Weighted Policy Constraints for Offline Reinforcement Learning

Zhiyong Peng, Changlin Han, Yadong Liu*, Zongtan Zhou*

College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China
pengzhyong@qq.com, ytcnhan@foxmail.com, liuyadong@nudt.edu.cn, narcz@163.com

Abstract

Offline reinforcement learning (RL) aims to learn policy from the passively collected offline dataset. Applying existing RL methods on the static dataset straightforwardly will raise distribution shift, causing these unconstrained RL methods to fail. To cope with the distribution shift problem, a common practice in offline RL is to constrain the policy explicitly or implicitly close to behavioral policy. However, the available dataset usually contains sub-optimal or inferior actions, constraining the policy near all these actions will make the policy inevitably learn inferior behaviors, limiting the performance of the algorithm. Based on this observation, we propose a weighted policy constraints (wPC) method that only constrains the learned policy to desirable behaviors, making room for policy improvement on other parts. Our algorithm outperforms existing state-of-the-art offline RL algorithms on the D4RL offline gym datasets. Moreover, the proposed algorithm is simple to implement with few hyper-parameters, making the proposed wPC algorithm a robust offline RL method with low computational complexity.

Introduction

RL has made great success in games, such as Atari games (Mnih et al. 2015), Go (Silver et al. 2016, 2017; Schrittwieser et al. 2020), Starcraft2 (Vinyals et al. 2019), etc. Another widely applied domain of RL is robot learning, such as quadrupedal robot (Hwangbo et al. 2019), robot manipulation (Andrychowicz et al. 2020) and mobile robot navigation (Fan et al. 2018). These success cases share a common characteristic of having perfect or near-perfect simulators, which enables RL agents easily to explore and collect large amounts of experience actively. In many real-world problems, however, it is difficult to model the environment dynamics, or requires a great deal of effort to construct a high-fidelity simulator. Deploying RL directly in the real world also faces several challenges: the time-consuming learning process and high learning cost caused by low sample efficiency. For example, it takes four months for a group of reality robotic arms to learn grasping tasks (Kalashnikov et al. 2018). Another problem is the safety risk associated with exploration, such as medical and autonomous driving, where unsafe exploration can be fatal. Offline RL directly utilizes passively collected static datasets to learn policy, providing a new approach to address the above challenges. Firstly, offline datasets are relatively easy to collect, such as for autonomous driving, where large amounts of data can be obtained from the driving records of vehicles; secondly, passive data recording does not introduce any new risks to the running systems.

The main challenge of offline RL is the distribution shift problem, which leads to an overestimation on the values of out-of-distribution (OOD) states. Due to the inability to interact with the environment, the value estimation error on OOD states cannot be corrected, which eventually leads to failure if the policy is not been properly constrained. A common practice to resist distribution shift is constraining the learned policy to be close to behavior policy explicitly (Kumar et al. 2019; Wu, Tucker, and Nachum 2019), or implicitly (Fujimoto, Meger, and Precup 2019). However, offline datasets often contain non-expert actions or even dangerous actions, constraining the learned policy to all these actions will force it to imitate undesirable behaviors and degrade the performance of the algorithm. For example, in autonomous driving, if the recorded dataset contains samples of accidents that may be caused by a drunken driver, imitating these kinds of behaviors will have catastrophic consequences. Based on such an observation, a natural question is whether it is possible to construct a more reasonable policy constraint method that only constrains the learned policy to desirable behaviors but not inferior behaviors by identifying the superiority of state-action pairs in the dataset? To recognize desirable actions, we borrow ideas from another class of methods in offline RL, i.e., weighted behavior cloning methods (Peng et al. 2019; Yang et al. 2021; Wang et al. 2020; Siegel et al. 2020). Weighted behavior cloning methods usually learn an advantage function to recognize desirable actions and then imitate them. However, weighted behavior cloning inherits the disadvantage of imitation learning: imitation learning is unable to outperform the demonstration policy, thus the policy learned by weighted behavior cloning cannot outperform the best behavior among the offline dataset. Moreover, weighted behavior cloning imitates only parts of states in the dataset, and the behaviors in other not imitated states are unknown, it may output unpredictable actions in these unconsidered states. In contrast, RL employs reward signals to

*Corresponding authors.

