UNCERTAINTY REGULARIZED POLICY LEARNING FOR OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent studies show the promising results of using online RL methods in the offline setting. However, such a learning diagram may suffer from an overtraining issue, that is, the performance of the policy degrades significantly as the training process continues when the dataset is not sufficiently large and diverse. In this work, we propose an alternative approach to alleviate and avoid the overtraining issue: we explicitly take the learning stability into account in the policy learning objective, and adaptively select a good policy before the overtraining issue happens. To do so, we develop an Uncertainty Regularized Policy Learning (URPL) method. URPL adds an uncertainty regularization term in the policy learning objective to enforce to learn a more stable policy under the offline setting. Moreover, we further use the uncertainty regularization term as a surrogate metric indicating the potential performance of a policy. Based on the low-valued region of the uncertainty term, we can select a good policy with considerable good performance and low computation requirements. On standard offline RL benchmark D4RL, URPL achieves much better final performance over existing state-of-the-art baselines.

1 INTRODUCTION

Reinforcement learning (RL) (Sutton & Barto, 2011) aims to build an agent that interacts with an online environment or simulator and learns from its own collected experience while offline RL try to learn purly from the logged dataset without the environment requirement (Fujimoto et al., 2019). However, offline RL scheme faces some new challenges, e.g., the extrapolation error or distribution mismatch (Fujimoto et al., 2019; Kumar et al., 2019). Existing offline RL methods try to alleviate these issues either by using the conservative or pessimistic update that learns a conservative Q function (Kumar et al., 2020; He & Hou, 2020) or by combining imitation learning to make the learned policy close to the behavior policy (Fujimoto et al., 2019; Chen et al., 2020; Rashidinejad et al., 2021). However, pessimistic update limits the final performance gain, while the imitation-learning based methods do not perform well when the data is collected by different behavior policies.

Recently, (Agarwal et al., 2020) propose an optimistic perspective on using online RL methods in the offline setting. That is, trained by sufficiently large and diverse offline datasets, online RL algorithms can achieve high quality policies without active interactions with the environment. However, sufficient datasets are not always available in practice. Moreover, it is also observed an performance drop issue based on our motivation example and the Figure 8-Hopper-v1 in (Agarwal et al., 2020) even when sufficient dataset is available. That is, the performance of the policy starts to degrade significantly as the training process goes through several epochs. Such an issue is more obvious on the complicated task where the trajectory length, state or action space is in high dimension.

The overtraining issue can be explained by the Ockham's Razor. Existing RL methods aim to achieve a stable learning process and simultaneously yield a good policy until the learning converges. However, it has been shown that, even in a linear situation, the learning process of Q-learning diversifies (Baird, 1995). The Deadly Triad introduced in (Sutton & Barto, 2011) also states that the danger of instability and divergence arises when one exploits function approximation, bootstrapping and off-policy training in the policy learning. The contradiction deteriorates when the online RL methods are directly applied to the offline setting.

In this paper, we follow the idea of using online RL methods in the offline setting from a better performance and further propose to handle the overtraining issue arised from Sutton & Barto (2011). Considering that the overtraining issue naturally exists when the offline dataset is not always suffi-

ciently large and diverse, we try the best to alleviate and avoid it instead of eliminating it. To alleviate, we explicitly take the stability as an intrinsic part of the learning objective. To avoid, different from previous offline RL methods that adopts the policy at the final learning step, we are dedicated to selecting a good policy before the overtraining happens. To do so, we propose an Uncertainty Regularized Policy Learning (URPL) method that can learn a performance predictable policy with the offline training, while maintain the online RL's high sample efficiency without conservative or pessimistic constraint. Specifically, we add an simple uncertainty regularization term in the policy update. The aim of the uncertainty regularization term is to regularize a policy with low variance so that it explicitly enforces the policy learning stable. It is worth noting that such a regularization term is for stabilizing the learning process instead of avoiding the distribution mismatch introduced in offline RL literature. Meanwhile, we also use the uncertainty regularization term as a surrogate metric indicating the potential performance of a policy for policy selection. By identifying the region of uncertainty regularization with low values, we can select the policy with a considerable good potential performance that may locate at any point/region of the learning process. On the other hand, even though we don't require the offline dataset to be sufficiently large and diverse, if the dataset only consists of very few different distributions, our method would not work well for such a case. In another word, we focus on an intermediate situation which is the most cases in practice.

We empirically demonstrate the efficiency of our approach on standard offline RL benchmark D4RL (Fu et al., 2020). In our experiments, we test URPL on the challenging tasks with small and mixed data, where previously offline RL methods are typically known to perform poorly (Fu et al., 2020). The results show that URPL outperforms existing state-of-the-art methods on these challenging tasks with respect to the final average return obtained by the learned policy. Moreover, we also investigate the connection of our proposed uncertainty regularization term with the policy performance, and we obtain that the small-valued uncertainty regions indeed correspond to high-valued performance regions. This demonstrates that the proposed uncertainty regularization term is an effective surrogate metric indicating the potential performance of a policy.

