OFFLINE REINFORCEMENT LEARNING WITH CLOSED-FORM POLICY IMPROVEMENT OPERATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Behavior constrained policy optimization has been demonstrated to be a successful paradigm for tackling Offline Reinforcement Learning. By exploiting historical transitions, a policy is trained to maximize a learned value function while constrained by the behavior policy to avoid a significant distributional shift. In this paper, we propose our closed-form policy improvement operators. We make a novel observation that the behavior constraint naturally motivates the use of first-order Taylor approximation, leading to a linear approximation of the policy objective. Additionally, as practical datasets are usually collected by heterogeneous policies, we model the behavior policies as a Gaussian Mixture and overcome the induced optimization difficulties by leveraging the LogSumExp's lower bound and Jensen's Inequality, giving rise to a closed-form policy improvement operator. We instantiate offline RL algorithms with our novel policy improvement operators and empirically demonstrate their effectiveness over state-of-the-art algorithms on the standard D4RL benchmark.

1 INTRODUCTION

Deploying Reinforcement Learning (RL) (Sutton & Barto, 2018) in the real world is hindered by its massive demand for online data. In domains such as robotics (Cabi et al., 2019) and autonomous driving (Sallab et al., 2017), rolling out a premature policy is prohibitively costly and unsafe. To address this issue, offline RL (a.k.a batch RL) (Levine et al., 2020; Lange et al., 2012) has been proposed to learn a policy directly from historical data without environment interaction. However, learning competent policies from a static dataset is challenging. Prior studies have shown that learning a policy without constraining its deviation from the data-generating policies suffers from significant extrapolation errors, leading to training divergence (Fujimoto et al., 2019; Kumar et al., 2019).

Current literature has demonstrated two successful paradigms for managing the trade-off between policy improvement and limiting the distributional shift from the behavior policies. Under the actor-critic framework (Konda & Tsitsiklis, 1999), behavior constrained policy optimization (BCPO) (Fujimoto et al., 2019; Kumar et al., 2019; Fujimoto & Gu, 2021; Wu et al., 2019; Brandfonbrener et al., 2021; Ghasemipour et al., 2021) explicitly regularizes the divergence between learned and behavior policies, while conservative methods (Kumar et al., 2020b; Bai et al., 2022; Yu et al., 2020; 2021) penalize the value estimate for out-of-distribution (OOD) actions to avoid overestimation error. However, most existing model-free offline RL algorithms still require learning off-policy value functions and a target policy through stochastic gradient descent (SGD). Unlike supervised learning, off-policy learning with non-linear function approximators and temporal difference learning (Sutton & Barto, 2018) is notoriously unstable (Kumar et al., 2020a; Mnih et al., 2015; Henderson et al., 2018; Konda & Tsitsiklis, 1999; Watkins & Dayan, 1992) due to the existence of the deadly-triad (Sutton & Barto, 2018; Van Hasselt et al., 2018). The performance can exhibit significant variance even across different random seeds (Islam et al., 2017). In offline settings, learning becomes even more problematic as environment interaction is restricted, thus preventing the learning from receiving corrective feedback (Kumar et al., 2020a). Consequently, training stability poses a major challenge in offline RL. Although some current approaches (Brandfonbrener et al., 2021) circumvent the requirement for learning an off-policy value function, they still require learning a policy via SGD.

Can we mitigate the issue of learning instability by leveraging optimization techniques? In this paper, we approach this issue from the policy learning perspective, aiming to design a stable policy

improvement operator. We take a closer look at the BCPO paradigm and make a novel observation that the requirement of limited distributional shift motivates the use of the first-order Taylor approximation (Callahan, 2010), leading to a linear approximation of the policy objective that is accurate in a sufficiently small neighborhood of the behavior action. Based on this crucial insight, we construct our policy improvement operators that return closed-form solutions by carefully designing a tractable behavior constraint. When modeling the behavior policies as a Single Gaussian, our policy improvement operator deterministically shifts the behavior policy towards a value improving direction derived by solving a Quadratically Constrained Linear Program (QCLP) in closed form. Therefore, our method only requires learning the underlying behavior policies of a given dataset with supervised learning, avoiding the training instability from policy improvement.