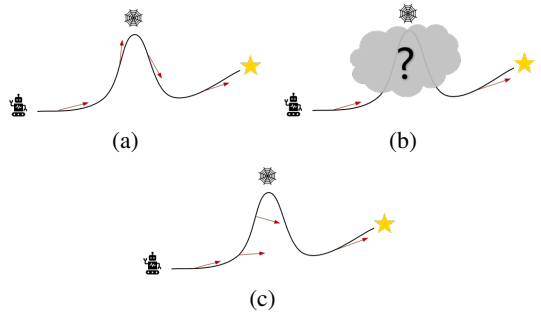

Figure 1: An intuitive comparison of a) constant policy constraint, b) weighted behavior cloning and c) weighted policy constraint methods. The task is to reach the goal ('star') and avoid being stuck ('net'). The constant policy constraint method constrains the learned policy not only to desirable actions but also dangerous ones. Behaviors in the dangerous area are unconsidered in the weighted behavior cloning method. Our weighted policy constraint method can imitate desirable actions while staying away from bad actions.

learn a value function, making it not only learns desirable behaviors but also stays away from inferior states. We argue that policy constraints-based approaches and weighted behavior cloning are complementary, and an appropriate combination of these two kinds of methods will benefit from each other. Figure 1 provides an intuitive comparison of the aforementioned methods and explains the motivation of our proposed method.

The main contributions of this paper are twofold. Firstly, we propose an offline RL framework: weighted policy constraints (wPC), which combines the advantages of policy constraints and weighted behavior cloning. The proposed framework slacks adverse constraints on the learned policy while retaining necessary constraints for desirable behaviors, improving the upper bound of the algorithm's performance. Secondly, a minimalist robust implementation is proposed. Experimental results on the offline RL datasets D4RL show that the proposed algorithm improves performance significantly upon both direct policy constraints counterpart and the weighted behavior cloning method, and outperforms existing state-of-the-art offline RL algorithms. Our algorithm implementation makes only minor modifications to the basic online RL method and introduces few hyperparameters.

Preliminaries

RL aims to solve sequential decision problems and is usually defined in the framework of Markov decision processes (MDPs). An infinite horizon discount MDP process is defined by $(S, A, P, r, \rho, \gamma)$ the tuple, where S is the state space, A is the action space, P is the dynamic transition function, r is the immediate reward, ρ defines the initial state distribution, and γ is the discount factor. Policy $\pi : a \sim \pi(s)$ defines the distribution of actions at a given state. The objective of the RL agent is to maximize the expected cumulative discounted reward:

$$J(\pi) = \mathbb{E}_{s_0\sim\rho, a_t\sim\pi(s_t), s_t\sim P(\cdot|s_t,a_t)}[\sum_{t=1}^{\infty}\gamma^t r_t(s_t, a_t)]. \quad (1)$$

Behavior cloning is a supervised learning method, which aims to reproduce behaviors given by expert demonstrations:

$$\pi_\theta := \arg\min_{\pi_\theta} D(\pi_\theta, \pi_b), \quad (2)$$

where π_θ is the learned policy and π_b is expert policy, D is a divergence metric such as KL divergence, least square error, etc.

Compared to behavior cloning, offline RL takes advantage of additional reward signals to learn policy and thus be able to outperform behavior policy. However, policy improvement on a static dataset will raise distribution shift, an unconstrained learning process tends to push the value function to infinity. One line of offline RL research alleviates distribution shift by directly constraining the learned policy to behavior policy, via an additional penalty term attached to the original RL optimization objective:

$$\pi_\theta := \arg\max_{\pi_\theta} \mathbb{E}_{s\sim B}[Q(S, \pi_\theta(s))] - \alpha D(\pi_\theta(s), \pi_b(s)), \quad (3)$$

where B is the dataset batch, α is a constant weighting factor that regulates the strength of constraint. Equation 3 is a combination of RL and imitation learning objectives. Intuitively, the first term aims to improve the learned policy so that it can outperform the behavior policy, and the second term guarantees the policy improved within a safety region.

A minimalist approach (Fujimoto and Gu 2021) of the aforementioned method combines TD3 (Fujimoto, Hoof, and Meger 2018) algorithm and behavior cloning (TD3+BC), which only adds few modifications to the original RL implementation:

$$\pi_\theta := \arg\max_{\pi_\theta} \mathbb{E}_{s,a\sim B}[\lambda Q(S, \pi_\theta(s)) - (\pi_\theta(s) - a)^2]. \quad (4)$$

Despite the simplicity, this algorithm works effectively and matches the performance of previous state-of-the-art offline RL algorithms. We build our wPC algorithm on top of this minimalist approach and regard it as a baseline.

Proposed Framework

In general policy constraint offline RL algorithms, the constraint weighting factor is a constant scalar, meaning that all actions are constrained equally. In order to make the learned policy only imitate desirable behaviors while get rid of inferior actions, we propose a new offline RL optimization objective as follows:

$$\pi_\theta := \arg\max_{\pi_\theta} \mathbb{E}_{s,a\sim B}[Q(s, \pi_\theta(s)) - w(s, a) D(\pi_\theta, \pi_b)], \quad (5)$$

where $w(s, a)$ is a weighting function depending on state and action, which can be calculated dynamically during the policy learning process. A suitable weighting function should assign greater weights to desirable state-action pairs and smaller weights to inferior behaviors. Equation 5 defines