2 PRELIMINARIES

Online reinforcement learning. Here, online RL mainly focus on conventional off-policy RL which can learn from logged data from different behavior policies to some extent. A Markov decision process (MDP) (Sutton & Barto, 2011) can be defined by a tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$ with a state space $\mathcal{S}$, an action space $\mathcal{A}$, a stochastic reward function $R(s,a)$, transition dynamics $P\left(s' \mid s, a\right)$ and a discount factor $\gamma \in [0,1)$. A stochastic policy $\pi(\cdot \mid s)$ maps each state $s \in \mathcal{S}$ to a distribution over actions. For an agent following the policy π, the action-value function, denoted $Q^\pi(s,a)$, is defined as the expectation of cumulative discounted future rewards, i.e.,

$$Q^\pi(s,a) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R\left(s_t, a_t\right)\right] s_0 = s, a_0 = a, s_t \sim P\left(\cdot \mid s_{t-1}, a_{t-1}\right), a_t \sim \pi\left(\cdot \mid s_t\right)$$

The goal of RL is to find an optimal policy π^* maximizing the expected return with $Q^{\pi^*}(s,a) \geq Q^\pi(s,a)$ for all π, s, a. The Bellman equations characterize the optimal policy in terms of the optimal Q values, denoted $Q^* = Q^{\pi^*}$, via:

$$Q^*(s,a) = \mathbb{E}R(s,a) + \gamma \mathbb{E}_{s' \sim P} \max_{a' \in \mathcal{A}} Q^*\left(s', a'\right)$$

For complex tasks, such as high dimensional state-action space or long time horizon, Q functions are usually approximated by neural networks. Suppose the approximator is represented by $Q_\phi(s,a)$, with parameters ϕ, and we have the dataset $\mathcal{D}$ of transitions $\left(s, a, r, s', d\right)$, where d indicates a terminal state. Then, we can use the mean-squared Bellman error(MSBE) to improve the Q function iteratively:

$$L(\phi, \mathcal{D}) = \mathop{\mathrm{E}}_{(s,a,r,s',d) \sim \mathcal{D}} \left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d) \max_{a'} Q_\phi\left(s', a'\right)\right)\right)^2\right]$$

Entropy-Regularized Reinforcement Learning Some state-of-the-art RL methods (Haarnoja et al., 2018; Chen et al., 2021) employ the entropy term to regularize the policy learning for a better

exploration and stability. Entropy is a quantity representing the uncertainty of a random variable (Wehrl, 1978). Suppose x is a random variable with the probability mass or density function $P(x)$. Then, the entropy of x is computed from its distribution $P(x)$ according to

$$H(P) = \mathop{\mathrm{E}}_{x\sim P}[-\log P(x)]$$

In entropy-regularized reinforcement learning, the policy gets a reward and the entropy of the policy at each time step. This reformulates the RL objective to:

$$\pi^* = \arg\max_{\pi} \mathop{\mathrm{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\left(R\left(s_t, a_t, s_{t+1}\right) + \alpha H\left(\pi\left(\cdot \mid s_t\right)\right)\right)\right]$$

where $\alpha > 0$ is the trade-off coefficient.

Offline Reinforcement Learning. The methods mentioned above are mainly considered as online RL as they alternate between updating a policy and using that policy to collect more data through interacting the environment. In contrast, offline RL optimizes the policy using logged data without any further interaction with the environment. Consequently, offline RL setting removes design choices related to the replay buffer and exploration. However, it is generally considered challenging due to the distribution mismatch between the learned policy and the policy from collected data, i.e., behavior policy. When the sampled action is beyond the distribution of the behavior policy, conventional off-policy RL methods usually yield overestimated Q-values (Fujimoto et al., 2019), and some offline RL methods (Kumar et al., 2020; Agarwal et al., 2020) aim to solve such issues.

3 Related Works

Offline RL. Previous model-free offline RL methods can be classified into two classes: policy regularization and critic penalty. Policy regularization methods attempt to solve the distribution mismatch by constraining the learned policy to be close to the behavior policy as measured by some distance metrics, e.g., KL-divergence (Fujimoto et al., 2019), Wasserstein distance (Wu et al., 2019), and MMD (Kumar et al., 2019). BAIL (Chen et al., 2020) and ABM (Siegel et al., 2020) constrain the policy to learn from the better trajectories instead of all data. Generally, the above methods try to combine the limitation learning with reinforcement learning to some extent. In terms of critic penalty, CQL (Kumar et al., 2020) aims to learn a conservative Q-function such that the expected value of a policy under this Q-function lower bounds its true value. POPO (He & Hou, 2020) learns a pessimistic value function to get a strong policy. REM (Agarwal et al., 2020) utilizes an ensemble Q functions for robust offline learning. It has been proved to be effective on Atari 2600 games (Bellemare et al., 2015). Different from these works, our method focus on those more challenging tasks with scarce or mixed data, where these existing offline RL methods perform poorly. Moreover, we do not regularize the learned policy stay close to the behavior policy or learn a conservative/pessimistic values functions, which makes our method is less conservative and capable of achieving a better performance.