Furthermore, we note that practical datasets are likely to be collected by heterogeneous policies, which may give rise to a multimodal behavior action distribution. In this scenario, a Single Gaussian will fail to capture the entire picture of the underlying distribution, limiting the potential of policy improvement. While modeling the behavior as a Gaussian Mixture provides better expressiveness, it incurs extra optimization difficulties due to the non-concavity of its log-likelihood function. We tackle this issue by leveraging the LogSumExp's lower bound and Jensen's inequality, leading to a closed-form policy improvement (CFPI) operator compatible with a multimodal behavior policy. Empirically, we demonstrate the effectiveness of Gaussian Mixture over the conventional Single Gaussian when the underlying distribution comes from hetereogenous policies.

In this paper, we empirically demonstrate that our CFPI operators can instantiate successful offline RL algorithms in a one-step or iterative fashion. Moreover, our methods can also be leveraged to improve a policy learned by the other algorithms. In summary, our main contributions are threefold:

- CFPI operators compatible with single mode and multimodal behavior policies.
- An empirical demonstration of the benefit to model the behavior policy as a Gaussian Mixture in model-free offline RL. To the best of our knowledge, we are the first to do this.
- One-step and iterative instantiations of our algorithm, which outperform state-of-the-art (SOTA) algorithms on the standard D4RL benchmark (Fu et al., 2020).

2 Preliminaries

Reinforcement Learning. RL aims to maximize returns in a Markov Decision Process (MDP) (Sutton & Barto, 2018) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, T, \rho_0, \gamma)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function R, transition function T, initial state distribution ρ_0, and discount factor $\gamma \in [0, 1)$. At each time step t, the agent starts from a state $s_t \in \mathcal{S}$, selects an action $a_t \sim \pi(\cdot|s_t)$ from its policy π, transitions to a new state $s_{t+1} \sim T(\cdot|s_t, a_t)$, and receives reward $r_t := R(s_t, a_t)$. The goal of an RL agent is to learn an optimal policy π^* that maximizes the expected discounted cumulative reward $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$ without access to the ground truth R and T. We define the action value function associated with π by $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a]$. The RL objective can then be reformulated as

$$\pi^* = \arg\max_\pi J(\pi) := \mathbb{E}_{s \in \rho_0, a \in \pi(\cdot|s)}[Q^\pi(s, a)] \tag{1}$$

In this paper, we consider *offline* RL settings, where we assume restricted access to the MDP $\mathcal{M}$, and a previously collected dataset $\mathcal{D}$ with N transition tuples $\{(s_t^i, a_t^i, r_t^i)\}_{i=1}^N$. We denote the underlying policy that generates $\mathcal{D}$ as π_β, which may or may not be a mixture of individual policies.

Behavior Constrained Policy Optimization. One of the critical challenges in offline RL is that the learned Q function tends to assign spuriously high values to OOD actions due to extrapolation error, which is well documented in previous literature (Fujimoto et al., 2019; Kumar et al., 2019). Behavior Constrained Policy Optimization (BCPO) methods (Fujimoto et al., 2019; Kumar et al., 2019; Fujimoto & Gu, 2021; Wu et al., 2019; Brandfonbrener et al., 2021) explicitly constrain the action selection of the learned policy to stay close to the behavior policy π_β, resulting in a policy improvement step that can be generally summarized by the optimization problem below:

$$\max_\pi \mathbb{E}_{s \sim \mathcal{D}} \left[\mathbb{E}_{\tilde{a} \sim \pi(\cdot|s)} \left[Q\left(s, \tilde{a}\right) \right] - \alpha D\left(\pi(\cdot \mid s), \pi_\beta(\cdot \mid s)\right) \right], \tag{2}$$

where $D(\cdot, \cdot)$ is a divergence function that calculates the divergence between two action distributions, and α is a hyper-parameter controlling the strength of regularization. Consequently, the policy is optimized to maximize the Q-value while staying close to the behavior distribution.