Algorithm 1: Weighted Policy Constraints

Initialize: Initialize value networks Q_{ϕ_1}, Q_{ϕ_2}, V_ψ, target networks via $\overline{\phi}_1 \leftarrow \phi_1$, $\overline{\phi}_2 \leftarrow \phi_2$, actor π_θ, target actor $\overline{\theta} \leftarrow \theta$, and replay buffer $\mathcal{D}$

Setting hyper-parameter $\{c, \sigma, \gamma, \tau\}$, α

for *i=1 to N* **do**

Sample batch $B = (s, a, r, s', d)$ from dataset $\mathcal{D}$

$a'(s') = clip(\pi_{\bar{\theta}}(s') + clip(\epsilon, -c, c), a_{min}, a_{max}), \epsilon \sim \mathcal{N}(0, \sigma)$

$y = r + \gamma(1-d)\min_{\phi_{1,2}} Q_{\bar{\phi}_{1,2}}(s', a'(s'))$

$\phi_{1,2} \leftarrow \arg\min_{\phi_{1,2}} \frac{1}{|B|}\sum_B (Q_{\phi_{1,2}}(s,a) - y)^2$

$\psi \leftarrow \arg\min_\psi \frac{1}{|B|}\sum_B (V_\psi(s) - y)^2$

if *i mod policy_update_freqence == 0* **then**

$w(s,a) \leftarrow \mathbb{I}[Q_{\phi_1}(s,a) - V_\psi(s) > 0]$

$\lambda \leftarrow \alpha / Q_{\phi_1}.mean$

$\theta \leftarrow \arg\max_\theta \frac{1}{|B|}\sum_B \lambda Q_{\phi_1}(s, \pi_\theta(s)) - w(s,a)(\pi_\theta(s) - a)^2$

$\overline{\phi}_{1,2} \leftarrow (1-\tau)\overline{\phi}_{1,2} + \tau\phi_{1,2}$

$\overline{\theta} \leftarrow (1-\tau)\overline{\theta} + \tau\theta$

end

end

a general learning framework, this framework can be instantiated by different weight functions, divergence metrics, or implemented by implicit policy constraints approaches.

For the concrete implementation of this framework, we prefer to use simpler approaches rather than elaborate ones. Inspired by (Fujimoto and Gu 2021), we make minimal modifications on TD3 algorithm, and proposed a weighted policy constraints optimization objective:

$$\pi_\theta := \arg\max_{\pi_\theta} \mathbb{E}_{s,a\sim B}[\lambda Q(s, \pi_\theta(s)) - w(s,a)(\pi_\theta(s) - a)^2], \tag{6}$$

$$w(s,a) = \mathbb{I}[\hat{A}(s,a) > 0] = \mathbb{I}[\hat{Q}_\phi(s,a) - \hat{V}_\psi(s) > 0], \tag{7}$$

where $\mathbb{I}$ is the indicator function, $\hat{A}(s,a)$, $\hat{Q}_\phi(s,a)$ and $\hat{V}_\psi(s,a)$ are estimated advantage, state-action value function, and state value function respectively. The function is updated via double target network, and the V function fits the expectation of Q value. In practical implementation, we approximate the expectation of Q value by sampling and fit the function to target Q value for stability. This forms the weighted Policy constraints (wPC) algorithm, the pseudocode is presented in Algorithm 1. The source code is available at https://github.com/qsa-fox/wPC.

It is crucial to calculate the weights on the fly. If using a fixed binary weight, which means we always employ behavior cloning on the same part of the dataset, and apply unconstrained policy iteration on the others, the unconstrained policy iterations will push Q values to infinite large and eventually lead to failure. While dynamically updating the weights, the overestimated Q values will give these corresponding state-action pairs a positive advantage, forcing them to imitate the behavior policy, and preventing further error exploitation. Our later experiments confirm this claim and more details are given in the next section.

The wPC algorithm is a combination of the direct policy constraints method and weighted behavior cloning. If setting the weights to be a constant one, it reduces to TD3+BC, if setting the hyper-parameter λ to zero, then we get a standalone weighted behavior cloning (wBC) algorithm. Compared to the weighted behavior cloning method, the first item in Equation 6 encourages the agent to explore actions that maximize Q values, if we remove this item, there will be no "exploration" but only imitation left. The standalone wBC algorithm is similar to several existing weighted behavior cloning algorithms such as CRR, BAIL, and AWR, but with a different advantage estimation method. What surprised us is that the wBC itself performs pretty well, and matches several prior state-of-the-art methods. We regard wBC as a strong baseline and compare it with wPC in the experiments section.

