Conservative In-Distribution Q-Learning for Offline Reinforcement Learning

Zhengdao Shao
University of Science and Technology of China
Anhui, China
zhengdaoshao@mail.ustc.edu.cn

Liansheng Zhuang✉
University of Science and Technology of China
Anhui, China
lszhuang@ustc.edu.cn

Jie Yan
Step.ai
Beijing, China
yanjie@step.ai

Liting Chen
McGill University
Canada
98chenliting@gmail.com

***Abstract*—Offline Reinforcement Learning (RL) aims to learn policies from pre-collected datasets without any additional interaction. In order to perform well and robustly in dynamic environments with noise or disturbances, the learned value function and derived policy should generalize well within and near the dataset distribution, rather than 'over-fitting' to training samples. To meet this requirement, we propose a new approach called Conservative In-Distribution Q-learning (CIDQL) that takes a step towards in-distribution offline RL. CIDQL is designed to learn in-distribution with respect to the dataset, using a perturbation-based interpolation technique and a quantile method for value regularization. It prohibits bootstrapping during value iteration, ensuring stable Q-value learning that is separated from policy improvement. The approach has theoretical guarantees for both Q-value underestimation and non-underestimation, and outperforms most SOTA algorithms on D4RL gym-MuJoCo benchmarks.**

***Keywords*—Offline Reinforcement Learning, Conservative Q-Learning, In-Distribution Learning, In-sample Learning, Out-of-distribution Generalization**

I. Introduction

Offline deep reinforcement learning (RL) is a prospective approach for learning a reasonable policy solely from the static dataset of historical experiences [1], [2]. Meanwhile we face an evident but not well-addressed fact that *to perform well and robust in online dynamic environments with noise or disturbance, the learnt value function and derived policy should generalize reasonably well in and near the dataset distribution, rather than 'over-fit' to training samples*. Unlike supervised learning where learning from samples naturally leads to learning from the underlying training distribution, in offline deep RL the generalization of value function cannot be learnt via only training samples because the value of a point depends on other points via Bellman backups.

In-sample offline RL optimizes both value and policy functions through the use of previously collected samples, without requiring active exploration near those samples [3]–[5]. To meet the generalization requirement mentioned above, the value function and policy extraction must be learned in-distribution, meaning that they should be learned from novel data points that are out-of-sample, but still drawn from the same distribution as the training dataset. Unfortunately, it is not well realized in existing offline RL algorithms, largely due to contradictions to their methods (i.e., policy constraints [6], value regularization [7] or in-sample learning [3], [8]) of handling out-of-distribution (OOD) actions (and states) to avoid value overestimation. Value regularization method in CQL learn the Q-value function only on states in the dataset and penalize out-of-dataset actions given the states, resulting in sharp differences between samples and out-of-sample areas even if the later is close to the sample, and thus cannot generalize in the distribution and is too conservative on points near but not in the dataset. COMBO [9], which extends CQL via learning on both the offline dataset and model-based synthetic roll-outs that are mixed in f-interpolation, goes towards learning in the distribution which however heavily depends on assumptions of an extra dynamic model rather than training samples themselves. In-sample learning in IQL (Implicit Q-Learning [3]) avoids to query out-of-sample actions via expectile regression in value function learning and AWR (Advantage Weighted Regression [8]) in policy extraction. However, IQL does not learn on state-action tuples that are out-of-sample, which limits its performance as stated in prior work [10], [11].

In this paper, as one forward step towards in-distribution offline RL, we propose a new approach, namely Conservative In-Distribution Q-learning (CIDQL). Our analysis shows that the Q-values of the out-of-sample state-action tuples under the same or similar distribution as the dataset should neither be ignored as in IQL nor overly underestimated as in CQL; Instead, their values should be smoothly estimated in terms of the distance to training samples. CIDQL applies a perturbation-based interpolation technique to obtain similar values for state-action tuples close to the training samples. Unlike existing conservative algorithms, CIDQL does not allow the agent to bootstrap when iterating over values. This ensures a more stable Q-value learning process that is completely separated from the policy improvement process. Besides, CIDQL also uses the quantile regression method as in in-sample q-learning to regularize the value.

Our main contributions are as follows. First, we identify the necessity of in-distribution learning in offline RL, and propose the new approach CIDQL which features perturbation-based interpolation to achieve conservative in-distribution Q-learning and corresponding in-distribution policy extraction. Second, we present the comprehensive theoretical analysis for

CIDQL, showing guarantees for Q-value underestimation and non-underestimation for points outside and inside the distribution, respectively. Finally, we implement two perturbation methods and experimentally demonstrate the effectiveness of our algorithm, which outperforms other comparable algorithms on gym-MuJoCo D4RL benchmarks [12].

II. Preliminaries and Motivation

A. *Offline Reinforcement Learning*

Suppose a Markov Decision Process (MDP) $M := (\mathcal{S}, \mathcal{A}, T, r, \gamma)$, which is defined by a tuple consisting of a set of states $\mathcal{S}$ and a set of actions $\mathcal{A}$, a transition function T that specifies the probability of transitioning from one state to another given a specific action, a reward function r that determines the reward received by the agent for taking a specific action in a particular state, and a discount factor γ that determines the relative importance of immediate and future rewards. The objective of RL is to learn a policy that maximizes the expected cumulative discounted rewards. Unlike traditional RL that learns online from trial and error, offline RL learns the value function and policy solely through the trajectory dataset and without any online interactions. The Bellman operator is a function projection:

$$\hat{\mathcal{B}}Q(s,a) := r(s,a) + \gamma E_{s' \sim P(\cdot|s,a)}[V(s')]. \tag{1}$$

B. *Motivation of CIDQL*

In statistical and deep learning, the golden rule is that the learnt model should generalize well on unseen samples of the same distribution as training samples. In offline RL, such a generalization is equally important for consistent and robust performance, because during online tests the agent will inevitably reach points that are not in but near the offline dataset. However, to the best of our knowledge, mainstream offline RL algorithms fail to generalize their learnt value and policy functions reasonably with respect to the distribution of the dataset. Conservative value learning algorithms, including CQL [7] and its variants, avoid value overestimation and visitation of the out-of-distribution points via adding explicit value penalization on points out of the dataset, which is overly conservative when it comes to points outside the dataset and results in non-smooth learned Q-values. "In-sample offline RL algorithms, e.g. IQL [3]], train the value and policy functions only on samples in the dataset, without any exploration or even access to out-of-sample points even if they are close to or in the same distribution as the dataset.