Offline policy selection/Off-policy evaluation. We further discuss the connection of our method with offline policy selection (OPS) or off-policy evaluation (OPE) (Yang et al., 2020; Voloshin et al., 2019). These methods usually employ the Bayesian posterior or importance sampling techniques to evaluate the value of the policy (Precup et al., 2000; 2001; Zhang et al., 2020; Gelada & Bellemare, 2019; Hallak & Mannor, 2017; Levine et al., 2020), which bring extra computational overhead. When the number of policies for selection/evaluation is huge, these methods may be impractical. In contrast, our method does not have such problems as URPL builds a connection between the policy performance and its uncertainty. We don not need to explicitly evaluate the policy. Instead, each policy's performance can be defined by the SLA metric without any extra computing. Moreover, the main purpose of our method is to learn a good policy with the performance as high as possible under the offline setting, rather than to evaluate a policy.

Uncertainty estimation. Uncertainty estimation in deep reinforcement learning have generally been used to encourage exploration (Azizzadenesheli et al., 2018; O'Donoghue et al., 2018). Other methods have examined approximating the Bayesian posterior of the value function(Touati et al., 2019; Osband et al., 2018), again using the variance to encourage exploration to unseen regions of the state space. In model-based reinforcement learning, uncertainty has been used for exploration, but also for the opposite effect–to push the policy towards regions of certainty in the model. This is used to combat the well-known problems with compounding model errors, and is presented in policy

search methods (Yu et al., 2020; Kidambi et al., 2020; Xu et al., 2019), combined with trajectory optimization (Chua et al., 2018) or value-based methods (Buckman et al., 2018). In our work, we employ an uncertainty regularization term to stabilize the policy learning. We build a connection between the uncertainty of a policy and its performance, that can be used for the final policy selection.

4 UNCERTAINTY REGULARIZED POLICY LEARNING(URPL)

In this section, we introduce our proposed method in details. To start, we firstly introduce an illustrative example to demonstrate the issue of using online RL methods (off-policy) in the offline setting. Figure 1 shows the performance of a state-of-the-art online algorithm REDQ (Chen et al., 2021), which uses an ensemble Q functions, on a mixed dataset. Herein, a mixed dataset contains data from diverse behavior policies collected previously. As can be seen, its performance grows up quickly but deteriorates drastically as training process continues. This is so-called over-training issue. Obviously, taking the policy at the last step as the final target policy is not a wise choice, however, this is the conventional solution adopted by most of existing offline RL methods. Moreover, we also observe that, even if we can select the best performance achieved by REDQ, which is located around $1e^5$ time steps, it is lower than the performance of the best trajectory contained in the mixed data, i.e., 1900 as indicated by the orange dotted line in Figure 1. As Agarwal et al. (2020) claimed when dataset is sufficiently diverse and large, off-policy RL can learn effectively compared with offline RL methods, to this end, the underlying reason may be the dataset is not sufficiently diverse and large. For the diverse dataset, the OOD issue (Kumar et al., 2020) may be not important for off-policy RL.

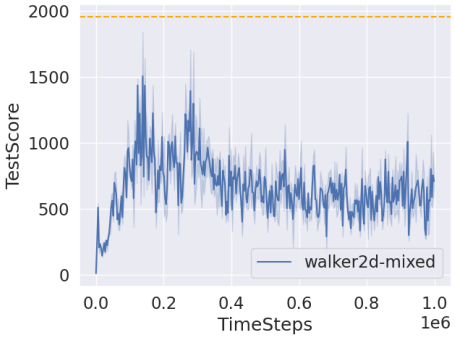


Figure 1: REDQ's training curve on D4RL mixed walker2d dataset.

Based on these observations, our method has two main purposes. Firstly, we aim to yield a policy whose performance is as good as that of the best trajectory in the mixed data or better than it when diverse data is available. Secondly, we want to adaptively select a good policy that may locate at any point/region of the whole learning process to avoid the over-training issue. To achieve the two purposes, we propose our method: Uncertainty Regularized Policy Learning (URPL).

Uncertainty Regularized Policy Update. The key idea of URPL is to explicitly take the stability into account in each step of policy learning. In each step of policy update, we reduce the uncertainty of the learned policy as much as possible. Specifically, we take advantage of the ensemble double q-learning: REDQ (Chen et al., 2021). URPL uses N Q functions. When compute a target for Q, URPL follows a similar step as REDQ, that is, it randomly samples a set $\mathcal{M}$ from N, and computes the Q target y with the following equation:

$$y = r + \gamma \left(\min_{i \in \mathcal{M}} Q_{\phi_{\text{targ},i}} \left(s', \tilde{a}' \right) - \alpha \log \pi_\theta \left(\tilde{a}' \mid s' \right) \right), \quad \tilde{a}' \sim \pi_\theta \left(\cdot \mid s' \right), \tag{1}$$

Then, every Q_i function parameters is updated by gradient descent using:

$$\nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s') \in B} \left(Q_{\phi_i}(s,a) - y \right)^2. \tag{2}$$

However, when updating the policy, we use:

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left(\frac{1}{N} \sum_{i=1}^{N} Q_{\phi_i} \left(s, \tilde{a}_\theta(s) \right) - \alpha \log \pi_\theta \left(\tilde{a}_\theta(s) \mid s \right) - \beta \sigma \left(\left\{ Q_{\theta_i}(s,a) \right\}_{i=1}^{N} \right) \right) \tag{3}$$

where σ is the standard deviation across the ensemble Q functions, α and β are weighted hyperparameters. As can be seen, the second (the entropy term) and the third (the variance term) terms play a role as the uncertainty regularization on the learned policy. By doing so, the policy not only maximizes the the mean value of Q as the conventional online RL methods do, but also minimizes its uncertainty explicitly. Note that the major difference of URPL from REDQ is on the policy update

Algorithm 1: Uncertainty Regularized Policy Learning

Initialization: Initialize policy parameters θ, N Q-function parameters $\phi_i, i = 1, \ldots, N$, replay buffer $\mathcal{D}$. Set target parameters $\phi_{\text{targ},i} \leftarrow \phi_i$, for $i = 1, 2, \ldots, N$, evaluation steps T, empty uncertainty list L
for *each time steps* t **do**
  Sample a mini-batch $B = \left\{\left(s, a, r, s'\right)\right\}$ from $\mathcal{D}$
  Sample a set $\mathcal{M}$ of M distinct indices from $\{1, 2, \ldots, N\}$
  Compute the Q target y (same for all of the N Q-functions) using equation 1
  Update Q using equation 2
  Update target networks with $\phi_{\text{targ},i} \leftarrow \rho\phi_{\text{targ},i} + (1-\rho)\phi_i$
  Update policy parameters θ with gradient ascent using equation 3
  Calculate the uncertainty $\mathcal{U}_t$ using equation 4
  Append the $\mathcal{U}_t$ to L
  Calculate the mean uncertainty value every T steps: $\mathcal{U} = \text{sum}(L)/T$
  Empty L
end

step, where the former considers both the entropy of the policy and variance of the Q functions while the latter only considers the entropy term. As empirically demonstrated by the experimental results, URPL significantly outperforms REDQ with respect to the best performance that can be achieved.

Adaptive Policy Selection. To avoid using the final policy of the last step that suffers from the over-training issue, we need to propose a surrogate metric indicating the potential performance of a policy to guide the policy selection. In this work, we propose to use the uncertainty term used in the policy update as the surrogate metric:

$$\mathcal{U} = H(\pi) + \varepsilon\sigma(Q(s, \pi(s))) \tag{4}$$

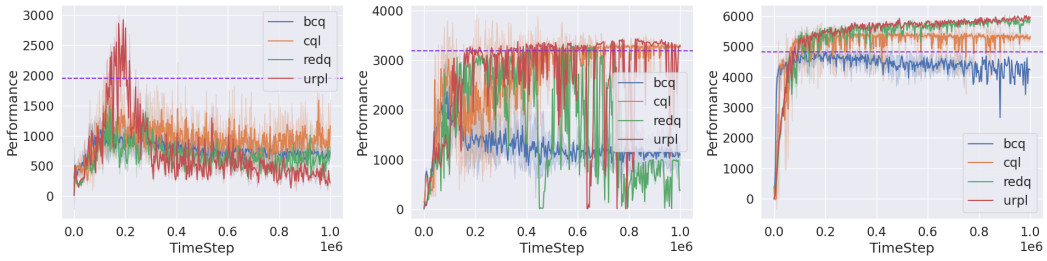
where ε is a hyperprameter that weight the Qs' standard deviation. H is the stochastic policy π's entropy defined by $H(\pi(\cdot|s)) = \underset{a\sim\pi}{\mathrm{E}}[-\log\pi(a|s)]$. The intuition behind such a metric is simple: a random policy will not have a high performance anyway. When the policy achieves a high performance, it should have a relatively low uncertainty. In the experiments, we explicitly analyze how the performance varies with respect to the individual entropy term and the individual variance term. We find that the connection of the performance with the entropy term is more steady and organized than the connection of the performance with the variance term. Therefore, we suggest using a small ε or directly use the individual entropy term in the surrogate metric. Moreover, the proposed surrogate metric can be applicable to other existing RL methods for the policy selection. We have also verified its effectiveness on the REDQ and CQL, in addition to our URPL. Note that the second term, i.e., the variance term, does not exist for some RL methods as it is specific to methods using multiple Q functions. In this case, we can directly use the entropy term as the surrogate metric.

Clearly, the relationship between the performance and the uncertainty is not monotonic. Thus, we are more interested in a region instead of a certain point. When $\mathcal{U}$ stays at a relatively low and stable level, we define it: **Stable Learning Area(SLA)**, the policy tends to keep increasing its performance. When the policy leaves that area, the policy's performance begins to degrade. As a result, we can utilize the SLA as a final criterion to gain a high performance final policy and avoid the overtraining issue. In practice, the uncertainty term is usually calculated by the recent average value for a stable estimation.

When we have such a connection, there are many methods to help us find a better final policy with high-possibility in practice. For instance, we can use the policy before the turning point as the final policy, if there is a clear surge point. However, there might be no clear turning point for some tasks where $\mathcal{U}$ only slowly go up, in such case, we can use the policy around the end position of the lowest SLA as the selected policy. The pseudocode of our method's training process is presented in details in algorithm 1.