Experiments

The purpose of our experiments is to answer the following questions: 1) How does the proposed algorithm perform compared to existing state-of-the-art offline RL methods? In this regard, we compare our algorithm with several recently proposed state-of-the-art offline RL methods, including TD3+BC (Fujimoto and Gu 2021), CQL (Kumar et al. 2020), OneStep (Brandfonbrener et al. 2021), DT (Chen et al. 2021), Rvs-R (Emmons et al. 2021) and baseline algorithms such as behavior clone (BC), best trajectories behavior clone (10%BC) and our own baseline wBC. 2) Does wPC algorithm effectively slack the constraints compare to the constant weight counterpart? We answer this question by comparing their Q-values during the learning process. 3) How do different policy constraint weight functions influence the performance? We compare four weight functions: pre-determined static weight function, exponential advantage weight function, binary advantage weight function (wPC), and randomized advantage binary weight function.

Performance on Offline RL Benchmarks

D4RL (Fu et al. 2020) is one of the main evaluation environments for offline RL, which consists of a wide of tasks and diverse datasets. We compare the performance of our algorithm to several prior offline RL methods (see Table 1). The results for BC, 10%BC, DT, Rvs-R, OneStep, and CQL are based on numbers summarized in the recent work by Emmons et al (Emmons et al. 2021), all experiments are evaluated on the "-v2" environments since there are major bugs in the old "-v0" version.

Our motivation for weighting the policy constraints item is to combine both advantages of policy constraint methods and weighted behavior learning methods, the results in Table 1 suggest that our algorithm does achieve this purpose. When taking a closer look at the results, we find wPC improves the baselines most notably on the "medium-replay" tasks. This phenomenon supports our argument well. The "medium" datasets are collected by a partially-trained policy and the "medium-replay" datasets consist of recording

Task name (-v2)	BC	10%BC	DT	RvS-R	OneStep	TD3+BC	wBC(ours)	CQL	wPC(ours)
halfcheetah-random	2.3	2.0	2.2	3.9	6.9	11.7	2.2	18.6	**19.6**
hopper-random	4.8	4.1	7.5	0.2	7.8	8.6	10.5	9.1	**19.9**
walker2d-random	1.7	1.7	2.0	7.7	6.1	0.9	**12.1**	2.5	0.7
halfcheetah-medium	42.6	42.5	42.6	41.6	**55.6**	48.2	45.6	49.1	53.3
hopper-medium	52.9	56.9	67.6	60.2	83.3	57.7	64.2	64.6	**86.5**
walker2d-medium	75.3	75.0	74.0	71.7	85.6	83.2	81.0	82.9	**86.0**
halfcheetah-medium-replay	36.6	40.6	36.6	38.0	42.4	44.6	41.1	47.3	**48.3**
hopper-medium-replay	18.1	75.9	82.7	73.5	71.0	67.4	86.6	**97.8**	97.0
walker2d-medium-replay	26.0	62.5	66.6	60.6	71.6	83.7	61.2	86.1	**89.9**
halfcheetah-medium-expert	55.2	92.9	86.8	92.2	93.5	90.7	92.5	85.8	**93.7**
hopper-medium-expert	52.5	**110.9**	107.6	101.7	102.1	106.1	105.8	102.0	95.7
walker2d-medium-expert	107.5	109.0	108.1	106	**110.9**	110.1	109.9	109.5	110.1
Total	475.5	674.0	684.3	657.3	736.8	712.9	711.4	755.5	**800.7**

Table 1: Performance of wPC and existing offline RL methods on D4RL gym locomotion-v2 datasets, measured by averaged normalized scores (bold indicates highest score). Our wPC algorithm outperforms prior methods and receives a highest total score.

all samples in the replay buffer observed during training until the policy reaches "medium" level of performance. Since the "medium-replay" dataset is a mix-up of "medium" and inferior "replay" trajectories, direct policy constraints methods such as TD3+BC constrain the learned policy to both "medium" quality and low-quality "replay" actions, and these low quality "replay" actions impair performance. As for weighted behavior cloning methods such as wBC, the algorithm filters out "medium" trajectories and imitates them. However, these kinds of methods have two disadvantages: the "replay" samples are completely discarded, which may also contain valuable information; behaviors on these "replay" states are unconsidered, which may be arbitrarily poor. Our proposed weighted policy constraint method can filter out desirable samples ("medium" samples in "medium-replay" tasks) for imitation and also further improve the policy by making use of these inferior samples ("replay").

The learning curves for wPC and two baselines (i.e., TD3+BC and wBC) are displayed in Figure 2. Curves are averaged over 5 seeds with shaded areas presenting deviation across seeds. We run 1 million steps for training, evaluate policy every 5 thousand steps, and report the average normalized returns of 10 evaluation episodes as the score. More experimental details are provided in Appendix A. This comparison can also be seemed as an ablation experiment to verify whether the dynamically calculated weights make sense compared to a constant weight and whether an additional Q-learning item improves performance effectively compared to the standalone weighted behavior cloning methods.

