [2]. Combined with deep learning, reinforcement learning (RL) has proven its great potential in a wide range of fields such as playing Atari games [3], playing chess, Go and shoji [4], beating human players in StarCraft [5] etc. However, it turns out that reinforcement learning is difficult to extend from physical simulators to the unstructured physical real world because most RL algorithms need to actively collect data due to the nature of sequential decision making, which is very different from the typical supervised learning setting. In this paper, we study how to utilize RL to solve sequential decision-making problems from a fixed data set, i.e., offline RL, a.k.a. batch RL, which is opposite to the research paradigm of active, interactive learning with the environment. In the physical world, we can usually obtain static data from historical experiences more easily than dynamics data, such as scheduling a region's power supply system. There are problems with model deviation for such a scenario, and too expensive for manufacturing a simulator. Therefore, learning from static datasets is a crucial requirement for generalizing RL to a system where the data collection procedure is time-consuming, risky, or expensive. In principle, if assumptions about the quality of the behavior policies that produced the data can be satisfied, then we can use imitation learning (IL) [6] to get a strong policy. However, many imitation learning algorithms are known to fail in the presence of suboptimal trajectories or to require further interaction with the environment in which the data is generated from [7] [8] [9]. Many off-policy RL methods have proven their excellent sample-efficiency in complex control tasks or simulation environments recently [10] [11] [12]. Generally speaking, off-policy reinforcement learning is considered to be able to leverage any data to learn skills. However, in practice, these methods still fail when facing arbitrary off-policy data without any opportunity to interact with its environment. Even the off-policy RL method would fail given high-quality expert data produced by the same algorithm. This phenomenon goes against our intuition about off-policy RL because if shown expert data, then exploration, RL's intractable problem, no longer exists. The sensitivity of existing RL algorithms to data limits the broad application of RL. We aim to develop a completely offline RL algorithm, which can learn from large, arbitrarily static datasets.

Our contributions are summarized as follows. Firstly, we show that a critical challenge arises when applying value-based algorithms to completely offline data that the estimation gap of the value function. When we evaluate the value function, the inability to interact with the environment makes it unable to eliminate the estimation gap through the Bellman equation. This gap can lead to the value function's catastrophic estimation issue for data that the actions do not appear in the data set. Secondly, We propose a novel offline policy optimization method, namely Pessimistic Offline Policy Optimization (POPO), where the policy utilizes a pessimistic distributional value function to approximate the true value, thus learning a strong policy. Finally, we demonstrate the effectiveness of POPO by comparing it with SOTA offline RL methods on the MuJoCo locomotion benchmarks [13]. Furthermore, we conduct fine-grained experiments to verify POPO reflects the principles that informed its development.

2 Related Work

Imitation learning [6] methods study how to learn a policy by mimicking expert experience demonstrations. IL has been combined with RL, either by learning from demonstrations [14] [9] [15], or using deep RL extensions [7] [16], or using variants policy gradient methods [17] [8]. Although this family of methods has proven its efficiency, it is still insufficient in the face of fully offline datasets. They either require interaction with the environment or need high-quality data. These requirements are unable to meet under offline setting, making the use of imitation learning from offline data impractical [18]. How to deal with the impact of noise is also an urgent area in imitation learning [19] [20]. Gao et al. [21] introduced an algorithm that learns from imperfect data, but it is not suitable for continuous control tasks. We borrow from the idea of imitation learning and introduce a generative model into POPO, which gives our model the potential of rapid learning.

