Exploiting MDP Symmetries for Offline Reinforcement Learning

Jinzhu Luo, Qi Zhang
University of South Carolina

Abstract: Reinforcement Learning (RL) algorithms continue to face challenges in addressing out-of-distribution (OOD) issue in offline environments. One primary cause of such issue can be attributed to extrapolation error, which occur when an RL agent encounters actions that are not present in the offline dataset. In this study, we propose leveraging the inherent symmetry of the environment to expand the range of actions available for the agent's learning process. Our results demonstrate that by incorporating environmental symmetry, the performance and sample efficiency of basic RL algorithms can be improved in offline environment. This finding highlights the potential of harnessing environmental properties to enhance the generalization and robustness of offline RL algorithms.

Keywords: Offline RL, Symmetry

1 Introduction

In recent years, offline reinforcement learning (RL) has encountered difficulties in offline environments due to the evaluation of state-action pairs that are not supported by the offline dataset [1]. This issue arises from the max operation in Temporal Difference (TD) error calculations, leading to overestimation of the value when an RL agent encounters unseen state-action pairs. Although several studies have proposed algorithms to address this problem and achieved insightful results, such as BCQ[2]. We can find that the majority of current offline RL algorithm focus on limiting action selection to the existing state-action pairs.

In our study, we explore the utilization of symmetry properties in offline RL to tackle this challenge. Specifically, we employ an Actor-Critic (AC) algorithm [3] that leverages environmental properties to construct an extra symmetric critic, guiding the actor's policy more effectively. By incorporating symmetry into the RL framework, our approach demonstrates improvements in handling unseen state-action pairs, resulting in enhanced performance and sample efficiency in offline RL.

2 Related Work

Markov Decision Processes (MDP) are renowned for modeling sequential decision-making in fields like RL. MDP symmetries involve equivalent states and actions, maintaining invariant value functions and optimal policies under specific transformations, aiding in the development of efficient learning algorithms by reducing complexity and encouraging generalization across symmetrical scenarios.

Ravindran and Barto pioneered the exploitation of MDP symmetries to optimize RL algorithms, introducing a framework called Homomorphism, aimed at addressing both exact and approximate MDP minimization issues, facilitating knowledge transfer and lessening computational demand [4]. MDP Homomorphic Networks [5] leverage advancements in graph-based neural networks to encode symmetries, displaying enhanced learning efficacy and convergence. Shoshtari et al. introduced a unique method, Continuous MDP Homomorphism, and Homomorphic Policy Gradient [6], pre-

7th Conference on Robot Learning (CoRL 2023), Atlanta, USA.

serving the MDP structure and demonstrating superior performance in classical MuJoCo-control environments [7].

3 Preliminaries on MDP Symmetries

A MDP is a tuple (S, A, R, P, γ), with state space S, action space A, immediate reward function $R\colon S \times A \to \mathbb{R}$, transition function $P\colon S \times A \times S \to \mathbb{R}$, and discount factor $\gamma \in [0, 1]$. The goal of solving an MDP is to find a policy $\pi \in \Pi, \pi\colon S \times A \to \mathbb{R}$ (written as $\pi(a|s)$), where π normalizes to unity over the action space, that maximizes the return $G_t = \sum_t^\infty \gamma^t r_{t+1}$. The expected return from a state s under a policy π is given by the value function V_π . A related object is the Q-value Q_π , the expected return from a state s after taking action a under π. V_π and Q_π are governed by the Bellman equations. In an MDP, optimal policies π^* attain an optimal value V^* and corresponding $Q^*(s, a)$ given by $V^*(s) = \max_\pi V_\pi(s)$ and $Q^*(s, a) = \max_\pi Q_\pi(s, a)$.