Obviously, the optimal approach is to learn the correct Q value within the distribution and conservatively estimate the Q value outside the distribution. This motivates us to develop a new method in this paper, namely Conservative In-Distribution Q-Learning or CIDQL.

C. *Preliminaries for CIDQL*

To develop the CIDQL in the next section, we define in-distribution learning in the context of offline RL, necessary smoothness assumptions on value and policy functions, and perturbation tools.

In-distribution Offline RL. For in-distribution learning through this paper, we refer to the distribution of the training dataset. A dataset of transitions generated from a distribution $P(s,a)$ is used to train a learning-based control system. To estimate this distribution, $\hat{P}(s,a)$ is used instead. Formally, a state-action tuple (s_t, a_t) is considered to be in-distribution if

$$\hat{P}(s_t, a_t) \geq c, \tag{2}$$

where c is the density level threshold ($c > 0$). The set of in-distribution states and actions is denoted as $\hat{\mathcal{D}} := \{(s_t, a_t) | \hat{P}(s_t, a_t) \geq c\}$. With c decreasing towards 0, the number of (s,a) tuples satisfying the in-distribution condition increases monotonically.

Smoothness Assumptions. The theoretical properties of our approach in section III are built upon assumptions about the continuity of the value and policy functions is built upon assumptions about continuity of value and policy functions. For simplicity, we assume that the distribution density function $P(s,a)$ satisfies the Lipschitz continuity on its domain: similar state-action tuples should have similar probability density, and the difference is bounded by the point distance with a Lipschitz constant multiplier. We use a slightly strong definition of Lipschitz continuity as prior work on deep neural network [13]–[15]. The formal definition and its relationship with gradient norm is shown as follows.

Definition 2.1: (Lipschitz continuity and its relationship with gradient norm) Let S $\subset \mathbb{R}^d$ be a convex bounded closed set and let h(x) : S $\rightarrow \mathbb{R}$ be a continuously differentiable function on an open set containing S . Then, h(x) is a Lipschitz continuous function if the following inequality holds for any x,y $\in$ S:

$$|h(x) - h(y)| \leq L_q \left\|x - y\right\|_p, \tag{3}$$

where $L_q = \max\{\left\|\nabla h(x)\right\|_q : x \in S\}$ is the Lipschitz constant, $\nabla h(x)$ is the gradient of h(x), and $\frac{1}{p} + \frac{1}{q} = 1, 1 \leq p, q \leq \infty$.

Note that for most real-world control problems the true value function under optimal policy is continuous and smooth [16], which means imposing constraints of Lipschitz continuity for value and policy function on domains around training samples helps generalization.

Implicit dataset distribution. Suppose $\pi_\beta(a|s)$ be the behavior policy, and $d^{\pi_\beta}(s)$ the discounted marginal state-distribution of $\pi_\beta(a|s)$. The dataset $\mathcal{D}$ is sampled from $\pi_\beta(s,a) = d^{\pi_\beta}(s)\pi_\beta(a|s)$. To generate $\hat{\mathcal{D}_r}$, we sample $\hat{s}, \hat{a}$ from $\mathcal{D}$ and add random noise whose distribution is defined by r. This way, we do not sample any new data besides the original dataset. $\hat{\mathcal{D}_r}$ can be equivalently sampled from an implicit distribution $\mu_r(s,a)$ given by the following equation:

$$\mu_r(\hat{s}, \hat{a}) = \int_{s,a} d^{\pi_\beta}(s)\pi_\beta(a|s)p_r((s,a),(\hat{s},\hat{a}))dsda, \tag{4}$$

where p_r is the perturbation transition probability density from (s,a) to $(\hat{s}, \hat{a})$, and $\int_{s,a} p_r((\overline{s}, \overline{a}),(s,a))dsda = 1$ for any $(\overline{s}, \overline{a})$. For the usual perturbation like Gaussian perturbation, p_r decreases strictly monotonically as the distance between (s,a) and $(\hat{s}, \hat{a})$ increases.

Perturbation. We use perturbation technique to sample in-distribution and out-of-distribution points with respect to the dataset. The perturbation is a symmetric distribution with scale r. For the two perturbations of r_{high} and r_{low}, $r_{high} > r_{low}$, considering points farther away from the dataset, we require r_{high} has a greater probability density than r_{low}. Formally, for any (s, a) tuples 'far from' the given tuple ($\hat{s}$, $\hat{a}$), it satisfies $p_{r_{high}}((s, a), (\hat{s}, \hat{a})) > p_{r_{low}}((s, a), (\hat{s}, \hat{a}))$. Here, 'far from' means that there is a large Euclidean distance between (s,a) and ($\hat{s}$, $\hat{a}$). Common perturbations, such as Gaussian perturbation and uniform perturbation, satisfy this assumption.

In-sample Q-learning vs in-distribution Q-learning. As supervised learning, the training dataset in the offline RL problems can be viewed as samples from the actual distribution. In-sample Offline RL optimizes value and policy functions using previously collected samples without active exploration near samples [3]–[5]. In-distribution Q-learning involves learning from those novel data tuples that are out-of-sample but drawn from the same distribution as the training dataset. Since the distribution of state-action tuples is continuous, tuples that are close to the dataset (e.g., with a low L_2 distance) should have similar sampling probabilities to those in the training sets. Thus, points in proximity to the dataset can also be considered in-distribution. As illustrated in Figure 1, the scope of in-sample is a subset of in-distribution. During training, agents should explore and learn within the in-distribution area while avoiding out-of-distribution (OOD) areas to prevent overestimation.