For unlimited data, some offline RL methods have proven their convergence, such as using non-parametric function approximation methods [22] and kernel methods [23]. Fitted Q-iteration, using function approximation methods, such as decision trees [24], neural networks [25], cannot guarantee convergence in the offline setting. Recently, many offline RL algorithm combined with deep learning have received significant attention [18] [26] [27] [28] [29][30] [31] [32] [33]. Offline RL suffers from the problem of the distribution shift, a.k.a. out-of-distribution (OOD) actions. Specifically, the target of the Bellman backup operator utilizes actions sampled from the learned policy in the policy evaluation and policy improvement process, which may not exist in the datasets. In the sense of batch-constrained, the BCQ algorithm [18] can ensure that it converges to the optimal policy under the given consistent datasets. Bootstrapping error accumulation reduction (BEAR) algorithm [26] uses maximum mean discrepancy (MMD) [34] to constrain the support of learned policy close to the behavior policy. Safe policy improvement with baseline bootstrapping (SPIBB) [27], similar to BEAR, constrains the support of learned policy w.r.t. behavior policy. Behavior regularized actor-critic (BRAC) [29] is an algorithmic framework that generalizes existing approaches to solve the offline RL problem by regularizing the behavior policy. AlgaeDICE [30] is an algorithm for policy gradient from arbitrary experience via DICE [35], which is based on a linear programming characterization of the Q-function. Critic Regularized Regression (CRR) [31] algorithm can be seen as a form of filtered behavior cloning where data is selected based on the policy's value function. Conservative Q-learning (CQL) [32] aims to learn a conservative Q-function such that the expected value of a policy under this Q-function lower-bounds its true value. Random ensemble mixture (REM) [28] uses random convex combinations of value functions to learn a static data set. Besides, REM proves that distributional RL can learn a better policy than the conventional form in offline setting. But there is a controversy that the success comes from the massive amount of data resulting in actions induced by concurrent policy always in the datasets [36]. However, their work did not consider distributional value functions to control the attitude towards the OOD actions, thus improving policy. We recommend that readers check IQN algorithm [37]. These methods focus on how to deal with OOD actions.

3 Background

The optimization goal of RL is to get an optimal policy by interacting with its environment in discrete timesteps. We formalize the standard RL paradigm as a Markov Decision Process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \rho_0, \gamma)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, transition probability function $p(s', r|s, a)$, initial state distribution ρ_0, and discount factor $\gamma \in [0, 1)$. At each time step t, the agent receives a state $s \in \mathcal{S}$ and selects an action $a \in \mathcal{A}$ with respect to its policy $\pi : \mathcal{S} \to \mathcal{A}$, then receiving a reward signal r and a new state s' from its environment. A four-element tuple (s, a, r, s') is named transition. The optimization problem in RL is to maximize the cumulative discounted reward, defined as $R_t = \sum_{i=t} \gamma^{i-t} r(s_i, a_i)$ with γ determining the priority of recent rewards. Here the return depends on the actions, thus on the policy π, deterministic or stochastic. Typically, the action-value function, a.k.a. Q-function, critic, is defined as $Q(s, a) = \mathbb{E}_{p,\pi}[R_t|s_0 = s, a_0 = a]$ which measures the quality of an action a given a state s. State-value function, a.k.a value function, V-function, is defined as $V(s) = \mathbb{E}_{p,\pi}[R_t|s_0 = s]$ measuring the quality of an individual state s. Both Q-function and V-function can be applied to evaluate the policy and further guide the agent to learn a higher quality value, i.e., a better policy. For a given policy π, the Q-function can be estimated recursively by Bellman backup operator [38]:

$$Q^\pi(s, a) = r + \gamma \mathbb{E}_{s',a'}[Q^\pi(s', a')], \tag{1}$$

where $a' \sim \pi(s')$. The Bellman operator is gamma-contraction when $\gamma \in [0, 1)$ with a unique fixed point $Q^\pi(s, a)$. We can recover the optimal policy through the corresponding optimal value function $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ in discrete action space. When we apply RL to large state space or continuous state space, the value function can be approximated by neural networks, which is called Deep Q-networks [3]. The Q-function is updated by $r + \gamma Q(s', \pi(s'); \theta')$, where $\pi(s') = \arg\max_{a'} Q(s', a'; \theta')$ and θ' is a delayed copy of θ. Generally, we sample mini-batch transitions from replay buffer $\mathcal{B}$ [39] and feed the data into the deep Q-networks. In offline policy optimization setting, we consider the buffer $\mathcal{B}$ static and no further data can be added into itself and call it datasets $\mathcal{D}$. A vitally significant improvement of DQN is soft update. When updating the network, we freeze a target network $Q(\cdots; \theta')$ to stabilize the learning processing further. The frozen network are updated by $\theta' = \eta\theta + (1 - \eta)\theta'$ every specific time steps t, where the η is a small scalar [3]. In continuous action space, the $\arg\max$ operator in Equation 1 is intractable. Thus Sutton et al. [40] introduced policy gradient method. Combined with aforementioned value-based method, actor-critic method was introduced, which is widely used in the field of deep RL. When we train an actor-critic agent, the action selection is performed through a policy network $\pi(\cdot; \phi)$, a.k.a. actor, and updated w.r.t. a value network [38]. Silver et al. [41] proposed deterministic policy gradient theorem to optimize policy:

$$\phi \leftarrow \arg\max_{\phi} \mathbb{E}_s \left[Q^\pi(s, \pi(s; \phi); \theta)\right], \tag{2}$$

which corresponds to optimizing the Q-function $Q^\pi(\cdot, \cdot; \theta)$ by the chain rule. When combined with tricks in DQN [3], this algorithm is referred to as deep deterministic policy gradients (DDPG) [10].

4 Diagnosing Value Function Estimation

Offline reinforcement learning suffers from OOD actions. Specifically, the target of the Bellman backup operator utilizes actions generated by the learned policy in the policy evaluation and policy improvement process. However, the generated actions may not exist in the dataset. Thus, we cannot eliminate the error through the Bellman update. Both the value-based method and policy gradient methods would fail for this reason. Hasselt et al. [42] observed that overestimation occurs in the DQN algorithm. We argue that the analogous phenomenon also occurs in offline scenario but for the different underlying mechanism. These two phenomena are coupled with each other, making the value function more difficult to learn than online setting. In the standard reinforcement learning setting, these errors due to erroneous estimation could be eliminated through the agent's exploration to obtain a true action value and then updated by the Bellman backup operator. But for offline setting, this error cannot be eliminated due to the inability to interact with the environment. Furthermore, due to the backup nature of the Bellman operator, the error would gradually accumulate, which would eventually cause the value function error to become larger, leading to the failure of policy learning. Some algorithms train policies through optimizing the Q-value indirectly. And some actor-critic style methods optimize the policies directly but are assisted by the value function. Therefore, out-of-distribution actions harm these RL algorithms' performance in offline setting. We call the aforementioned error estimation gap.

Definition 1. We define estimation gap for policy π in state s as $\delta_{\text{MDP}}(s) = V^{\pi}(s) - V^{\pi}_{\mathcal{D}}(s)$ where $V^{\pi}(s)$ is true value and $V^{\pi}_{\mathcal{D}}(s)$ is estimated on dataset $\mathcal{D}$.

Theorem 1. Given any policy π and state s, the estimation gap $\delta_{\text{MDP}}(s)$ satisfies the following Bellman-like equation:

$$\delta_{\text{MDP}}(s) = \sum_{a} \pi(a|s) \sum_{s',r} \left[p(s',r|s,a) - p_{\mathcal{D}}(s',r|s,a)\right] \big(r + \gamma V^{\pi}_{\mathcal{D}}(s')\big) + \gamma \sum_{a} \pi(a|s) \sum_{s',r} p(s',r|s,a)\delta_{\text{MDP}}(s')$$

Proof. Through the definition of the V function, it can be proved by expanding this equation. □

Theorem 1 shows that the estimation gap is a divergence function w.r.t. the transition distributions, which means if the policy carefully chooses actions, the gap can be minimized by visiting regions where the transition probability is similar.

Remark 1. For any reward function, $\delta_{\text{MDP}} = 0$ if and only if $p(s',r|s,a) = p_{\mathcal{D}}(s',r|s,a)$.