Constraint Strength Comparison via Q-value

Constraining the learned policy to behavior policy uniformly is over conservative and may be unnecessary, we instead employ weighted policy constraints and slack these unnecessary constraints, making room for policy improvement. Since the wPC algorithm is supposed to impose fewer constraints on policy compared to constant policy constraints counterparts like TD3+BC, allowing the policy more freely to pursue a maximum Q-value, thus the Q-values of wPC are expected to be higher. Figure 3 presents the Q-value learning curves, wPC reaches a higher number consistently compares to TD3+BC and wBC, which is well in line with our speculation. For the wBC approach, it can be observed that there are declining Q-value trends on "medium-replay" datasets, it is caused by a minimum operator on double Q, which is a standard approach in many online RL for addressing Q-value overestimation. We also try a mean operator on double Q in wBC algorithm, but find the Q-value unstable and reaches a very large number on some datasets.

Comparison of Different Weight Functions

Pre-determined static weight function The first thing we are curious about is whether it is feasible to utilize a pre-determined static weight instead of calculating it on the fly. In the weighted policy constraints framework, a straightforward idea is to pick out high-quality samples beforehand, then imitate these desirable samples while improving policy on others. We experiment on the 'walker2d-medium-v2' dataset (Figure 4) by filtering out one-half of the highest return trajectories as high-quality samples, assigning weight one on these samples and weight zero to others,

The result suggests that directly employing a pre-determined weight fails to combat the notorious extrapolation error (Fujimoto, Meger, and Precup 2019), which gives us a further inspection of the role dynamically updating weights plays. In the policy iteration steps of wPC algorithm, the overestimated out-of-distribution (OOD) actions will have a larger Q-value than the average action value (i.e., the function value), which will force the policy to imitate these state-action pairs by assigning them a positive weight, and prevent the policy to exploit the overestimated Q-value further. The importance of calculating weight dynamically is not straightforward at the first glance, but it's one of the crucial components for the practical algorithm to learn effectively.

Exponential advantage weight functions Exponential advantage weight function is a popular choice in weighted behavior cloning offline RL, which is formulated as follows:

$$w(s,a) = \exp \frac{\hat{A}(s,a)}{\beta} = \exp \frac{\hat{Q}_\phi(s,a) - \hat{V}_\psi(s)}{\beta}, \quad (8)$$

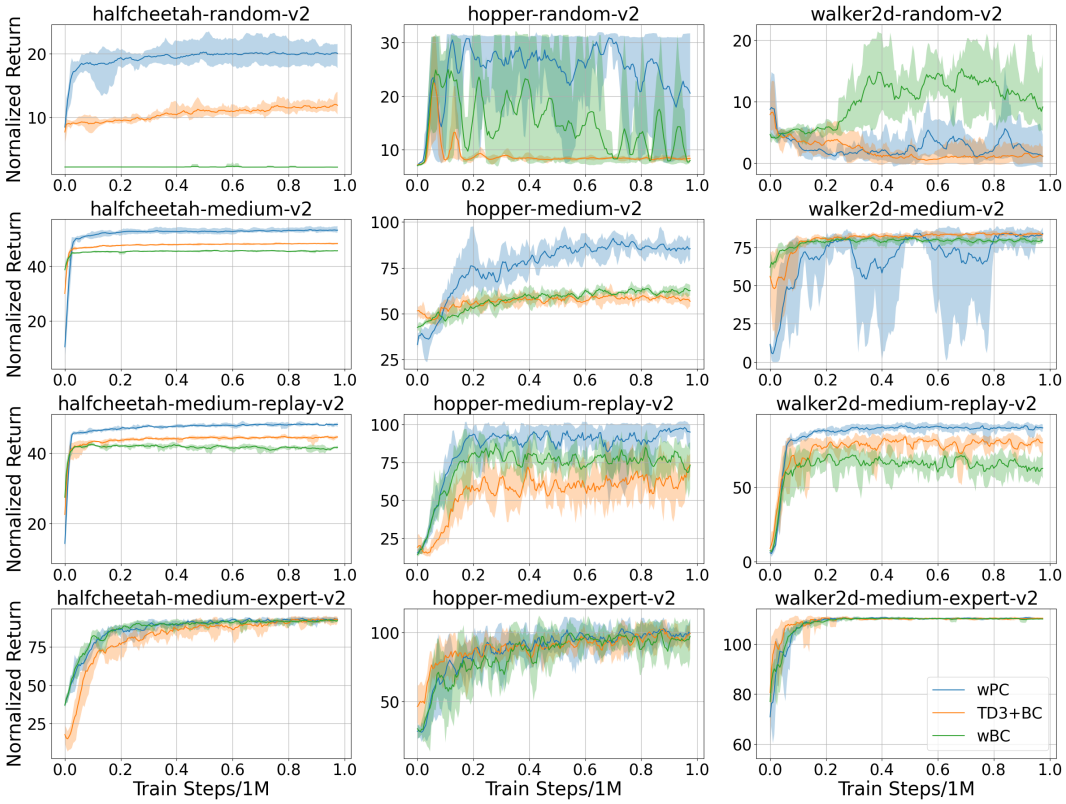

Figure 2: Learning curves of wPC (blue) algorithm compares to two baselines TD3+BC (orange) and wBC (green).

where β is a positive temperature hyper-parameter. The exponential advantage weight function can be regarded as a moderate version of the binary weight function, which constrains the policy to all actions in the dataset but with different degrees. Since it can also reduce the strength of constraints on inferior actions, we expect a performance improvement on top of the constant policy constraint baseline. The experimental results are presented in Fig 5, which shows that the exponential advantage weight method outperforms the constant policy constraint baseline method TD3+BC, though not as good as the binary weight version wPC.