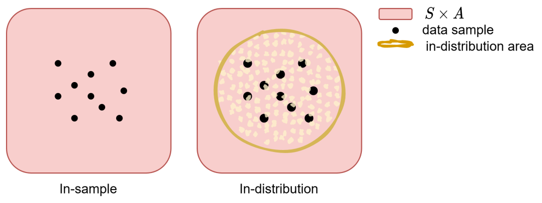

Fig. 1. Learning generalization from in-sample to in-distribution and beyond. The samples are considered to be drawn from a discounted distribution over $S \times A$, determined by the environment transition model, initial state distribution and behavior policy. The boundary between in-distribution and out-of-distribution areas is approximated by Eq. 2.

III. Methodology

We present the method of CIDQL that extends in-sample Q-learning to in-distribution Q-learning. The key idea is to use the perturbation technique to sample out-of-sample but in-distribution state-action tuples and estimate their conservative values with continuity assumptions as regularization. In particular, our approach consists of perturbation-based interpolation for in-distribution Q value estimation, and random perturbation in-distribution exploration during policy extraction.

A. Perturbation-based In-dist. Value Estimation

Our goal is thus to retain the comparatively robust (and accurate when possible) value estimation on samples and generalize with reasonable conservatism to out-of-sample but in-distribution areas. Our key idea is to do the TD error backups in samples only as IQL, but adopt the perturbation-based method to sample and ensure that the Q-valued function satisfies: able to conservatively estimate the Q value of out-of-distribution areas, but accurately estimate the Q value of in-distribution areas. As described in section II , 'in-distribution' and 'out-of-distribution' are implicitly given by the distance to samples in the dataset – it is more likely that the small perturbation added to a sample leads to out-of-sample but in-distribution regions, while the large perturbation leads to out-of-distribution regions.

We first give the basic form of perturbation-based in-distribution conservative Q-learning method which estimates the Q-values of in-sample state-action tuples via usual TD errors while minimizes the expected Q-value of points out of the dataset. The resulting iterative update is thus as follows.

$$\begin{gathered}\hat{Q}^{k+1} \leftarrow \arg\min_{Q} E_{s,a\sim\mathcal{D}}\left[\frac{1}{2}(Q(s,a) - \hat{\mathcal{B}}\hat{Q}^{k})^2\right] + \\ \alpha E_{\hat{s},\hat{a}\sim\mathcal{D}_r}[Q(\hat{s},\hat{a})]\end{gathered} \tag{5}$$

where r is a perturbed distribution over the dataset and $\hat{\mathcal{B}}$ is the empirical Bellman operator. The trade-off factor α is used to balance the usual in-sample learning and out-of-sample conservatism. Its form is similar to CQL [7], but the Q-values learned by our algorithm are estimate of π_β, so there is no need to access the current policy during training. As shown later (Theorem 3.2), the resulting Q-function, $\hat{Q} := \lim_{k\to\infty} \hat{Q}^k$, lower-bounds Q at all (s, a) tuples.

Note that our goal is to learn the value function that generalizes in-distribution with respect to the dataset, with conservatism guarantees. Thus, the in-distribution areas in D_r should be less affected. We achieve this by introducing an additional term of maximizing Q-values near the dataset samples, resulting in the following equation.

$$\begin{gathered}\hat{Q}^{k+1} \leftarrow \arg\min_{Q} E_{s,a\sim\mathcal{D}}\left[\frac{1}{2}(Q(s,a) - \hat{\mathcal{B}}\hat{Q}^{k})^2\right] + \\ \alpha E_{\hat{s}_{high},\hat{a}_{high}\sim\mathcal{D}_{r_{high}},\hat{s}_{low},\hat{a}_{low}\sim\mathcal{D}_{r_{low}}} \\ [Q(\hat{s}_{high},\hat{a}_{high}) - Q(\hat{s}_{low},\hat{a}_{low})]\end{gathered} \tag{6}$$

where r_{low} represents small perturbations over the dataset. As stated later in Theorem 3.1, $(\hat{s}_{low}, \hat{a}_{low}) \sim \mathcal{D}_{r_{low}}$ can be considered an in-distribution tuple. We can prove (Theorem 3.3) that, even if the (s,a) tuples far away from the sample may still be a conservative estimate, the Q values of the in-distribution tuples near the samples are not underestimated. Compared to the Eq. 5, it has a better bound.

Practical implementation. We use a target value to stabilize the Bellman backups, as in the common deep RL practice. Following IQL [3], the target Q values are updated with a soft interpolation factor $\omega \in$ (0, 1), and the state values are updated with a expectile $\tau \in (0, 1)$, as shown in Equation 7, 8, respectively. This makes the algorithm performance more stable.

$$\overline{\hat{Q}^{k+1}} \leftarrow (1-\omega)\overline{\hat{Q}^{k}} + \omega\hat{Q}^{k+1} \tag{7}$$

$$\hat{V}^{k+1} \leftarrow \arg\min_{V} E_{(s,a)\sim\mathcal{D}}\left[L_2^{\tau}(\overline{Q_{\hat{\theta}}^{k}}(s,a) - V^k(s))\right] \quad (8)$$

In Equation 8, $L_2^{\tau}(u) = |\tau - \mathbb{1}(u<0)|u^2$.

Theoretic Analysis. Now we present the theoretic analysis on perturbation-based in-distribution sampling (Theorem 3.1), Eq. 5 lower-bound analysis (Theorem 3.2), and Eq. 6 lower-bound analysis (Theorem 3.3) in order.

We first determine when the perturbed samples should still be considered to be in the distribution of the dataset, under the definition of in-distribution in Eq. 2. As described in Section II, we assume that the system dynamics satisfies the Lipschitz continuity with a Lipschitz constant L_q.