4.1 Does this phenomenon occurs in practical?

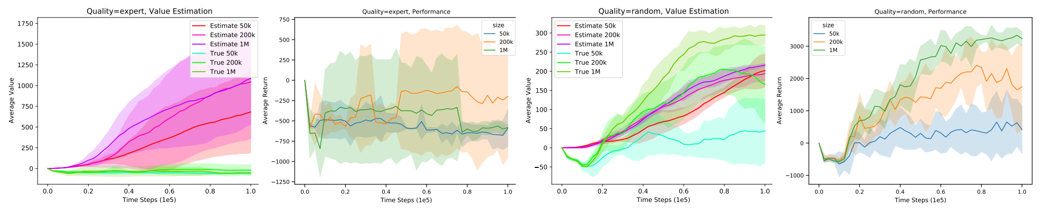

Figure 1: The relationship between data quality, quantity, and corresponding average return. We train TD3 algorithm on MuJoCo halfcheetah-v2 environment over five random seeds. 'Estimate 50k' means the curve shows the agent's value estimation on the size=50k dataset. To maximally control the influence of random factors, we control the random seed of each experiment. The shaded area represents a standard deviation.

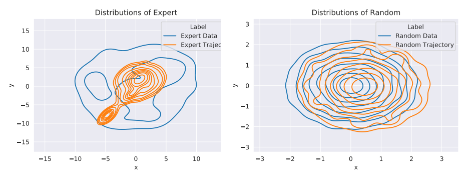

Figure 2: Visualization of data generated by the halfcheetah-v2 environment. Left: expert data, visiting the different area. Right: random data, visiting a similar area. The state space is 17-dim, and the action space is 6-dim. We concat a trajectory as a vector, and we reduce the trajectories with a dimension of 23k to a two-dimensional plane.

We utilize datasets [43] of different qualities and sizes to verify our analysis. We train TD3, a SOTA algorithm on continuous control tasks, on halfcheetah-v2 environment in offline setting. We show results in Figure 1. Surprisingly, training on random data gives us a better average return than expert data. Observing its value function, we find that for expert data, as the data set capacity increases, the estimated value function deviates more and more from the true value, which verifies there does exist an estimation gap. The erroneous estimation of the value function further leads to the failure of policy learning. Why can random data learn better? The theory above inspires us that if the policy chooses actions carefully, it can eliminate the estimation gap by visiting regions with similar transition probability, suggesting that the phenomenon may be due to the large difference in the visited state. Thus, we visualizing [44] the distributions of the datasets and trajectories in Figure 2. We collect five trajectories every 5,000 training steps. Expert/random trajectory means we train TD3 on expert/random dataset in offline setting. We find that the TD3 agent does visit a similar area on random data. Still, for expert data, the agent visits different area from expert data even though they have the same origin, which is consistent with our theory.

5 Pessimistic Offline Policy Optimization

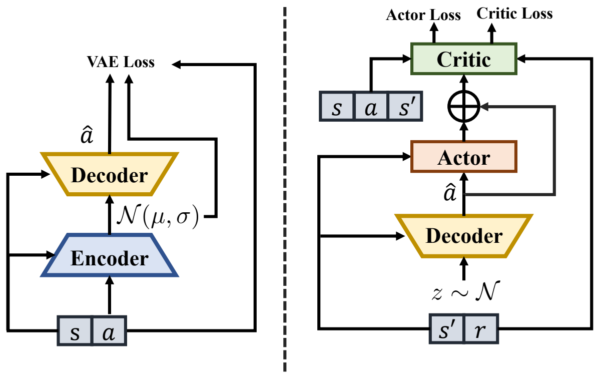

Figure 3: The architecture of POPO. Left: Conditional VAE Optimization. Right: Policy Optimization.

Our insight is if the agent could maintain a pessimistic attitude towards the actions out of the support of behavior policy when learning the value function, then we can suppress the estimation gap of the value function outside the data set so that the algorithm can obtain a more melancholic value function to learn a strong policy through an actor-critic style algorithm. To capture more information on the value function, we utilize distributional value function [45] [37], which has proved its superiority in the online learning setting.

