Model-Based Offline Weighted Policy Optimization* (Student Abstract)

Renzhe Zhou, Zongzhang Zhang, Yang Yu

National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
zhourz@lamda.nju.edu.cn, {zzzhang, yuy}@nju.edu.cn

Abstract

A promising direction for applying reinforcement learning to the real world is learning from offline datasets. Offline reinforcement learning aims to learn policies from pre-collected datasets without online interaction with the environment. Due to the lack of further interaction, offline reinforcement learning faces severe extrapolation error, leading to policy learning failure. In this paper, we investigate the weighted Bellman update in model-based offline reinforcement learning. We explore uncertainty estimation in ensemble dynamics models, then use a variational autoencoder to fit the behavioral prior, and finally propose an algorithm called Model-Based Offline Weighted Policy Optimization (MOWPO), which uses a combination of model confidence and behavioral prior as weights to reduce the impact of inaccurate samples on policy optimization. Experiment results show that MOWPO achieves better performance than state-of-the-art algorithms, and both the model confidence weight and the behavioral prior weight can play an active role in offline policy optimization.

Introduction

Offline reinforcement learning attempts to learn policies from a fixed dataset, in which distribution shift between behavioral policy and target policy often results in the inability of online algorithms to learn effective policies. To address the problem, model-free algorithms focus on restricting policies to the support set of the behavioral policies (Kumar et al. 2019). Model-based algorithms, on the other hand, learn a dynamic model from the dataset and perform policy optimization based on the model. However, due to lack of data, regions not covered by the dataset may subject the policy to extrapolation error, which can lead to policy learning failure.

In this paper, to properly utilize model samples, we propose a weighted policy optimization algorithm called MOWPO, in which weights combining model confidence and behavioral priors are added to Bellman updates. MOWPO can make full use of the accurate data of the model and reduce the negative impact of inaccurate data on policy optimization, alleviating the problem of overly pessimism in some offline reinforcement learning methods.

*Corresponding author: Zongzhang Zhang. This work is supported by the NSFC (No. 62276126).

Our Method

In this section, we will describe our method in detail and propose the practical algorithm MOWPO in the end.

Dynamics Learning

Given an offline dataset $\mathcal{D} = \{(s, a, s', r)\}$, we use a deep fully connected neural network to characterize the dynamics model and reward function, modeled together with a Gaussian distribution, with parameters ϕ: $\hat{T}_\phi(s_{t+1}, r_t|s_t, a_t) = \mathcal{N}(\mu_\phi(s_t, a_t), \Sigma_\phi(s_t, a_t))$. To train this model, we minimize its negative log-likelihood on $\mathcal{D}$:

$$\mathcal{L}(\phi) = \mathbb{E}_{(s,a,s',r)\sim\mathcal{D}}[-\log \hat{T}_\phi(s', r|s, a)]. \tag{1}$$

We use N deep neural networks with the same architecture and different initialization to form an ensemble of dynamic models $\{T^i_\phi = \mathcal{N}(\mu^i_\phi, \Sigma^i_\phi)\}^N_{i=1}$. We define the uncertainty of (s, a) as $u(s, a) = \max_{i\in\{1,2,\cdots,N\}} \|\mu^i_\phi(s, a) - \frac{1}{N}\sum_{j=1}^N \mu^j_\phi(s, a)\|_2$. And we denote the model confidence on (s, a) based on its uncertainty as $c(s, a) = \exp(-u(s, a))$, thus scaling it to $(0, 1)$.

Behavioral Prior Estimation

The behavioral policy $\beta(a|s)$ is the policy for interactively sampling the dataset in the real environment. We denote the visiting probability of the behavioral policy to state s in the real environment as $\mu_\beta(s)$. Correspondingly, the state-action distribution of the behavioral policy in the real environment is $d^\beta(s, a) = \mu_\beta(s)\beta(a|s)$, which is the behavioral prior.

We use a variational autoencoder to estimate $d^\beta(s, a)$ with parameter α, aiming to simulate the visiting probability of (s, a). The encoder $E_{\alpha_1}(s, a)$ maps (s, a) into a latent space $\mathcal{Z}$, and outputs a latent vector z under the Gaussian distribution $E_{\alpha_1}(s, a) = \mathcal{N}(\mu_{\alpha_1}(s, a), \Sigma_{\alpha_1}(s, a))$. The decoder $D_{\alpha_2}(z)$ reconstructs the input from z: $(\hat{s}, \hat{a}) \sim D_{\alpha_2}(z)$. Our goal is to maximize the evidence lower bound, which is equivalent to minimizing the following loss function:

$$\mathcal{L}(\alpha) = KL[E_{\alpha_1}(z|s, a)\|P(s)] - \mathbb{E}_{z\sim E_{\alpha_1}}[\log D_{\alpha_2}(s, a|z)]. \tag{2}$$

Here, $P(s) \sim \mathcal{N}(0, I)$. We use the reconstruction probability $D_{\alpha_2}(s, a|z)$ as a proxy for $d^\beta(s, a)$ since a high reconstruction probability reflects a high visiting probability. We hope that samples with high visiting probability are close to the distribution of the offline dataset.