Theorem 3.1: The continuous control problem assumes that the probability distribution $\hat{P}(s,a)$, from which the training set is sampled, is globally Lipschitz over $\mathcal{S}\times\mathcal{A}$, and the Lipschitz constant is L_q. It is also assumed that for each (s,a) tuple in the training set, P(s,a) $\geq c_0 > 0$. To ensure that the tuple of near-samples also falls within this distribution and P$(\hat{s},\hat{a}) \geq c$, where c = $c_0 - d\cdot L_q > 0$, small perturbations r are selected. These perturbations should satisfy $\forall(\hat{s},\hat{a})$ in $\mathcal{D}_r$, L$_\infty((s,a),(\hat{s},\hat{a})) \leq d$ and $d \leq c_0/L_q$.

As Theorem 3.1 suggests, the resulting $(\hat{s},\hat{a})$ tuple remains in-distribution as long as a sufficiently small L_∞-norm is selected for the perturbation. In other words, if the perturbation r_{low} in Equation 6 is small enough, the resulting $(\hat{s}_{low},\hat{a}_{low})$ tuple from the maximized perturbation will still be in-distribution. This prevents overestimation of out-of-distribution (s,a) tuples.

We then examine the impact of our perturbation technique on the Q-value function. Note that Equations 5 and Equations 6 use the Bellman operator, $\hat{\mathcal{B}}$, instead of the actual Bellman operator, $\mathcal{B}$. In contrast to out-of-sample learning, where the policy π changes at each iteration step, in-sample Q-learning employs the same policy at each iteration, which is the implicit policy in the dataset, denoted as π_β. Formally, for all $s,a\in\mathcal{D}$, with probability $\geq 1-\delta$, $|\hat{\mathcal{B}}Q - \mathcal{B}Q|(s,a) \leq \frac{C_{r,T,\delta}}{\sqrt{|\mathcal{D}(s,a)|}}$, where $C_{r,T,\delta}$ is a constant dependent on the variance of r(s, a) and $T(s'|s,a)$, and $\delta\in(0,1)$. Here, $|\mathcal{D}(s,a)|$ is $|\mathcal{S}\times\mathcal{A}|$ counts for each state-action tuple in the dataset.

Theorem 3.2 shows that Equation 5 results in point-wise underestimation of Q-values across the entire state-action space.

Theorem 3.2: For $s\in\mathcal{D}$ and $\hat{\pi}_\beta(a|s) > 0$, supp $\mu_r(a|s) \subset$ supp $\hat{\pi}_\beta$, with probability $\geq 1-\delta$, $\hat{Q}$ (the Q-function obtained by iterating Equation 5) satisfies:

$$\begin{aligned}\hat{Q}(s,a) \leq & Q(s,a) - \alpha\left[(I-\gamma P^{\pi})^{-1}\frac{\mu_r}{\hat{\pi}_\beta}\right](s,a) + \\ & \left[(I-\gamma P^{\pi})^{-1}\frac{C_{r,T,\delta}R_{max}}{(1-\gamma)\sqrt{|\mathcal{D}|}}\right](s,a).\end{aligned} \quad (9)$$

Thus, if α is sufficiently large, then $\hat{Q}(s,a) \leq Q(s,a), \forall s\in\mathcal{S}$, a. When $\hat{\mathcal{B}} = \mathcal{B}$, any $\alpha > 0$ guarantees $\hat{Q}(s,a) \leq \hat{Q}(s,a), \forall(s,a)\in\mathcal{S}\times\mathcal{A}$.

Theorem 3.3 shows that Equation 6 guarantees Q-values for points far from the dataset will be underestimated, whereas in-distribution points in the vicinity of the dataset will not be underestimated.

Theorem 3.3: As Equation 6, if we introduce an additional Q-value maximization term under the small scale perturbation, r_{low}, then $\hat{Q}$ (the Q-function obtained by iterating Equation 6) satisfies:

$$\begin{aligned}\hat{Q}(s,a) \leq Q(s,a) - \alpha & \left[(I-\gamma P^{\pi})^{-1}\frac{\mu_{r_{high}} - \mu_{r_{low}}}{\hat{\pi}_\beta}\right](s,a) + \\ & \left[(I-\gamma P^{\pi})^{-1}\frac{C_{r,T,\delta}R_{max}}{(1-\gamma)\sqrt{|\mathcal{D}|}}\right](s,a).\end{aligned} \quad (10)$$

For (s,a) tuple away from samples, $\mu_{r_{high}} - \mu_{r_{low}} \geq 0$, and $\hat{Q}(s,a) < Q(s,a)$ if α is large enough, with probability $\geq 1-\delta$. For (s,a) tuple near samples, $\mu_{r_{high}} - \mu_{r_{low}} \leq 0$, $\hat{Q}$ will not be underestimated.

Intuitively, it is inefficient to conservatively estimate over the full space. To improve the efficiency, we only need to underestimate the $\hat{Q}$ value of the point at a medium distance from the sample. This can prevent the policy from deviating too far from the sample when exploring. We illustrate this by showing that the upper bound of the $\hat{Q}$ value is the tightest when at a medium distance from the sample, as shown in Proposition 3.4.

Proposition 3.4: It can be demonstrated through the application of a suitable perturbation technique, such as Gaussian perturbation, that the boundary of α exhibits maximal looseness at a medium distance from the sample. In other words, when α is fixed, it can be shown that the tightest upper bound of Q is achieved at a medium distance from the sample.

Thus, if a suitable α is chosen, the basic evaluation in Equation 5 learns a Q-function that lower-bounds the true Q-function Q^{π_β}, and the evaluation in Equation 6 provides accurate Q value estimation for in-distribution points close to the sample. As more data becomes available and $|\mathcal{D}(s,a)|$ increases, the theoretical value of α that is needed to guarantee a lower bound decreases, which indicates that in the limit of infinite data, a lower bound can be obtained by using extremely small values of α.