Environment	Max time horizon	State dimension	Action dimension	Data set size
Walker2d	Infinite	17	6	100930
Hopper	Infinite	11	3	401598
Halfcheetah	finite	17	6	101000

Table 1: Complexity for each tasks. max time horizon means the maximum steps when the task terminates. From this table, we can see the walker2d is the most complicated while HalfCheetah is the least one.



(a) Walker2d-mixed (b) Hopper-mixed (c) Halfcheetah-mixed

Figure 2: Comparative performance of different methods on benchmark D4RL's mixed data set, averaged over 4 seeds. The purple dotted line is the best trajectory in the data.

5 Experiments

In this section, we conduct empirical evaluation to (1) verify the performance superiority of our method compared other state-of-the-art offline methods, and (2) analyze if the uncertainty concerns in our policy loss is effective or not to build the performance connection.

5.1 Experiment Setup

All the evaluations are done on the continuous control task: Mujoco (Todorov et al., 2012), and we use the commonly used offline dataset D4RL(Fu et al., 2020) for training which is under the Apache License 2.0 license. As we focus on the diverse dataset, we only use the mixed data for our experiments. Only 3 tasks meet the requirement, namely Walker2d-medium-replay, Hopper-medium-replay and HalfCheetah-medium-replay which consists of recording all samples in the replay buffer observed during training until the policy reaches the "medium" level performance. We call them Walker2d-mixed, Hopper-mixed, and Halfcheetah-mixed for short. The description of the three tasks are is shown in Table 1. In practice, all above tasks take 1000 time steps as their max horizon. As can be seen, walker-2d is the most complicated task with all large volumes on Max time horizon, state dimension and action dimension.

As the authors of D4RL claimed that there is a bug on the Hopper-mixed-v0 dataset[1], we use the fixed version Hopper-mixed-v2 instead of the Hopper-mixed-v0. However, we still use the v0 version data of Walker2d and HalfCheetah tasks for backwards compatibility with other baselines. For a fair comparison, we re-run BCQ and CQL for all three tasks. We use their official code from their papers, and follow the default hyper-parameter settings. In the experiments, we set the evaluation step T as 3000, the number of Q function N as 5. All experiments results are averaged over 4 seeds, and we run it on a PC with an NVIDIA RTX 2060 GPU and 16 cores CPU.

5.2 Comparative Evaluation

We firstly compared our method with two offline RL baselines: BCQ (Fujimoto et al., 2019) and CQL (Kumar et al., 2020). We also compared with a state-of-the-art online method REDQ (Chen et al., 2021) under the offline setting. To give a comprehensive comparison, we include the evaluation results tested by the D4RL authors (Fu et al., 2020). All methods'score are reported against the number of their training steps.

[1]D4RL: github.com/rail-berkeley/d4rl

Environment	BT	BC	SAC-off	BEAR	BRAC-p	BRAC-v	AWR	URPL(ours)	REDQ	BCQ	CQL
walker2d-mixd	1956	518	87	883	-11.5	44.5	712	463	711	688	1123
hopper-mixd	3189	364	93	1076	0.5	-0.8	904	**3322**	384	989	**3271**
halfcheetah-mixd	4830	4492	-581	4517	5208	4926	4727	**5964**	**5856**	4463	5301

Table 2: 1 million time-steps training or gradient steps for URPL, REDQ-off, BCQ and CQL. BC means behavior cloning. The performance are represented by the last iteration average return (raw, un-normalized scores). BT means the best trajectories(BT) return from the offline data. The scores for BC/SAC-off/BEAR/BRAC-p/BEAC-v/AWR are directly taken from the paper D4RL (Fu, Kumar, Nachum, Tucker, and Levine, 2020).

Environment	BT	URPL$_w$	REDQ$_w$	CQL$_w$	URPL$_{wo}$	REDQ$_{wo}$	BCQ$_{wo}$	CQL$_{wo}$
walker2d-mixd	1956	**2924**	968	1203	463	711	688	1123
hopper-mixd	3189	**3289**	2975	3062	**3322**	384	989	**3271**
halfcheetah-mixd	4830	**5578**	**5390**	**5201**	**5964**	**5856**	4463	**5301**
Average	3328	**3930**	3111	3155	3249	2317	2047	3232

Table 3: 1 million time-steps training or gradient steps. The performance are reported as the average return obtained using SLA. "w" means with the SLA metric in our experiments.

The learning curves for different methods are shown in Figure 2. As can be seen, almost all methods perform stable on the simplest task: HalfCheetah, and gain a better or comparable final performance than the BT. On the contrary, all methods perform unstable on the most complicated task: Walker2d, and the over-training issue is most obvious on this task. As can be seen, BCQ generally fails to gain a better performance than BT because it constrains its policy close to the behavior policy. In contrast, less conservative methods such as CQL, URPL or REDQ can achieve a better or comparable final performance than BT on the tasks Hopper-mixed and Halfcheetah-mixed.