5.1 Pessimistic Value Function

Now we introduce pessimistic value function estimation from distributional RL view. The distributional Bellman optimality operator is defined by:

$$\mathcal{T} Z^\pi(s, a) := r + \gamma Z^\pi(s', \arg\max_{a' \in \mathcal{A}} \mathbb{E} Z(s', a')), \tag{3}$$

where $s' \sim p(\cdot|s, a)$ and random return $Z^\pi(s, a) := \sum_{t=0}^{\infty} r(s_t, a_t)$, $s_0 = s, a_0 = a, s_t \sim p(\cdot|s_{t-1}, a_{t-1}), a_t \sim \pi(\cdot|s_t)$, and $X = Y$ denotes that random variable X and Y have equal distribution. Let $F_Z^{-1}(\tau)$ denotes the quantile function at $\tau \in [0, 1]$ for random variable Z. We write $Z_\tau := F_Z^{-1}(\tau)$ for simplicity. Similar to [37], we model the state-action quantile function as a mapping from state-action to samples from certain distribution, such as $\tau \sim U(0, 1)$ to $Z_\tau(s, a)$. Let $\beta : [0, 1] \to [0, 1]$ be a distortion risk measure. Then the distorted expectation of random variable $Z(s, a)$ induced by β is:

$$Q_\beta(s, a; \theta) := \mathbb{E}_{\tau \sim U(0,1)}[Z_{\beta(\tau)}(s, a; \theta)]. \tag{4}$$

We also call Z_β critic. By choosing different β, we can obtain various distorted expectation, i.e., different attitude towards the estimation value. To avoid the abuse of symbols, τ in the following marks τ acted by β. For the critic loss function, given two samples, $\tau, \tau' \sim U(0, 1)$, the temporal difference error at time step t is:

$$\Delta_t^{\tau,\tau'} = r_t + \gamma Z_{\tau'}(s_{t+1}, \pi_\beta(s_{t+1})) - Z_\tau(s_t, a_t). \tag{5}$$

Then the critic loss function of POPO is given by:

$$\mathcal{L}(s_t, a_t, r_t, s_{t+1}) = \frac{1}{N'} \sum_{i=1}^{N} \sum_{j=1}^{N'} \rho_{\tau_i}^{\kappa}(\Delta_t^{\tau_i, \tau_j'}), \tag{6}$$

where

$$\rho_\tau^\kappa(x) = |\tau - \mathbb{I}\{x < 0\}| \frac{\mathcal{L}_\kappa(x)}{\kappa}, \quad \text{with} \tag{7}$$

$$\mathcal{L}_\kappa(x) = \begin{cases} \frac{1}{2}x^2, & \text{if} |x| \leq \kappa \\ \kappa(|x| - \frac{1}{2}\kappa), & \text{otherwise} \end{cases}$$

in which N and N' is the number of i.i.d. samples τ_i, τ_j' draw from $U(0,1)$ respectively. Thus, given Z function, we can recover the $Q_\beta(s,a)$ from the Equation 4, further guides the learning process of policy.

5.2 Distribution-Constrained Optimization

To tackle OOD actions, we introduce a generative model, specifically, conditional Variational Auto-Encoder (VAE) $G(\cdot;\omega)$, consists of Encoder $E(\cdot|\cdot;\omega_1)$ and Decoder $D(\cdot|\cdot;\omega_2)$. Furthermore, VAE could constrain the distance between the actions sampled from the learned policy and that provided by the datasets. VAE reconstructs action on condition state s. We call the action produced by the VAE the central action $\hat{a}$. Thus, the loss function of VAE is:

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{s,a}\big[(a - \hat{a})^2 + \frac{1}{2}D_{\text{KL}}(\mathcal{N}(\mu,\Sigma) \| \mathcal{N}(0,I)\big]. \tag{8}$$

where $s, a \sim \mathcal{D}$ and $\hat{a} = D(z|s;\omega_2)$. To generate actions a' w.r.t. state s', firstly we copy the action n times and send it to VAE in order to incorporate with policy improvement. Then we feed the actor network with central action $\hat{a_i'} = D(z_i|s';\omega_2)$ and state s', then the actor network $\pi(\cdots;\phi)$ outputs a new action $\bar{a_i'}$. Combining $\hat{a_i'}$ and $\bar{a_i'}$ with residual style with coefficient ξ, we get the selected action $\tilde{a_i'}$. We choose action of n output with highest value as the final output:

$$a'_{\text{new}} = \arg\max_{a_i} Q_\beta(s', \tilde{a_i'};\theta), \tag{9}$$

where $\{\tilde{a_i'} = (\pi \circ D)(z_i|s')\}_{i=1}^n$. We call this action generation method the residual action generation. The whole model structure is shown in Figure 3. We use the DPG method (Equation 2) to train actor network π. The benefits of residual action generation are apparent. In this way, for a given state, the generated action can be close to the actions contained in the data set with a similar state. At the same time, residual action generation maintains a large potential for policy improvement. We summarize the entire algorithm on Algorithm 1.

Algorithm 1 Pessimistic Offline Policy Optimization (POPO)

```
Require
  • Data set $\mathcal{D}$, num of quantiles $N$, target network update rate $\eta$, coefficient $\xi$.
  • Distortion risk measure $\beta$, random initialized networks and corresponding target networks,
    parameterized by $\theta_i' \leftarrow \theta_i, \phi' \leftarrow \phi$, VAE $G = \{E(\cdot,\cdot;\omega_1), D(\cdot,\cdot;\omega_2)\}$.
for iteration $= 1, 2, ...$ do
    Sample mini-batch data $(s, a, r, s')$ from data set $\mathcal{D}$.
    # Update VAE
    $\mu, \sigma = E(a|s;\omega_1)$, $\hat{a} = D(z|s,;\omega_2)$, $z \sim \mathcal{N}(\mu,\sigma)$
    $\omega \leftarrow \arg\min_\omega \sum(a - \hat{a})^2 + \frac{1}{2}D_{\text{KL}}(\mathcal{N}(\mu,\Sigma)\|\mathcal{N}(0,I)$.
    # Update Critic.
    Set Critic loss $\mathcal{L}(\cdots;\theta)$ (Equation 6).
    $\theta \leftarrow \arg\min_\theta \mathcal{L}(\cdots;\theta)$.
    # Update actor
    Generate $a_{\text{new}}$ from Equation 9
    $\phi \leftarrow \arg\max_\phi Q_\beta(s, a_{\text{new}})$
    # Update target networks
    $\theta_i' \leftarrow \eta\theta_i + (1-\eta)\theta_i'$, $\phi' \leftarrow \eta\phi + (1-\eta)\phi'$
end for
```