Randomized weight function The motivation of wPC algorithm is to relieve undesirable constraints on policy while maintaining necessary ones, but it is difficult to inspect which actions in the dataset are desirable and whether the algorithm assigns proper weights to these samples. Since randomized binary weights can also slack constraints imposed on policy, the performance improvement of wPC may just come from such a random relaxation effect. We shuffle the calculated binary weights on wPC to get a randomized weight function, which eliminates preference on samples while maintaining the relaxation effects. Experimental results (Fig 5) suggest that the randomized weight function cannot work well. Though improves the performance slightly on several datasets compared to the constant policy constraint baseline, it impairs performance on other datasets, especially on "medium-expert" tasks, and gets a low total score. The performance of different weight functions is summarized in Table 2, the hyper-parameter α is the same for wPC-e, wPC-rnd and wPC, and for wPC-e, we set hyper-parameter β=1.

Related Works

Policy constraints are prevalent in offline RL to mitigate distribution shift. Weighted behavior cloning is another kind of method to learn policy from a non-expert dataset by selective imitation. To the best of our knowledge, there is no research that combines these two lines of work so far.

Policy Constraint

The policy constraints method measures the divergence of learned policy and behavior policy and appends the divergence as an additional penalty on the policy optimization objective. Different algorithms consider different divergence measurements. BEAR (Kumar et al. 2019) proposes support a set matching approach hat constrains actions to the dataset support set via maximum mean discrepancy(MMD). BRAC (Wu, Tucker, and Nachum 2019) utilizes KL divergence constraint on both policy and value function, and performs detailed experiments to validate the effects of different divergence metrics, such as KL divergence, MMD and Wasserstein Distance. TD3+BC (Fujimoto and Gu 2021) proposes a minimalist offline RL method that directly adds the behavioral cloning loss to the policy optimization ob-

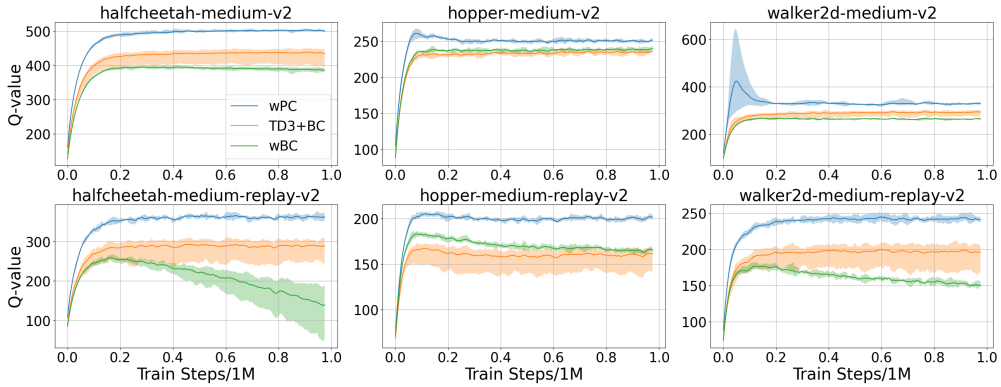

Figure 3: Q-value curves of wPC (blue), TD3+BC (orange) and wBC (green).

Task name (-v2)	**TD3+BC**	**wPC-e**	**wPC-rnd**	**wPC**
halfcheetah-random	11.7	16.0	15.5	19.6
hopper-random	8.6	9.1	7.9	19.9
walker2d-random	0.9	4.4	0.2	0.7
halfcheetah-medium	48.2	48.6	50.5	53.3
hopper-medium	57.7	67.0	66.5	86.5
walker2d-medium	83.2	83.8	59.3	86.0
halfcheetah-medium-replay	44.6	44.6	47.0	48.3
hopper-medium-replay	67.4	88.9	78.5	97.0
walker2d-medium-replay	83.7	85.6	77.1	89.9
halfcheetah-medium-expert	90.7	91.5	75.0	93.7
hopper-medium-expert	106.1	103.0	59.6	95.7
walker2d-medium-expert	110.1	110.2	103.1	110.1
Total	712.9	752.7	640.2	800.7

Table 2: Performance of different weight functions on D4RL gym datasets, scores are averaged on 5 seeds.