B. Perturbation-based In-dist. Policy Extraction

Given the learnt conservative in-distribution Q-value function, we can then extract the policy for in-distribution states. In particular, we start from in-sample policy learning with constrained exploration, and extend it to the in-distribution situation.

For in-sample states, it is natural to derive a constrained policy learning algorithm with the following loss function.

$$\begin{aligned}L(\pi) = & L_{SAC}(\pi) + \lambda\mathbf{D}_{KL}(\pi_\beta,\pi) \\ = & E_{s\sim\mathcal{D},a\sim\pi(s)}\left[-Q(\hat{s},a) + \alpha'\log\pi(a|s)\right] - \\ & \lambda E_{s,a\sim\mathcal{D}}[\log\pi(a|s)].\end{aligned} \quad (11)$$

The underlying ideas are straightforward. On one hand, the SAC-like (Soft Actor-Critic) term L_{SAC} learns the policy for in-sample states while encourages the entropy-based action exploration. One the other hand, the KL divergence constraint ensures such exploration be close to the behavior policy π_β of the dataset. Note that RIQL [10] also used Equation 11 for policy extraction, and demonstrated the effectiveness of these ideas.

Now we extend our approach to policy learning on in-distribution states, resulting in the following loss function.

$$\begin{aligned} L(\pi) = & E_{\hat{s}\sim\mathcal{D}_{r_{low}}, a\sim\pi(\hat{s})}\left[-Q(\hat{s},a) + \alpha' \log\pi(a|\hat{s})\right] - \\ & \lambda E_{\hat{s},\hat{a}\sim\mathcal{D}_{r_{low}}}\left[\log\pi(\hat{a}|\hat{s})\right] \end{aligned} \tag{12}$$

where $D_{r_{low}}$ is defined as before and considered to be in-distribution with respect to the dataset. Note that in the second term, both state and action are sampled from the perturbed distribution $D_{r_{low}}$. The perturbation trick used here is also called Random Perturbation Training in the general context of deep learning. It is wildly applied to increase robustness [17], which isotropically smooths the function around each state. This approach is effective because it instructs the actor on how states may appear differently yet remain likely to be observed. In practice, allowing the policy to explore the in-distribution states following $s \in \mathcal{D}_r$ leads to better test performance. Concurrently, this can also be interpreted as a form of exploration around the sample, further capitalizing on the beneficial properties conferred from the critic.

C. Algorithm in Summary

Putting all above together, we propose the algorithm of Conservative In-Distribution Q-Learning in the actor-critic framework, as shown in Alg. 1. As discussed before, our algorithm implements perturbation-based conservative Q value estimation, while enabling the actor to explore in-distribution. The perturbation-based method of CIDQL is primarily used to implement in-distribution learning based on CQL and IQL. This approach leads to improved and more stable experimental results with similar computational complexity.

Algorithm 1 Perturbation-based Impl. of CIDQL

```
Input: Initialized parameters $\phi$ for Value net, $\theta$ for Q net, $\psi$
    for policy net
Output: policy $\pi$
1: while not stop do
2:     update $Q_\theta$ via loss in Equation 6
3:     update target $\overline{Q_\theta}$ via Equation 7
4:     update expectiles value $V_\phi(s)$ via loss in Equation 8.
5:     update policy $\pi_\psi$ via loss in Equation 12
6: end while
```

Note that the policy π does not affect the update of the value function. Therefore, policy extraction can be performed either concurrently with or after TD learning.

IV. Experimental Evaluation

The main goal of this section is to evaluate the effectiveness of our proposed CIDQL algorithm. In particular, we compare the performance of CIDQL with other recent SOTA algorithms, and assess the effect of in-distribution learning components and hyper-parameter choices via ablation study.

A. Experiment Settings

CIDQL Implementation. We implemented two different interpolation-based perturbation methods: Gaussian noise (CIDQL$-\sigma^2$) and uniform perturbation with a given upper limit (CIDQL$-L_\infty$). In this paper, we did not distinguish between the scales of different dimensions of states and actions, the magnitudes of the two kinds of perturbation additions are isotropic. CIDQL code is developed in the JAX library [18]. Like prior work, our models of Q-value, V-value and policy use the same architecture – the neural network with two hidden layers, each having a size of 256x256. The training batch size is set to 256. We employed the double Q-value critic technique to estimate the Q-value. Additionally, we linearly normalized the reward using the smallest and largest trajectory scores within each dataset. To calculate the scores, we uniformly iterated all algorithms for 10 million steps training, then averaged the scores from the last 10 iterations. Each score represents the average of the 10 track scores. This approach effectively reduces the variance caused by different initial states and provides a better reflection of the algorithm's true performance.

Default Hyper-parameters. We chose values for α in Equation 6 from the set $\{1,\ 0.1\}$, and values for λ in Equation 12 from the set $\{1,\ 0\}$. For a fair comparison across all tasks, we employs a consistent perturbation level, although the optimal perturbation amplitude should be chosen according to the dataset distribution. For gym-MuJoCo D4RL tasks, we average mean returns over 10 evaluation trajectories and 5 random seeds. Rewards are standardized by dividing them by the difference between the returns of the best and worst trajectories. We use the Adam optimizer with a learning rate of 3×10^{-4} and a 2-layer MLP with ReLU activations and 256 hidden units for all networks [19]. The policy is parameterized as a Gaussian distribution with a state-independent standard deviation. We chose 0.7 for the expectile τ and 256 for the batch size. In addition, the temperature operator parameter of SAC is 3.0. These hyperparameter settings are consistent with IQL [3] and CQL [7].

This paper introduces three main hyper-parameters: α, which controls the degree of reshaping the Q function, λ, which controls the magnitude of KL divergence constraints, and the method of perturbation interpolation. We found that when both α and λ take a fixed value of 1, the experiment can already achieve good results. Regarding perturbation addition for state-action tuples, we employ two different methods: Gaussian noise with fixed variance and isotropy, and uniform perturbation with a fixed upper limit. The former has a fixed variance, while the latter has a fixed upper limit. Note that the perturbing amplitude should vary with the experimental environment. For the fairness of the comparison, we use the same perturbation amplitude in different experimental environments. We uniformly use r_{low} as 0.001 and r_{high} as 0.01.