Definition 1. *(MDP equivalence). A MDP symmetry function h from a MDP $M = (S, A, R, P, \gamma)$ to an MDP $\overline{M} = (\overline{S}, \overline{A}, \overline{R}, \overline{P}, \gamma)$ is a subjection map function. we define it as $h = (f, g_s)$ where $f : S \to \overline{S}$ and $g_s = A \to \overline{A}$, we also can got that:*

$$\overline{P}(f(s), g_s(a), f(s')) = \sum_{s'} P(s, a, s') \quad \overline{R}(f(s), g_s(a)) = R(s, a)$$

MDP symmetries refers to the presence of a structural equivalence or similarity between different parts of an MDP. By exploiting symmetries, we can identify equivalent states and actions, enabling us to compute optimal policies more efficiently. In the definition and theorem, we present the formulation of MDP symmetries and its underlying principles. These theoretical foundations were initially established and proved by Ravindran and Barto [4].

Theorem 1. *(optimal value equivalence). Let $\overline{M} = (\overline{S}, \overline{A}, \overline{R}, \overline{P}, \gamma)$ be the symmetry image of $M = (S, A, R, P, \gamma)$ under the $h = (f, g_s)$. Then:*

$$\overline{V}^*(f(s)) = V^*(s) \quad \overline{Q}^*(f(s), g_s(a)) = Q^*(s, a)$$

In the symmetry MDP, we define the V-value and Q-value as $\overline{V}$ and $\overline{Q}$.

4 Exploiting Symmetries for Offline RL

In the realm of offline RL, addressing the challenges posed by extrapolation errors in algorithms remains a paramount research focus. In this context, we propose a novel methodology to augment the valid or observable action space by capitalizing on the concept of symmetry. Our work is based on an actor-critic algorithm which is a widely used RL approach that combines the strengths of both value-based and policy-based methods by utilizing two separate neural networks, the actor for policy learning and the critic for value estimation, to optimize decision-making in complex environments. We introduce an extra symmetry critic on it, which we refer to as the "Abstract Critic". For our abstract critic, we obtain the abstract state-action pairs through the MDP symmetries function. We then use these abstract state-action pairs to update the abstract critic network:

$$\mathcal{L}_{abstract\_critic} = E_{(s_t, a_t)\sim\mathcal{B}}[(r_t + \gamma\overline{Q}(f(s_{t+1}), g_s(a_{t+1})) - \overline{Q}(f(s_t), g_s(a_t))^2]$$

Ultimately, we employ both the original critic and the abstract critic to update the actor network.

To leverage MDP symmetries in policy evaluation and policy improvement, it is imperative to establish that the original MDP and the abstract MDP possess equivalent policies. Given that the abstract MDP is derived from the original MDP's symmetries, a key advantage of symmetries is the preservation of object properties before and after the transformation. Consequently, the abstract MDP ought to preserve the properties inherent to the original MDP. This implies that the optimal policy should be congruent in both the original and the abstract MDPs. Thus, we make the assumption 1.

Assumption 1. *(policy equivalence). Let $\overline{M}$ be the symmetry image of M under the $h = (f, g_s)$. These policies maintain equivalence:*

$$\forall s \in S, \forall a \in A, \overline{\pi}(g_s(a) \mid f(s)) = \pi(a \mid s)$$

Similarly to prior works that use policy gradients [8] to get the optimal policy, we also define the performance measure as where the expectation is over the uncertainty in transitions, rewards, and initial states. We introduce a deep actor-critic algorithm, incorporating DDPG [9] and SAC [10], selected for their comparative simplicity, enabling a concentrated examination of MDP symmetries' impacts. DDPG and SAC concurrently act as standard algorithms for deterministic and stochastic policies, respectively. Pseudo-code is provided in Appendix A.

5 Experiment

5.1 Dataset

We evaluated our algorithm on two OpenAI Gym tasks that exhibit perfect symmetry propriety: Pendulum and Inverted-Pendulum [11]. The Pendulum and Inverted-Pendulum problems are classic control challenges. Both tasks serve as benchmark problems to evaluate and compare RL methods due to their nonlinear and dynamic nature. For data collection,we utilize DDPG as our base agent, we train it from a random policy to obtain a stable, high-reward policy, which we consider to be the optimal policy. During the training process, we will store every time step's information (s, a, r, s') as our dataset. We have set the maximum number of time steps to 10^6, which means our dataset will consist of 10^6 data points.