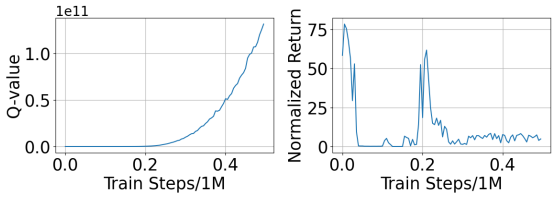

Figure 4: Q-value and performance of a pre-determined weight policy constraints approach on the "walk2d-medium-v2" dataset

jective, and matches the state-of-the-art methods despite its simplicity. Besides direct policy constraints as mentioned above, many offline RL algorithms perform implicit policy constraints to match the dataset distribution. CQL (Kumar et al. 2020) learns a conservative value function, which implies being constrained to behavior policy. BCQ (Fujimoto, Meger, and Precup 2019) learns a generative model to implicitly constrain the generated actions to the dataset distribution.

Weighted Imitation Learning

The basic idea of weighted imitation learning is to filter out inferior state-action pairs and imitate only desirable ones, which is first utilized in RL to improve RL performance. Self-imitation learning (Oh et al. 2018) imitates past high-returned trajectories to speed up learning in hard exploration problems. AWR (Peng et al. 2019) constructs an advantage function and to weight the loss of behavior cloning. ABM (Siegel et al. 2020) and CRR (Wang et al. 2020) construct different advantage functions from AWR. BAIL (Chen et al. 2020) defines the concept of dataset upper envelope and applies the upper bound envelope to select best actions for imitating. EVL (Ma et al. 2021) and IQL (Kostrikov et al. 2021) employ expectile regression to first learn the optimal value function and then perform behavior cloning based on advantage weighted regression. Separating the optimal value function regression and policy learning steps can avoid visiting the OOD actions thus making the learning process not affected by the distribution shift problem.

Conclusion

We propose a weighted policy constraint offline RL algorithm (wPC), which combines the advantages of policy constraint methods and weighted behavior cloning. Our algo-

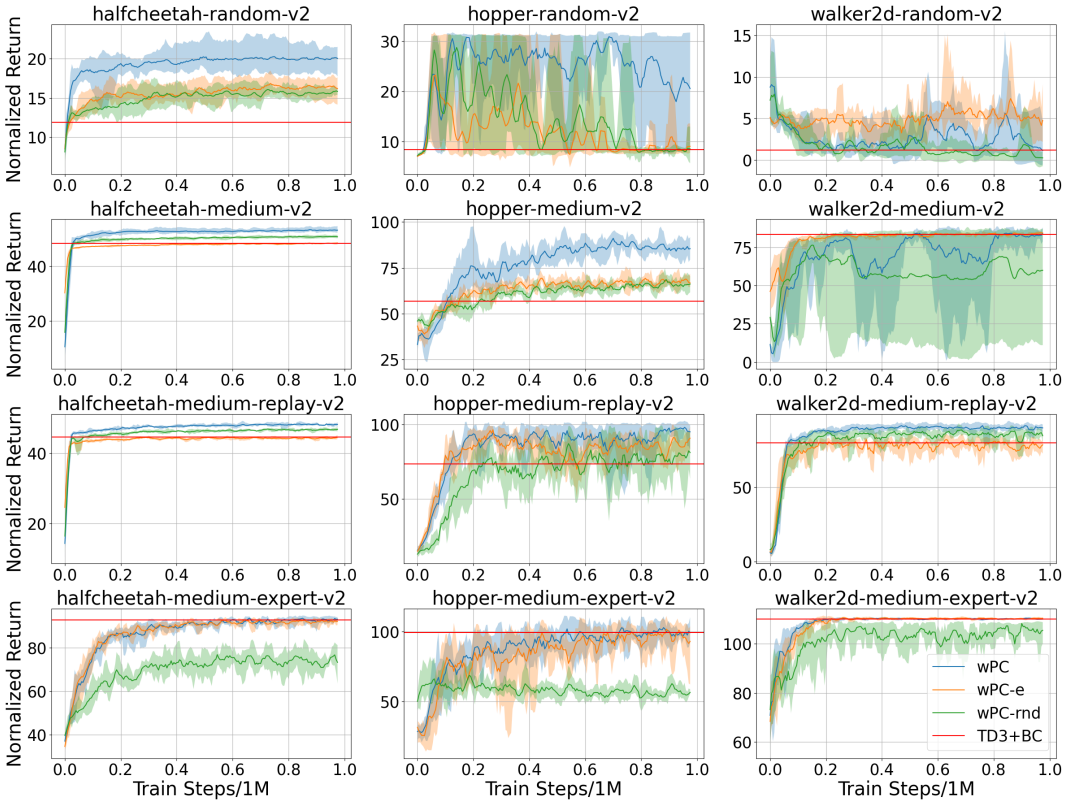

Figure 5: Learning curves of different policy constraint weight functions, where "wPC-e" and "wPC-rnd" represents exponential wPC and randomized wPC respectively.

rithm slacks constraints on the learned policy effectively and outperforms prior state-of-the-state offline RL methods on the D4RL gym datasets, via a minimal modification on the standard online RL method. Experiments suggest the importance of calculating the weights dynamically. The practical implementation builds on another minimalist offline RL approach TD3+BC, and improves the performance while introducing no additional hyper-parameters. The simplicity and efficiency make our algorithm a worthwhile attempt for many real-world offline RL problems.