Baselines. Our compared baseline algorithms include: (1) TD3+BC [20], a simple TD3 algorithm whose policy update is regularized with Behaviour Cloning; (2) CQL [7] and its model-based variant COMBO [9]; (3) IQL [3]; (4) RIQL [10] which relaxes IQL by enabling regularized action exploration in policy extraction.

We implemented our experiments using the JAX library [18]. In all comparative experiments, we used the original parameters provided in the corresponding papers. In our experiments, we employed the double Q-value critic technique to estimate the Q-value. Additionally, we linearly normalized the reward using the smallest and largest trajectory scores within each dataset. To calculate the scores, we uniformly iterated all algorithms for 10 million steps training, then averaged the scores from the last 10 iterations. Each score represents the average of the 10 track scores. This approach effectively reduces the variance caused by different initial states and provides a better reflection of the algorithm's true performance.

B. Overall Performance

Table I provides a comprehensive performance evaluation of CIDQL and baseline algorithms on gym-mujoco tasks within the D4RL framework. Our algorithm(the $3^{rd}, 4^{th}$ column from the right of the table) outperforms these state-of-the-art baseline methods on average in these environments.The 2^{nd} column on the right corresponds to the experiment in subsection 'CQL on perturbation dataset', and the 1^{st} on the right corresponds to the experiment in subsection 'Experiments on hyper-parameters'.The results are the average score over 5 seeds and the standard deviation is reported in parentheses. For CIDQL, we present the outcomes of two distinct implementations, which employ different perturbation techniques, specifically CIDQL$-\sigma^2$ and CIDQL$-L_\infty$.The results for TD3+BC, COMBO, and IQL are derived from their respective original publications. In the case of CQL, we utilize the findings replicated in [21], as the original paper did not include results pertaining to D4RL v2 datasets. RIQL, a recent model-free approach closely related to our method, is incorporated into a table within the main body of the text for comparative purposes. As shown, our method performs better than other advanced methods on the D4RL dataset. Across 15 environments, our method consistently outperforms other in-sample learning algorithms and surpasses all benchmark algorithms on average. Out of 15 tasks, for the top three scores in seven algorithms (two of which belonged to us), CIDQL-σ^2 obtained 10 and CIDQL-L_∞ obtained 11. Besides, compared to IQL and CQL, our method has even less variability, even their implementation workload and computation cost are similar. In particular, our algorithm's benefits are particularly pronounced on medium-level datasets, which we attribute to the distribution and characteristics of the data. Thus, it is especially worthwhile to explore in-distribution states and actions on a medium-level dataset.

One question is *whether the performance advantage of CIDQL is simply due to the data augmentation by perturbation, rather than the in-distribution learning during training?* We generate the perturbed dataset with r_{low} on datasets, and empirically investigate with one representative baseline CQL on 5 random seeds. The results are shown in the 'CQL on perturbed dataset' column of Table I. As shown, CQL does benefit from the perturbed datasets, which is reasonable since the perturbation-augmented samples help CQL to estimate values better on tuples near the original samples in the dataset; and, to some degree, it supports the positive effect of 'in-distribution' compared to 'in-sample'. However, as a comparison, CIDQL ($-\sigma^2$ and $-L_\infty$) still have significant and almost consistent advantage.

C. Ablation study

We conducted ablation studies on the medium and medium-replay environments. We implemented two independent sets of ablation experiments. The first group verifies the effect of our algorithm on critic and actor changes. The second group verifies the effect of perturbation on state and action. The experimental results of the first group are shown in Table II. The results are the average score over 5 seeds and the standard deviation is reported in parentheses. Only perturb action means $\alpha = 0$ for critic in Equation 6. And only reshape critic means add no perturbation for actor in Equation 11.

The average scores across six environments demonstrate the individual contributions of each module. Our results indicate that both changes have a complementary effect and their combined implementation results in a greater performance boost than either change alone. Concurrently, our empirical findings provide robust corroboration for our assertion that a critic procured through IQL training demonstrates considerable promise when paired with suitable exploration and stability assurance.

In the field of offline RL, OOD actions are usually considered more problematic than OOD states. Our method adds perturbations to both states and actions to alleviate the OOD problem. To demonstrate that interpolation perturbation is effective, we tested two algorithms: one that perturbs only the state, and another that perturbs only the action. The experimental results, shown in Table III, present the average scores across six environments and illustrate the individual contributions of each module. The results are the average score over 5 seeds and the standard deviation is reported in parentheses. Our findings suggest that perturbing only the state leads to actions that lack theoretical guarantees and result in poor performance. Only perturbing the action cannot cooperate well with the perturbation on the critic, and performance improvement is also limited. Therefore, it is necessary to cooperate with perturbations on both the action and the critic in order to achieve the best performance of the algorithm.

D. Direct exploration in IQL

IQL is a popular in-sample learning algorithm that employs expetile for iterative training of the critic's value and AWR [8] for improvement of the actor's strategy. However, the in-sample strategy improvement method of AWR has become a bottleneck for the continuous enhancement of IQL's performance.

TABLE I
PERFORMANCE COMPARISON ON GYM-MUJOCO D4RL CONTROL TASKS V2.