Dataset	Environment	BC	SAC	BEAR	UWAC	MOPO	MOWPO (Ours)
random	halfcheetah	2.1	30.5	25.1	14.5 ± 3.3	$\mathbf{35.4 \pm 2.5}$	33.5 ± 2.6
random	hopper	1.6	11.3	11.4	$\mathbf{22.4 \pm 12.1}$	11.7 ± 0.4	11.7 ± 0.8
random	walker2d	9.8	4.1	7.3	15.5 ± 11.7	13.6 ± 2.6	$\mathbf{15.8 \pm 4.8}$
medium	halfcheetah	36.1	−4.3	41.7	46.5 ± 2.5	42.3 ± 1.6	$\mathbf{47.1 \pm 0.7}$
medium	hopper	29.0	0.8	52.1	$\mathbf{88.9 \pm 12.2}$	28.0 ± 12.4	23.7 ± 4.0
medium	walker2d	6.6	0.9	59.1	57.5 ± 7.8	17.8 ± 19.3	$\mathbf{74.8 \pm 5.9}$
medium-replay	halfcheetah	38.4	−2.4	38.6	46.8 ± 3.0	53.1 ± 2.0	$\mathbf{57.6 \pm 1.2}$
medium-replay	hopper	11.8	3.5	33.7	39.4 ± 6.1	67.5 ± 24.7	$\mathbf{69.0 \pm 13.6}$
medium-replay	walker2d	11.3	1.9	19.2	27.0 ± 6.3	39.0 ± 9.6	$\mathbf{46.3 \pm 17.9}$

Table 1: Results on the D4RL benchmark. Each number is the normalized score of the policy at the last iteration, averaged over 3 random seeds, with $\pm$ representing the standard deviation. We bold the highest mean normalized score for each task.

Weighted Bellman Update

During policy evaluation, we use a weighted Bellman update to evaluate the Q-function with parameter ψ:

$$\mathcal{L}(\psi) = \frac{1}{2}\mathbb{E}_{(s,a)\sim d_f^\pi}[w(s,a)(Q_\psi(s,a) - \hat{\mathcal{B}}^\pi Q_{\psi^-}(s,a))^2]. \tag{3}$$

Here, d_f^π is the f-proportional mixture distribution of model data and offline data, $\hat{B}^\pi$ is the empirical Bellman operator applied to the target value function Q_{ψ^-}, and $w(s,a)$ is exactly the weight of the state-action pair (s,a).

Standard Bellman update is susceptible to error propagation. For model-generated samples, inaccurate target values $\hat{\mathcal{B}}^\pi Q_{\psi^-}(s,a)$ will affect the current $Q_\psi(s,a)$, which may cause the training process to be unstable and difficult to converge. In weighted Bellman update, different loss weights are applied to different samples, which change the step size of each sample in gradient descent. The proposed objective down-weights samples with lower confidence, while larger weights correspond to higher quality samples, which should play a more important role in updating. Therefore, weighted Bellman update makes the learning process more stable and consistent, reducing the effect of error propagation.

Intuitively, we want samples with smaller uncertainty estimates and larger behavioral priors to be weighted more heavily. Therefore, we design the weight as the combination of model confidence $c(s,a)$ and reconstruction probability $D_{\alpha_2}(s,a|z)$ for trade-off with hyperparameter λ:

$$w(s,a) = \lambda c(s,a) + (1-\lambda) D_{\alpha_2}(s,a|z). \tag{4}$$

Total Algorithm

Our algorithm MOWPO first learns ensemble dynamics models and a variational autoencoder, and then performs policy optimization under the MOPO (Yu et al. 2020). When updating the Q-function, MOWPO calculates the weights of the samples and applies a weighted Bellman update.

Experiments

We compare MOWPO with offline RL algorithms MOPO, UWAC (Wu et al. 2021), and BEAR (Kumar et al. 2019), together with BC and SAC (Haarnoja et al. 2018) on the D4RL benchmark and. Results are shown in Table 1. MOWPO achieves the highest scores on 6 of all 9 tasks. It beats other algorithms on most medium datasets and all medium-replay datasets. Compared to the base algorithm MOPO, MOWPO achieves higher scores with lower standard deviations on almost all datasets, empirically demonstrating the effect of weighted Bellman update.

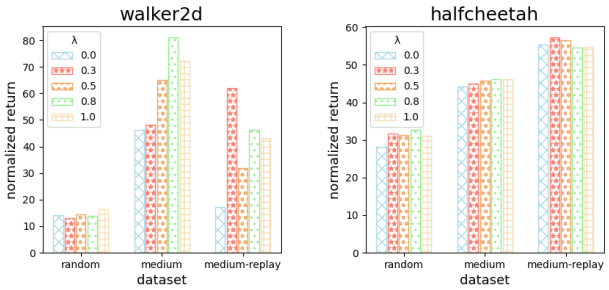

Figure 1: Comparison between different λ values on walker2d and halfcheetah.

The hyperparameter λ balances model confidence with behavioral prior. In Figure 1, we explore the impact of different values of λ. It shows that for random and medium datasets, the model is easier to learn so higher λ performs better. For diverse medium-replay datasets, lower λ is better. Both model confidence and behavioral prior positively influence the weighted Bellman update.

References

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 1856–1865.

Kumar, A.; Fu, J.; Soh, M.; Tucker, G.; and Levine, S. 2019. Stabilizing off-policy Q-learning via bootstrapping error reduction. In *NeurIPS*, 11761–11771.

Wu, Y.; Zhai, S.; Srivastava, N.; Susskind, J. M.; Zhang, J.; Salakhutdinov, R.; and Goh, H. 2021. Uncertainty weighted actor-critic for offline reinforcement learning. In *ICML*.

Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J. Y.; Levine, S.; Finn, C.; and Ma, T. 2020. MOPO: Model-based offline policy optimization. In *NeurIPS*.