Experimental Details

The practical wPC algorithm implementation builds on top of TD3+BC. The only hyper-parameter attached to standard online RL is the α, which regulates the constraint strength. We set α to 0.1 for "medium-expert" datasets and 2.5 for others. For TD3+BC, we follow the original implementation, keep it for all datasets and find it works better than other settings. There are no additional hyper-parameters to set for wBC, since it only has the standalone weighted behavior cloning item. Other hyper-parameters for TD3 components are presented in Table 3, and the neural network architectures are presented in Table 4. For hyperparameters choice, we determine α using grid search in [0.1, 1.0, 2.5, 5.0], the value 2.5 refers to TD3+BC. For the wPC-e experiments, we used β=1.0, which is selected using grid search in [0.1, 1.0, 10.0].

Hyperparameter	Value
Exploitation noise	0.1
Batch size	256
Discount	0.99
Tau	0.005
Policy noise	0.2
Noise clip	0.5
Policy updating frequency	2
Policy learning rate	3e-4
Q network learning rate	3e-4
V network learning rate	3e-4

Table 3: Hyperparameters for TD3 component.

Neural network	Hidden size	Activation function
Policy network	[256, 256]	ReLU, Tanh
Q network	[256, 256]	ReLU
V network	[256, 256]	ReLU

Table 4: Neural network architectures.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant U19A2083.

References

Andrychowicz, O. M.; Baker, B.; Chociej, M.; Jozefowicz, R.; McGrew, B.; Pachocki, J.; Petron, A.; Plappert, M.; Powell, G.; Ray, A.; et al. 2020. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1): 3–20.

Brandfonbrener, D.; Whitney, W.; Ranganath, R.; and Bruna, J. 2021. Offline rl without off-policy evaluation. *Advances in Neural Information Processing Systems*, 34: 4933–4946.

Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; and Mordatch, I. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34: 15084–15097.

Chen, X.; Zhou, Z.; Wang, Z.; Wang, C.; Wu, Y.; and Ross, K. 2020. BAIL: Best-action imitation learning for batch deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 18353–18363.

Emmons, S.; Eysenbach, B.; Kostrikov, I.; and Levine, S. 2021. RvS: What is Essential for Offline RL via Supervised Learning? *arXiv preprint arXiv:2112.10751*.

Fan, T.; Cheng, X.; Pan, J.; Manocha, D.; and Yang, R. 2018. Crowdmove: Autonomous mapless navigation in crowded scenarios. *arXiv preprint arXiv:1807.07870*.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2020. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*.

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

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.

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

Hwangbo, J.; Lee, J.; Dosovitskiy, A.; Bellicoso, D.; Tsounis, V.; Koltun, V.; and Hutter, M. 2019. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26): eaau5872.

Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. 2018. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, 651–673. PMLR.

Kostrikov, I.; Fergus, R.; Tompson, J.; and Nachum, O. 2021. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, 5774–5783. PMLR.

Kumar, A.; Fu, J.; Soh, M.; Tucker, G.; and Levine, S. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32.

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

Ma, X.; Yang, Y.; Hu, H.; Liu, Q.; Yang, J.; Zhang, C.; Zhao, Q.; and Liang, B. 2021. Offline Reinforcement Learning with Value-based Episodic Memory. *arXiv preprint arXiv:2110.09796*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Oh, J.; Guo, Y.; Singh, S.; and Lee, H. 2018. Self-imitation learning. In *International Conference on Machine Learning*, 3878–3887. PMLR.

Peng, X. B.; Kumar, A.; Zhang, G.; and Levine, S. 2019. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609.

Siegel, N. Y.; Springenberg, J. T.; Berkenkamp, F.; Abdolmaleki, A.; Neunert, M.; Lampe, T.; Hafner, R.; Heess, N.; and Riedmiller, M. 2020. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587): 484–489.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Wang, Z.; Novikov, A.; Zolna, K.; Merel, J. S.; Springenberg, J. T.; Reed, S. E.; Shahriari, B.; Siegel, N.; Gulcehre, C.; Heess, N.; et al. 2020. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33: 7768–7778.

Wu, Y.; Tucker, G.; and Nachum, O. 2019. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*.

Yang, Y.; Ma, X.; Chenghao, L.; Zheng, Z.; Zhang, Q.; Huang, G.; Yang, J.; and Zhao, Q. 2021. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 10299–10312.