6 Experiments

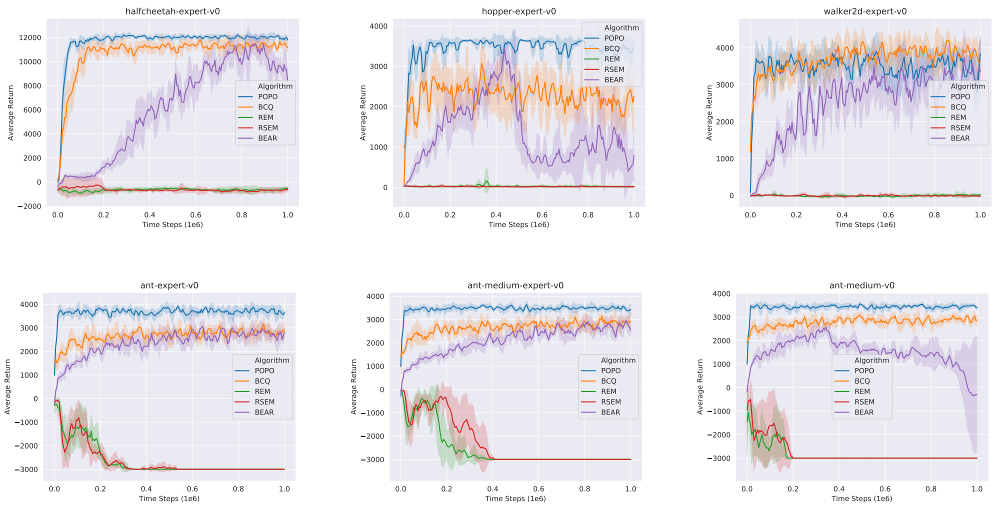

Figure 4: Performance curves for OpenAI gym continuous control tasks in the MuJoCo suite. The shaded region represents a standard deviation of the average evaluation over five seeds. The BCQ is stable when tested, but it is not as good as the POPO. BEAR suffers from performance decrease when training too much time. REM almost failed during testing.

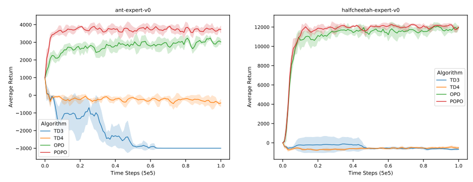

Figure 5: Performance curves for ablation study. The shaded region represents a standard deviation of the average evaluation. The results show that the pessimistic critic does have improvement than the original TD3 algorithm. VAE makes offline optimization successful because it deals with the OOD action issue. The combination between VAE and pessimistic critic would produce better results.

To evaluate our algorithm, we measure the performance of POPO on OpenAI gym [13] continuous control tasks. We utilize the various quality original transitions from the d4rl datasets [43] to train our model. "Expert" means the dataset is generated by a fine-tuned RL policy. "Medium-expert" marks the dataset is produced by mixing equal amounts of expert demonstrations and suboptimal data. The "medium" dataset consists of data generated by the suboptimal policy. Given the recent concerns about algorithms reflect the principles that informed its development [46] [47], we implement POPO without any engineering tricks so that POPO works as we originally intended for. We compare our algorithm with the recently proposed SOTA offline RL algorithms BCQ[1], REM[2], and BEAR[3]. We use the authors' official implementations. The performance curves are graphed in Figure 4. Note that we have not performed fine-tuned to the POPO algorithm due to the limitation of computing resources. Nevertheless, the results show that POPO matches or outperforms all compared algorithms.

6.1 Ablation Study

The main modifications of POPO are VAE and pessimistic distributional critic. To explore the rule of each component, we design the ablation study. We call the POPO version without pessimistic distributional value function OPO, meaning OPO adopts TD3 style value function. We call the POPO version without VAE TD4 since it is TD3 with a pessimistic critic. Also, we use the original TD3

[1]`https://github.com/sfujim/BCQ`.
[2]`https://github.com/agarwl/off_policy_mujoco`
[3]`https://github.com/rail-berkeley/d4rl_evaluations`

algorithm as a baseline because it neither has VAE nor a pessimistic critic. In this experiment, all the algorithms are tested over four seeds. The performance curves are graphed in Figure 5. The results show that the pessimistic distributional value function does have a robust performance improvement compared to TD3 and OPO. Besides, VAE makes the offline RL successful because it solves the OOD actions issue. The combination between VAE and pessimistic critic would produce better results. Because VAE brings significant performance improvements, the pessimistic value function's positive impact on performance is relatively small but still significant.

7 Discussion and Future Work

In this work, we study why off-policy RL methods fail to learn in offline setting and propose a new offline RL algorithm. Firstly, we show that the inability to interact with the environment makes offline RL unable to eliminate the estimation gap through the Bellman equation. We conduct fine-grained experiments to verify the correctness of our theory. Secondly, We propose the Pessimistic Offline Policy Optimization (POPO) algorithm, which learns a pessimistic value function to get a strong policy. Finally, we demonstrate the effectiveness of POPO by comparing it with SOTA offline RL methods on the MuJoCo locomotion benchmarks datasets.