Task Name	TD3+BC	CQL	COMBO	IQL	RIQL	**CIDQL-σ^2**	**CIDQL-L_∞**	CQL on perturbed dataset	CIDQL-σ^2 ($\sigma^2, r_{low} = 0$)
HalfCheetah-r	10.2(1.1)	17.5(1.5)	**38.8**(3.7)	18.2(1.7)	28.4(3.1)	28.0(1.6)	30.7(1.9)	13.9(0.9)	20.4(1.3)
Hopper-r	11.0(0.6)	7.9(0.4)	**17.9**(1.4)	16.3(8.3)	8.9(1.2)	8.7(0.8)	10.2(2.1)	8.1(0.8)	14.2(15.3)
Walker2D-r	1.4(1.7)	5.1(1.3)	7.0(3.6)	5.5(1.2)	7.9(0.5)	7.2(0.2)	**8.4**(1.9)	5.6(3.5)	7.5(6.9)
HalfCheetah-m	42.8(0.2)	47.0(0.5)	54.2(1.5)	47.4(0.1)	55.9(0.3)	**58.5**(1.6)	56.0(5.9)	47.1(0.1)	47.4(0.0)
Hopper-m	**99.5**(2.0)	53.0(28.5)	94.9(2.2)	66.3(3.3)	88.7(4.2)	92.0(0.2)	91.5(3.0)	62.6(2.8)	46.0(2.1)
Walker2D-m	79.7(0.6)	73.3(17.7)	75.5(2.8)	78.3(2.0)	**81.4**(1.0)	80.4(0.5)	81.2(0.5)	80.4(1.5)	75.5(1.9)
HalfCheetah-m-r	43.3(0.5)	45.5(0.7)	**55.1**(1.0)	44.2(0.7)	48.3(0.5)	54.3(0.6)	53.7(0.6)	45.2(0.5)	44.1(0.5)
Hopper-m-r	31.4(18.8)	88.7(12.9)	73.1(1.8)	88.7(14.9)	96.2(7.5)	**100.5**(2.2)	93.0(5.8)	92.8(5.6)	78.3(1.3)
Walker2D-m-r	25.2(5.5)	83.3(2.7)	56.0(8.6)	73.9(7.3)	73.7(4.7)	85.6(4.6)	**89.2**(8.8)	76.7(2.7)	76.8(1.3)
HalfCheetah-m-e	**97.9**(2.3)	75.6(25.7)	90.0(5.6)	86.7(1.3)	92.9(1.2)	94.5(0.3)	93.6(0.6)	90.6(2.6)	89.6(6.6)
Hopper-m-e	**112.2**(0.7)	105.6(12.9)	111.1(2.9)	91.5(27.0)	95.2(5.5)	105.7(3.7)	105.9(2.3)	105.5(2.4)	99.0(11.1)
Walker2D-m-e	101.1(0.2)	107.9(1.6)	96.1(5.6)	**109.6**(5.3)	108.2(0.6)	109.2(0.2)	107.6(1.2)	106.8(4.0)	109.4(0.3)
HalfCheetah-e	**105.7**(2.3)	96.3(1.3)	-	94.6(1.5)	95.1(1.2)	95.1(0.0)	94.9(0.3)	96.9(0.3)	67.5(22.0)
Hopper-e	**112.2**(0.3)	96.5(28.0)	-	109.0(1.8)	110.1(1.7)	108.2(1.6)	110.5(0.6)	107.7(2.2)	109.2(2.4)
Walker2D-e	105.7(0.2)	108.5(0.5)	-	**109.4**(2.3)	108.3(0.5)	107.9(0.3)	108.2(0.0)	108.8(1.6)	108.9(0.6)
Avg. (all)	65.3(2.5)	67.4(9.1)	-	69.7(5.2)	73.3(1.8)	**75.7**(1.2)	**75.6**(2.4)	69.9(2.1)	66.3(5.1)
Avg. (9 medium tasks)	70.3(3.7)	75.5(11.4)	78.4(3.6)	77.0(6.8)	82.2(2.0)	**86.7**(1.5)	**85.7**(3.2)	78.6(1.5)	74.0(2.8)

TABLE II
ABLATION EXPERIMENTS ON ACTOR AND CRITIC PERTURBATIONS.

Task Name	relaxed in-sample	only pertur actor	only reshape critic	**CIDQL-σ^2**
HalfCheetah-medium	55.9(0.3)	55.7(0.5)	55.9(0.5)	**58.5**(1.6)
Hopper-medium	88.7(4.2)	82.4(10.1)	88.4(10.6)	**92.0**(0.2)
Walker2D-medium	**81.4**(1.0)	81.0(1.0)	81.0(0.5)	80.4(0.5)
HalfCheetah-medium-replay	48.3(0.5)	**54.5**(0.9)	53.7(0.8)	54.3(0.6)
Hopper-medium-replay	96.2(7.5)	91.9(3.9)	97.8(4.0)	**100.5**(2.2)
Walker2D-medium-replay	73.3(4.7)	83.5(2.7)	77.4(6.8)	**85.6**(4.6)
Avg of 6 env	73.9(3.0)	74.8(4.3)	75.7(3.9)	**78.5**(1.6)

TABLE III
ABLATION EXPERIMENTS ON STATE AND ACTION PERTURBATIONS.

Task Name	relaxed in-sample	only pertur state	only pertur action	**CIDQL-σ^2**
HalfCheetah-medium	55.9(0.3)	55.5(0.2)	55.7(0.3)	**58.5**(1.6)
Hopper-medium	88.7(4.2)	69.9(4.9)	75.1(17.8)	**92.0**(0.2)
Walker2D-medium	**81.4**(1.0)	80.8(0.6)	81.0(0.5)	80.4(0.5)
HalfCheetah-medium-replay	48.3(0.5)	**54.8**(1.4)	52.5(0.5)	54.3(0.6)
Hopper-medium-replay	96.2(7.5)	82.2(15.9)	97.1(4.2)	**100.5**(2.2)
Walker2D-medium-replay	73.3(4.7)	75.7(9.2)	79.1(5.7)	**85.6**(4.6)
Avg of 6 env	73.9(3.0)	69.8(5.4)	73.4(4.8)	**78.5**(1.6)

To address this issue, we propose a new approach that involves replacing the actor's iterative algorithm with policy iteration outside the sample. This is achieved by utilizing the max-Q operator and the SAC operator for iterations, as shown in Equation 13 and Equation 14, respectively. Importantly, our approach does not change the critic training method. Through experiments, we aim to demonstrate that this new approach may lead to significant performance improvements for IQL.