In the Inverted-Pendulum problem, the state space comprises four elements: the cart's position υ_1, the pole's vertical angle υ_2, the cart's linear velocity υ_3, and the pole's angular velocity υ_4. The action space is continuous, with actions representing the force exerted on the cart, in the range of $[-3, 3]$, where the magnitude indicates the force's strength and the sign denotes the direction. An interesting aspect of this environment is the perfect symmetry between the left and right spaces of the equilibrium position. Shown as Figure 1.

MDP Symmetries		
Inverted-Pendulum	Original	Symmetry
state	$(\upsilon_1, \upsilon_2, \upsilon_3, \upsilon_4)$	$(-\upsilon_1, -\upsilon_2, -\upsilon_3, -\upsilon_4)$
action	a	$-a$
Pendulum		
state	(χ_1, χ_2, χ_3)	$(-\chi_1, -\chi_2, -\chi_3)$
action	a	$-a$

Table 1: Symmetry Transformation Rules.

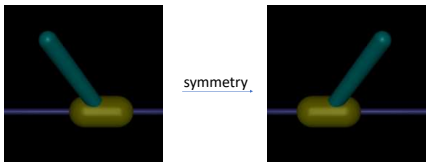


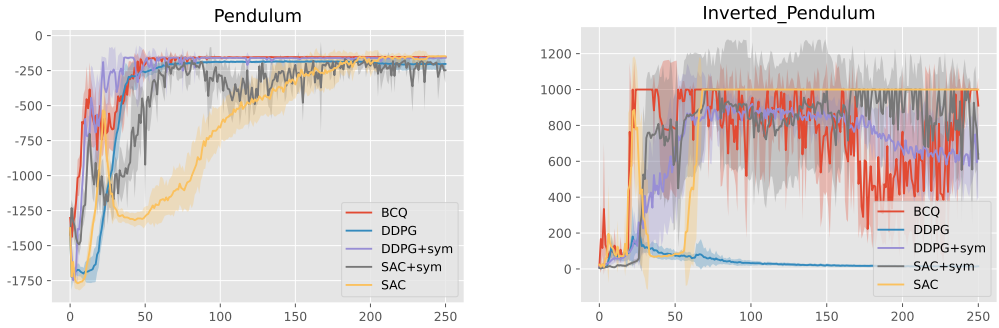
Figure 1: Example of Symmetry in Inverted-Pendulum.

In the Pendulum problem, the state space consists of three elements: the x (χ_1) – y (χ_2) coordinates of the pendulum's free end and its angular velocity (χ_3). The action space represents the torque applied to the pendulum. This problem has a similar mapping function to the Inverted-Pendulum.

5.2 Results

In our experiments, we also present results from the current state-of-the-art offline RL algorithms BCQ. BCQ is specifically tailored for offline RL and incorporates a Variational Autoencoder (VAE) network during action selection [12]. This addition constrains the range of actions to those present in the dataset, effectively preventing the selection of actions outside the dataset to avoid extrapolation error.

In Figure 2(a), we evaluate our proposed method and traditional RL algorithms on the dataset collected from Pendulum environment. This environment is considered relatively simple compared to others, due to its state dimensions and action range is less than others. Consequently, all the tested methods are capable of achieving high rewards and maintaining stability. However, this figure reveals that our method attains the optimal policy more rapidly than the corresponding original methods. These findings indicate that, in comparison to traditional algorithms, incorporating symmetry can enhance the efficiency of the learning process in offline RL tasks such as the Pendulum environment.



(a) Result of Pendulum.

(b) Result of Inverted-Pendulum.

Figure 2: Experimental results. The X-axis is the evaluation step and the Y-axis is the total reward. We evaluate these algorithms after every 4,000 training iterations

In Figure 2(b), we observe that the traditional classical control algorithm, DDPG, fails to achieve an optimal policy on the dataset collected using the same DDPG method. In contrast, our approach, which incorporates an abstract critic, reaches a significantly higher reward. The performance gap between the original DDPG algorithm and our method is quite substantial. The enhanced online algorithm, SAC, eventually attains a high and stable reward policy. However, during the initial stages, it exhibits an unstable learning curve. Our method, on the other hand, demonstrates a more favorable curve, achieving similarly high rewards more quickly. As previously mentioned, the integration of symmetry in RL algorithms can lead to greater learning efficiency, as exemplified by the improved performance and accelerated convergence of our proposed method.