8 Acknowledgments

We thank anonymous reviewers for their comments. Xinwen Hou is the corresponding author. Qiang He thanks Yuxun Qu for helping to draw figures. This project was supported by the National Natural Science Foundation of China (61379099).

Broader Impact

In this work we we propose the Pessimistic Offline Policy Optimization (POPO) algorithm which learns a conservative value function to get a strong policy in continuous control tasks. Our proposed method compares or outperforms the previous state-of-the-art policy gradient methods even in the challenging high-dimensional control tasks. Thus more fine-grained investigations on applying distributional RL methods into offline setting should be a promising future direction. Empirically, it is promising to apply our method to areas where interaction with the environment is extremely expensive, e.g., autonomous-driving, robotics, etc.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[4] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[5] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[6] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[7] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, et al. Deep q-learning from demonstrations. *arXiv preprint arXiv:1704.03732*, 2017.

[8] Wen Sun, J Andrew Bagnell, and Byron Boots. Truncated horizon policy search: Combining reinforcement learning & imitation learning. *arXiv preprint arXiv:1805.11240*, 2018.

[9] Jessica Chemali and Alessandro Lazaric. Direct policy iteration with demonstrations. In *IJCAI-24th International Joint Conference on Artificial Intelligence*, 2015.

[10] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[11] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[12] Qiang He and Xinwen Hou. Reducing estimation bias via weighted delayed deep deterministic policy gradient. *arXiv preprint arXiv:2006.12622*, 2020.

[13] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[14] Beomjoon Kim, Amir-massoud Farahmand, Joelle Pineau, and Doina Precup. Learning from limited demonstrations. In *Advances in Neural Information Processing Systems*, pages 2859–2867, 2013.

[15] Bilal Piot, Matthieu Geist, and Olivier Pietquin. Boosted bellman residual minimization handling expert demonstrations. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 549–564. Springer, 2014.

[16] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[17] Jonathan Ho, Jayesh Gupta, and Stefano Ermon. Model-free imitation learning with policy optimization. In *International Conference on Machine Learning*, pages 2760–2769, 2016.

[18] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062, 2019.

[19] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[20] Owain Evans, Andreas Stuhlmüller, and Noah D Goodman. Learning the preferences of ignorant, inconsistent agents. *arXiv preprint arXiv:1512.05832*, 2015.

[21] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[22] Geoffrey J Gordon. Stable function approximation in dynamic programming. In *Machine Learning Proceedings 1995*, pages 261–268. Elsevier, 1995.

[23] Dirk Ormoneit and Śaunak Sen. Kernel-based reinforcement learning. *Machine learning*, 49(2-3):161–178, 2002.

[24] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

[25] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

[26] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11784–11794, 2019.

[27] Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning*, pages 3652–3661, 2019.

[28] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. Striving for simplicity in off-policy deep reinforcement learning. *arXiv preprint arXiv:1907.04543*, 2019.

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

[30] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

[31] Ziyu Wang, Alexander Novikov, Konrad Żołna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *arXiv preprint arXiv:2006.15134*, 2020.

[32] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[33] Qing Wang, Jiechao Xiong, Lei Han, Han Liu, Tong Zhang, et al. Exponentially weighted imitation learning for batched historical data. In *Advances in Neural Information Processing Systems*, pages 6288–6297, 2018.

[34] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[35] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems*, pages 2318–2328, 2019.

[36] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. Benchmarking batch deep reinforcement learning algorithms. *arXiv preprint arXiv:1910.01708*, 2019.

[37] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. *arXiv preprint arXiv:1806.06923*, 2018.

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

[39] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[40] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[41] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. 2014.

[42] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[43] 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.

[44] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[45] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. *arXiv preprint arXiv:1707.06887*, 2017.

[46] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*, 2020.

[47] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.