Unlike the AWR algorithm in IQL, our algorithm allows actors to iterate strategies through bootstrapping, enabling a kind of exploration of meaning. However, as the critic in IQL has not been estimated at tuples other than the sample, this method does not have a theoretical guarantee.

$$L_\pi(\phi) = E_{s\sim\mathcal{D},a\sim\pi(s)}\left[-Q(\hat{s},a)\right]. \tag{13}$$

$$L_\pi(\phi) = E_{s\sim\mathcal{D},a\sim\pi(s)}\left[-Q(\hat{s},a) + \alpha' \log \pi(a|s)\right]. \tag{14}$$

We replaced the actors in IQL with maxQ and SAC to achieve out-of-sample exploration, resulting in two algorithms:

- MaxQ-IQL: This algorithm uses max-Q actor to directly maximize the payoff for the critic of IQL.
- SAC-IQL: This algorithm uses SAC actor to directly maximize the payoff for the critic of IQL.

The experimental results, presented in Table IV, demonstrate that direct exploration can improve performance in some environments. However, in other environments, exploration may invalidate the algorithm's strategy entirely.

The MaxQ-IQL and SAC-IQL algorithms demonstrate stable performance on medium-replay tasks. However, their performance becomes unstable on some challenging problems due to the lack of exploration protection. Our results indicate that IQL with exploration can sometimes significantly outperform IQL, as evidenced by the performance of SAC-IQL on HalfCheetah-medium-v2. This suggests that the inability to train out-of-sample hinders the further improvement of in-sample Q-learning performance. In other words, direct exploration by the in-sample Q-learning algorithm can be beneficial when the dataset meets certain ideal conditions. However, such conditions are often difficult to satisfy in practice, and there is no theoretical guarantee for direct exploration with IQL. As a result, exploration outcomes can be unpredictable. Once the effectiveness of exploration has been established, the key challenge becomes ensuring stable exploration by the actor. This can be achieved by reshaping the Q function to provide stability guarantees for the algorithm.

TABLE IV
ABLATION OF OUT-OF-SAMPLE EXPLORATION.

Task Name	IQL	maxQ-IQL	SAC-IQL
HalfCheetah-m	47.4(0.1)	55.3(1.7)	57.0(2.8)
Hopper-m	66.3(3.3)	2.6(1.5)	3.6(1.7)
Walker2D-m	78.3(2.0)	0.7(2.4)	-0.3(0.0)
HalfCheetah-m-r	44.2(0.7)	54.5(0.9)	53.6(0.8)
Hopper-m-r	66.3(14.9)	70.7(30.5)	100.4(1.6)
Walker2D-m-r	78.3(7.3)	74.6(20.4)	90.5(8.9)

E. Performance on Antmaze

These D4RL tasks require merging segments of suboptimal trajectories to generate policies that are more optimal in achieving goals on a Mujoco Ant robot. For Antmaze tasks, we set τ=0.9 and apply the relevant preprocessing to the dataset [12]. The detailed results can be found in Table V and the results of CIDQL are the average score over 3 seeds.

TABLE V
PERFORMANCE COMPARISON ON ANTMAZE.

Task Name	CQL	IQL	CIDQL(σ^2)
umaze	84.4	85.5	91.0
umaze-diverse	43.4	66.7	61.9
medium-replay	65.2	72.2	66.8
medium-diverse	54.0	71.0	67.5

As shown, CIDQL can also outperform CQL consistently, while behaves slightly lower than IQL in 3 out of 4 tasks. The data of Antmaze tasks are generated via a goal reaching policy, which makes the strict in-sample learning algorithm IQL perform better.

V. CONCLUSION

In this paper, we present CIDQL, a new approach for in-distribution offline reinforcement learning. CIDQL uses perturbation-based interpolation to accurately estimate Q-values for state-action pairs that are similar to data samples. By prohibiting bootstrapping during value iteration, CIDQL ensures stable Q-value learning that is separate from policy improvement. Additionally, CIDQL has theoretical guarantees for both Q-value underestimation and non-underestimation, offering robust exploration capabilities and leveraging in-distribution value estimation. Our experimental results on gym-MuJoCo D4RL benchmarks demonstrate superior performance compared to benchmark algorithms, suggesting that CIDQL has the potential to advance future research in the field of offline RL.

ACKNOWLEDGE

This work was supported by National Natural Science Foundation of China (NSFC) under contract No.U20B2070 and No.61976199.

REFERENCES

[1] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.
[2] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.
[3] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2021.
[4] Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline RL with no OOD actions: In-sample learning via implicit value regularization. In *The Eleventh International Conference on Learning Representations*, 2023.
[5] Jiafei Lyu, Aicheng Gong, Le Wan, Zongqing Lu, and Xiu Li. State advantage weighting for offline rl, 2022.
[6] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.
[7] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *CoRR*, abs/2006.04779, 2020.
[8] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage weighted regression: Simple and scalable off-policy reinforcement learning, 2020.
[9] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.
[10] Yuwei Fu, Di Wu, and Benoit Boulet. A closer look at offline rl agents. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 8591–8604. Curran Associates, Inc., 2022.
[11] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *ArXiv*, abs/2304.10573, 2023.
[12] 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.
[13] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J. Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks, 2023.
[14] Fabian Latorre, Paul Rolland, and Volkan Cevher. Lipschitz constant estimation of neural networks via sparse polynomial optimization, 2020.
[15] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach, 2018.
[16] Kavosh Asadi, Dipendra Misra, and Michael L. Littman. Lipschitz continuity in model-based reinforcement learning, 2018.
[17] Rui Yang, Chenjia Bai, Xiaoteng Ma, Zhaoran Wang, Chongjie Zhang, and Lei Han. Rorl: Robust offline reinforcement learning via conservative smoothing. *arXiv preprint arXiv:2206.02829*, 2022.
[18] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.
[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
[20] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.
[21] Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhi-Hong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. In *International Conference on Learning Representations*, 2021.