6 Conclusion and Future Work

In this paper, we have presented the symmetry method incorporates an additional abstract critic into DDPG and SAC algorithms, utilizing identified symmetries. The symmetry function enables us to obtain corresponding abstract state-action pairs, which we employ to improve the RL algorithm performance in offline RL. Our results show that the proposed method can improve the performance and learning efficiency of the online algorithm in the offline environment, showing the potential of integrating symmetry-based techniques into offline RL algorithms.

In future work on the application of symmetry functions in offline RL, we aim to address several critical aspects to broaden our understanding and enhance the effectiveness of our approach. Firstly, we plan to test our method in various environments, as our current work focuses on only two environments, which limits the demonstration of the generality of symmetry functions. Secondly, our current experiments are based on two perfect symmetry functions, which are challenging to find in real-world scenarios. Therefore, we need to explore the use of approximation symmetry functions. Lastly, we will integrate symmetry functions into existing state-of-the-art offline RL algorithms, such as BCQ. These algorithms have already demonstrated promising results, and we hope that incorporating symmetry functions into offline RL algorithms will further enhance their performance.

Acknowledgment

This work is supported in part by NSF CAREER IIS-2237963.

References

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

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

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

[4] B. Ravindran and A. G. Barto. Symmetries and model minimization in markov decision processes, 2001.

[5] E. Van der Pol, D. Worrall, H. van Hoof, F. Oliehoek, and M. Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4199–4210, 2020.

[6] S. Rezaei-Shoshtari, R. Zhao, P. Panangaden, D. Meger, and D. Precup. Continuous mdp homomorphisms and homomorphic policy gradient. *arXiv preprint arXiv:2209.07364*, 2022.

[7] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[8] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

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

[11] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[12] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Algorithm 1 Symmetry Augmented DDPG

Input:policy $\pi_\theta(a|s)$, actual critic $Q_\varphi(s,a)$, abstract critic $Q'_\varphi(s',a')$, state map function$f(s)$, action map function $g_s(a)$, and dataset $\mathcal{B}$
Parameter: actor θ, θ', critic ϕ, ϕ', abstract critic φ, φ', target network update weight α

Let $t = 0$.
for t to T **do**
  Sample mini batch of N transitions (s, a, r, s') from $\mathcal{B}$.
  Critic and abstract critic update:
  Compute the Critic Loss $\mathcal{L}_{actuacl\_critic} = E_{(s_t,a_t)\sim\mathcal{B}}[(r_t+\gamma Q_{\phi'}(s_{t+1},a_{t+1})-Q_\phi(s_t,a_t))^2]$
  Compute the abstract Critic Loss $\mathcal{L}_{abstract\_critic} = E_{(s_t,a_t)\sim\mathcal{B}}[(r_t + \gamma\overline{Q}_{\varphi'}(\overline{s}_{t+1},\overline{a}_{t+1}) - \overline{Q}_\varphi(\overline{s}_t,,\overline{a}_t))^2]$
  Update parameter:

$$\phi \leftarrow \phi - \lambda_Q \nabla_\phi \mathcal{L}_{actual\_critic}(\phi)$$
$$\varphi \leftarrow \varphi - \lambda_{\overline{Q}} \nabla_\varphi \mathcal{L}_{abstract\_critic}(\varphi)$$

  Actor Update:
  if t mod 1000 **then**
    Compute policy loss

$$\mathcal{L}_{actor} = -E_{s\sim\mathcal{B}}[Q_\phi(s,\pi_\theta(s)) + \overline{Q}_\varphi(f(s), g_s(\pi_\theta(s)))]$$

    Update policy: $\psi \leftarrow argmax\mathcal{L}_{actor}$
    Update target networks:

$$\theta' \leftarrow \alpha\theta + (1-\alpha)\theta'$$
$$\phi' \leftarrow \alpha\phi + (1-\alpha)\phi'$$
$$\varphi' \leftarrow \alpha\varphi + (1-\alpha)\varphi'$$

  end if
end for

Appendix A
Pseudo-code

The components of our algorithm are: actual critic $Q(s,a)$, abstract critic $\overline{Q}(\overline{s},\overline{a})$, deterministic actor $a = \pi(s)$ or stochastic actor $a \sim \pi_\theta(a \mid s)$, symmetry function $h = (f(s), g_s(a))$, reward function $R(s)$, and probabilistic transition model $p(s' \mid s, a)$.