In details, Table 2 describe the last iteration policy's performance for all methods including the results taken from D4RL, which are also commonly used offline RL baselines. It is clear that all methods got a poor final policy on the complicated task: Walker2d, while most methods can gain a good performance on the simple task HalfCheetah.

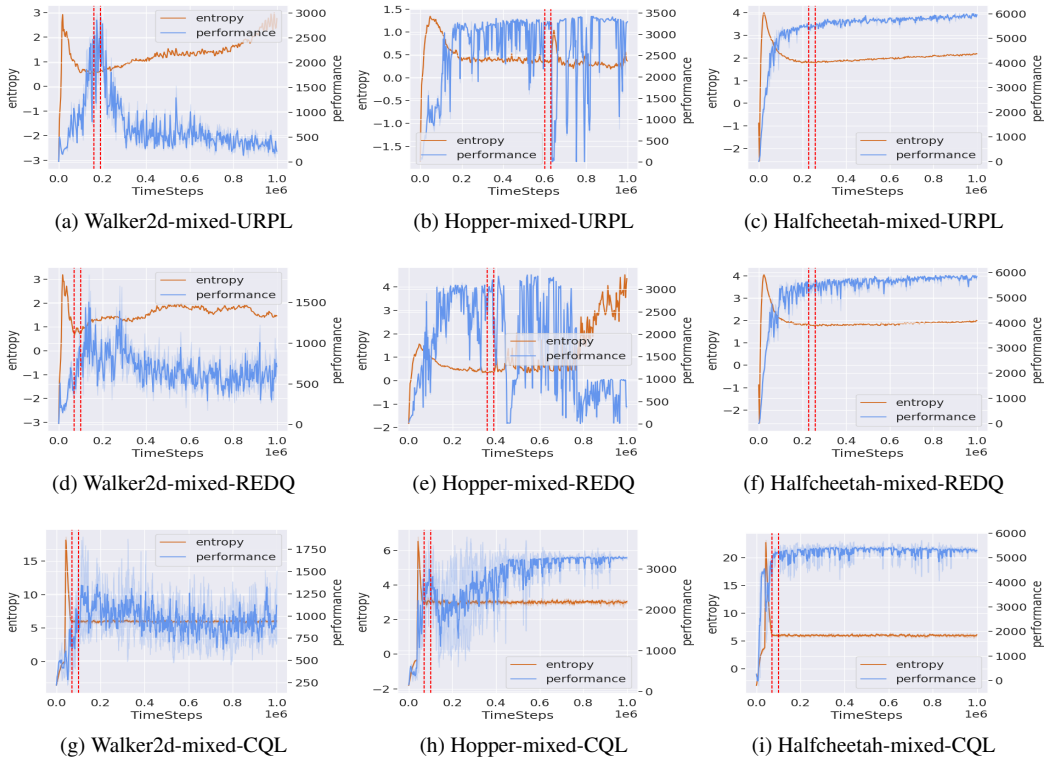
We further conduct two sets of empirical comparisons: one uses policy at the last step of the learning process **without** the SLA metric(methods with the subscript "wo"), and another uses the policy obtained **with** our adaptive policy selection (methods with the subscript "w"). Note that we use the entropy alone as the surrogate metric for the policy selection in this experiment. More comparisons on using the entropy and the variance for policy selection can be found in the next section. The comparison results of "wo" and "w" set are shown in Table 3. BT results are also listed as references. Moreover, if the result of a method beats the BT, we highlight it using bold. We highlight the winner in each task using red.

From Table 3, we can see that URPL$_{wo}$ achieves better results than BT on hopper-mixed and halfcheetah-mixed tasks, but yields much worse results than BT on the most complicated task walker2d-mixed. The results of other baselines, REDQ$_{wo}$, BCQ$_{wo}$ and CQL$_{wo}$, are even more inferior. In average, all methods that without SLA achieves worse results than BT, while URPL$_{wo}$ and CQL$_{wo}$ are comparable. This is due to either the overtraining issue or the conservative update that limits the performance gain. URPL can alleviate these issues in some extent, but still fails in the complicated task.

When using the adaptive policy selection, we can see that almost all the baselines improve their performance compared with the wo version in average. Note that BCQ does not use stochastic policy so that we exclude it. Specifically, we observe a significant improvement of URPL$_w$ on the task walker2d-mixed. URPL$_w$ sweeps BT on all the three tasks, and achieves much better average performance. For the other two baselines, REDQ$_w$ also achieves significant improvement while CQL$_w$ only obtains comparable results. These results verify the effectiveness of our proposed adaptive policy selection strategy, especially on the complicated tasks.

5.3 Stable Learning Area

In this section, we try to verify whether SLA can really find a policy with high performance or not. To do so, we plot both the performance curves and the uncertainty curves, e.g., policy entropy and the

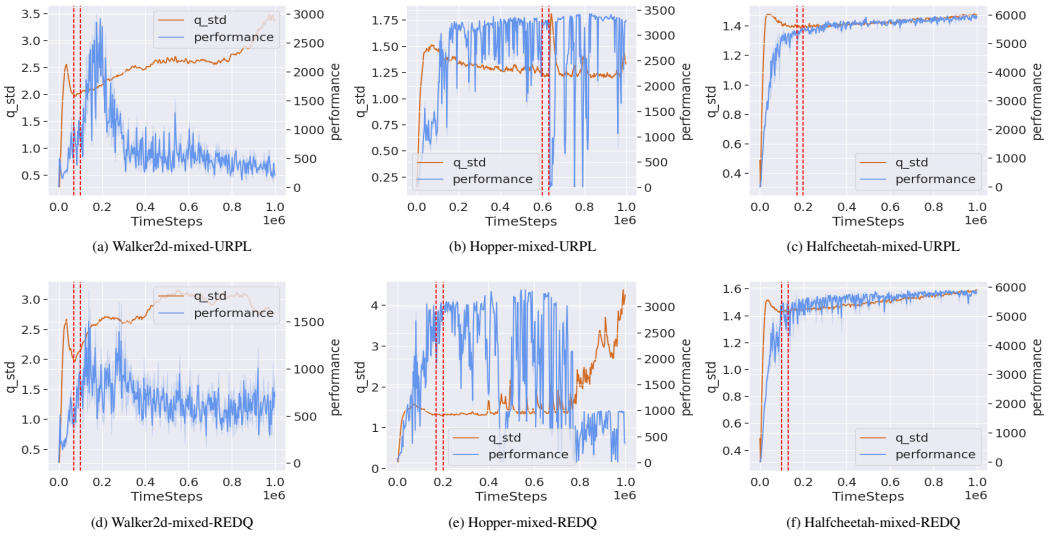


(a) Walker2d-mixed-URPL (b) Hopper-mixed-URPL (c) Halfcheetah-mixed-URPL

(d) Walker2d-mixed-REDQ (e) Hopper-mixed-REDQ (f) Halfcheetah-mixed-REDQ

(g) Walker2d-mixed-CQL (h) Hopper-mixed-CQL (i) Halfcheetah-mixed-CQL

Figure 3: The policy's entropy vs. policy's performance.



(a) Walker2d-mixed-URPL (b) Hopper-mixed-URPL (c) Halfcheetah-mixed-URPL

(d) Walker2d-mixed-REDQ (e) Hopper-mixed-REDQ (f) Halfcheetah-mixed-REDQ

Figure 4: The Q's std/variance vs. policy's performance.

variance of Q values among URPL, REDQ and CQL(BCQ do not use stochastic policy and ensemble Q functions so that we exclude it). We record every time step or update step's uncertainty(the CQL policy's uncertainty is represented by policy's entropy only, as there is no ensemble Q functions for CQL.), and report the average value for every 3000 steps. We test the policy's performance every 1000 steps and report its average value every 3000 steps.

Environment	BT	URPL$_{wE}$	REDQ$_{wE}$	URPL$_{wV}$	REDQ$_{wV}$
walker2d-mixd	1956	**2924**	968	1108	959
hopper-mixd	3189	**3289**	2975	**3267**	2890
halfcheetah-mixd	4830	**5578**	**5390**	**5359**	**5231**
Average	3328	**3930**	3111	3245	3027

Table 4: One million time-steps training or gradient steps. The performance are reported as the average return.

Figures 3 and 4 shows the performance vs. the entropy and the variance, respectively. The red dotted line indicates the final obtained SLA. Herein, we use the simple method to define the SLA, that is, we artificially select the area with the low and stable uncertainty. It is worth noting that the uncertainty in the beginning of the learning has the low uncertainty value, but it is unstable as it surges up rapidly. Thus, we exclude such areas in the selection. Specially, we use the mean performance of all the policies in SLA to represent the performance of SLA. In practice, we can also take the final policy in SLA or the middle policy in SLA as the final policy. The size of SLA area is 3e4 steps that is the same for all tasks.

More specifically, for the simplest task HalfCheetah, we can see all three methods can learn a stable curves. Even the online method REDQ achieves a better performance than the offline method CQL, and the performance grows up slowly and the whole learning process is stable. As the policy's performance grow up slowly, the policy's entropy and variance also grow up slowly. However, for the complicated task, the overtraining issue becomes more serious as the performance curves drops significantly, for instance, in the task Walker2d-mixed. When utilize the SLA, we can see that almost all methods can select a better or comparable policy using much less time steps training.

Furthermore, when compared Figure 3 and Figure 4, we can see that using the entropy can achieve a better SLA than using the variance, especially in the more complicated task Walker2d-mixed. We also list out the final performance obtained by using these two metrics as shown in Table 4. The methods with the subscript "wE" is using the entropy to define SLA, while the methods with the subscript "wV" is using the Qs' variance to define SLA. As can be seen, methods using the entropy for policy selection generally achieve better results than the corresponding alternatives using the variance for policy selection. The suppository is more significant in the first walker2d-mixed task. This suggests us to weight the entropy more as the surrogate metric, especially in the complicated tasks.

It is worth noting that all methods equipped with the SLA (defined by entropy or Q std) gained a better or comparable policy than the original one. That is to say, when we have the SLA, it is effective to avoid the over-training issue, which can save computational resources and achieve a better policy than simply take the policy at the last iteration.