We use our MDP symmetry function $h = (f, g_s)$ to get the abstract state-action pairs. But according the determines policy and stochastic policy property, we make some difference in critic function sample action process.
For determinant policy: $a = \pi(s)$ and $\overline{a} = \pi(\overline{s})$.
For stochastic policy: $a \sim \pi_\theta(a \mid s)$ and $\overline{a} = g_s(a)$.
This is because ,we can got a same action from a deterministic policy if we have a same state. But for stochastic policy, we can not make sure we can got a same action due to the property of random sample in actor.

Actual and abstract critics are trained using TD(0) error for a faster reward propagation. The loss function for each critic is therefore defined as the expectation of the Bellman error estimated over transitions samples from the dataset $\mathcal{B}$. The two gradients are added together and a single actor policy update is conducted.

Here, $\mathcal{T}_\theta$ in SAC with Symmetry is a neural network transformation for reparameterize the policy. There is more info about this in the SAC paper.

Algorithm 2 Symmetry Augmented SAC

Input:policy $\pi_\theta(a|s)$, actual critic $Q_\phi(s,a)$, abstract critic $Q_\varphi(\overline{s},\overline{a})$, State map function$f(s)$, action map function $g_s(a)$, and dataset $\mathcal{B}$

Parameter: actor θ,θ', critic ϕ,ϕ', abstract critic φ,φ', target network update weight α, temperature parameter ρ

Let $t=0$.
for t to T **do**
  Sample mini batch of N transitions (s,a,r,s') from $\mathcal{B}$.
  Critic and abstract critic update:
  Compute the Critic Loss $\mathcal{L}_{actuacl\_critic} = E_{(s_t,a_t)\sim\mathcal{B}}[\frac{1}{2}(Q_\phi(s_t,a_t) - (r_t + \gamma(Q_{\phi'}(s_{t+1},a_{t+1}) - \rho\log(\pi_\theta(a_{t+1}|s_{t+1})))))^2]$
  Compute the abstract Critic Loss $\mathcal{L}_{abstract\_critic} = E_{(s_t,a_t)\sim\mathcal{B}}[\frac{1}{2}(\overline{Q}_\varphi(\overline{s}_t,\overline{a}_t) - (r_t + \gamma(\overline{Q}_{\varphi'}(\overline{s}_{t+1},\overline{a}_{t+1}) - \rho\log(\pi_\theta(\overline{a}_{t+1}|\overline{s}_{t+1})))))^2]$
  Update parameter:

$$\phi \leftarrow \phi - \lambda_Q\nabla_\phi\mathcal{L}_{actual\_critic}(\phi)$$
$$\varphi \leftarrow \varphi - \lambda_{\overline{Q}}\nabla_\varphi\mathcal{L}_{abstract\_critic}(\varphi)$$

  Actor Update:
  if t mod 1000 **then**
    Compute policy loss

$$\mathcal{L}_{actor} = E_{s_t\sim\mathcal{B},\epsilon_t\sim\mathcal{N}}[Q_\phi(s_t,\mathcal{T}_\theta(\epsilon_t;s_t))$$
$$+Q_\varphi(\overline{s}_t,\mathcal{T}_\theta(\epsilon_{\overline{s}};\overline{s}_t)) - \rho\log\pi_\theta(\mathcal{T}_\theta(\epsilon_t;s_t)|s_t)]$$

    Update target networks:

$$\theta' \leftarrow \alpha\theta + (1-\alpha)\theta'$$
$$\phi' \leftarrow \alpha\phi + (1-\alpha)\phi'$$
$$\varphi' \leftarrow \alpha\varphi + (1-\alpha)\varphi'$$

  end if
end for