6 Conclusion

In this paper, we aim to solve the overtraining issue when applying off-policy RL to offline learning. Particularly, we focus on a intermediate case where the offline dataset distribution is not sufficient diverse and not identical either. We develop an Uncertainty Regularized Policy Learning (URPL) method that explicitly takes the learning stability into account in the policy learning objective, and then propose an adaptive policy selection strategy that can select a good policy before the overtraining issue happens. Our approach is tested on a range of mixed dataset in D4RL, and show that our method can achieve much better final policy over existing state-of-the-art baselines. Furthermore, we analyze how the performance varies with different surrogate metrics, including the policy's entropy and the variance of Q functions. Experimental results show the superiority of using the policy's entropy as the surrogate metric in the complicated tasks. However, if the data comes from single policy, such as expert policy or medium, our method is not suitable for such tasks. Imitation learning (Rashidinejad et al., 2021) have been proved its efficiency on such tasks. Hence, for such tasks, imitation learning or its variant may be a better alternate choice. In terms of the potential negative societal impacts, even though offline RL can help pre-train a policy, we should still be careful before employing such policy to the real environment, especially for the high-risk environment, such as medical treatment or automatic driving. One single mistake in such environment could lead to serious consequence. One way to alleviate such risks is employing a safety supervisor.

REFERENCES

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, 2020.

K. Azizzadenesheli, Emma Brunskill, and Anima Anandkumar. Efficient exploration through bayesian deep q-networks. *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–9, 2018.

L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *ICML*, 1995.

Marc G. Bellemare, Yavar Naddaf, J. Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents (extended abstract). In *IJCAI*, 2015.

J. Buckman, Danijar Hafner, G. Tucker, E. Brevdo, and H. Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *NeurIPS*, 2018.

Xinyue Chen, Z. Zhou, Z. Wang, Che Wang, Y. Wu, Qing Deng, and K. Ross. Bail: Best-action imitation learning for batch deep reinforcement learning. *ArXiv*, abs/1910.12179, 2020.

Xinyue Chen, Che Wang, Z. Zhou, and K. Ross. Randomized ensembled double q-learning: Learning fast without a model. *ArXiv*, abs/2101.05982, 2021.

Kurtland Chua, R. Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

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

Scott Fujimoto, D. Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *ICML*, 2019.

Carles Gelada and Marc G. Bellemare. Off-policy deep reinforcement learning by bootstrapping the covariate shift. In *AAAI*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Assaf Hallak and Shie Mannor. Consistent on-line off-policy evaluation. In *ICML*, 2017.

Qiang He and X. Hou. Popo: Pessimistic offline policy optimization. *ArXiv*, abs/2012.13682, 2020.

R. Kidambi, A. Rajeswaran, Praneeth Netrapalli, and T. Joachims. Morel : Model-based offline reinforcement learning. *NeurIPS*, abs/2005.05951, 2020.

A. Kumar, Justin Fu, G. Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *NeurIPS*, 2019.

Aviral Kumar, Aurick Zhou, G. Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *ArXiv*, abs/2006.04779, 2020.

Sergey Levine, Aviral Kumar, G. Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643, 2020.

Brendan O'Donoghue, Ian Osband, R. Munos, and V. Mnih. The uncertainty bellman equation and exploration. *ICML*, abs/1709.05380, 2018.

Ian Osband, J. Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *NeurIPS*, 2018.

Doina Precup, R. Sutton, and Satinder Singh. Eligibility traces for off-policy policy evaluation. In *ICML*, 2000.

Doina Precup, R. Sutton, and S. Dasgupta. Off-policy temporal difference learning with function approximation. In *ICML*, 2001.

Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart J. Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *ArXiv*, abs/2103.12021, 2021.

Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, and Martin A. Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *ArXiv*, abs/2002.08396, 2020.

Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2011.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

A. Touati, Harsh Satija, Joshua Romoff, Joelle Pineau, and Pascal Vincent. Randomized value functions via multiplicative normalizing flows. In *UAI*, 2019.

Cameron Voloshin, Hoang M Le, Nan Jiang, and Yisong Yue. Empirical study of off-policy policy evaluation for reinforcement learning. *arXiv preprint arXiv:1911.06854*, 2019.

Alfred Wehrl. General properties of entropy. *Reviews of Modern Physics*, 50(2):221, 1978.

Y. Wu, G. Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *ArXiv*, abs/1911.11361, 2019.

Huazhe Xu, Y. Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based reinforcement learning with theoretical guarantees. *ICLR*, abs/1807.03858, 2019.

Mengjiao Yang, Bo Dai, Ofir Nachum, G. Tucker, and D. Schuurmans. Offline policy selection under uncertainty. *ArXiv*, abs/2012.06919, 2020.

Tianhe Yu, G. Thomas, Lantao Yu, S. Ermon, J. Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *NeurIPS*, abs/2005.13239, 2020.

Ruiyi Zhang, Bo Dai, Lihong Li, and D. Schuurmans. Gendice: Generalized offline estimation of stationary values. In *ICLR*, 2020.