Different algorithms may choose different $D(\cdot,\cdot)$ (e.g., KL Divergence (Wu et al., 2019; Jaques et al., 2019), MSE (Fujimoto & Gu, 2021) and MMD (Kumar et al., 2019)). However, to the best of our knowledge, all existing methods tackle this optimization via SGD. In this paper, we take advantage of the regularization and solve the problem in closed form.

3 Closed-Form Policy Improvement

In this section, we introduce our policy improvement operators that map the behavior policy to a higher-valued policy, which is accomplished by solving a linearly approximated BCPO. We show that modeling the behavior policy as a Single Gaussian transforms the approximated BCPO into a QCLP and thus can be solved in closed-form (Sec. 3.1). Given that practical datasets are usually collected by heterogeneous policies, we generalize the results by modeling the behavior policies as a Gaussian Mixture to facilitate expressiveness and overcome the incurred optimization difficulties by leveraging the LogSumExp's lower bound (LB) and Jensen's Inequality (Sec. 3.2). We close this section by presenting an offline RL paradigm that leverages our policy improvement operators (Sec. 3.3).

3.1 Approximated behavior constrained optimization

We aim to design a learning-free policy improvement operator to avoid learning instability in offline settings. We observe that optimizing towards BCPO's policy objective (2) induces a policy that admits limited deviation from the behavior policy. Consequently, it will only query the Q-value within the neighborhood of the behavior action during training, which naturally motivates the employment of the first-order Taylor approximation to derive the following linear approximation of the Q function

$$\begin{aligned}\bar{Q}(s,a;a_\beta) &= (a-a_\beta)^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta} + Q(s,a_\beta) \\ &= a^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta} + \text{const.}\end{aligned} \tag{3}$$

By Taylor's theorem (Callahan, 2010), $\bar{Q}(s,a;a_\beta)$ only provides an accurate linear approximation of $Q(s,a)$ in a sufficiently small neighborhood of a_β. Therefore, the choice of a_β is critical.

Recognizing (2) as a Lagrangian and with the linear approximation (3), we propose to solve the following surrogate problem of (2) given any state s:

$$\max_{\pi} \mathop{\mathbb{E}}_{\tilde{a}\sim\pi}\left[\tilde{a}^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta}\right], \quad \text{s.t.} \quad D\left(\pi(\cdot \mid s), \pi_\beta(\cdot \mid s)\right) \le \delta. \tag{4}$$

Note that it is not necessary for $D(\cdot,\cdot)$ to be a (mathematically defined) divergence measure since any generic $\mathcal{D}(\cdot,\cdot)$ that can constrain the deviation of π's action from π_β can be considered.

Single Gaussian Behavior Policy. In general, (4) does not always have a closed-form solution. We analyze a special case where $\pi_\beta = \mathcal{N}(\mu_\beta, \Sigma_\beta)$ is a Gaussian policy, $\pi = \mu$ is a deterministic policy, and $D(\cdot,\cdot)$ is a negative log-likelihood function. In this scenario, a reasonable choice of μ should concentrate around μ_β to limit distributional shift. Therefore, we set $a_\beta = \mu_\beta$ and the optimization problem (4) becomes the following:

$$\max_{\mu} \quad \mu^T \left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}, \quad \text{s.t.} \quad -\log \pi_\beta(\mu|s) \le \delta \tag{5}$$

We now show that (5) has a closed-form solution.

Proposition 3.1. *The optimization problem* (5) *has a closed-form solution that is given by*

$$\mu_{sg}(\tau) = \mu_\beta + \frac{\sqrt{2\log\tau}\,\Sigma_\beta \left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}}{\sqrt{\left[\nabla_a Q(s,a)\right]^T_{a=\mu_\beta} \Sigma_\beta \left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}}}, \quad \textit{where} \;\; \delta = \frac{1}{2}\log\det(2\pi\Sigma_\beta) + \log\tau \tag{6}$$

Proof sketch. (5) can be converted into the QCLP below that has a closed-form solution given by (6) (Boyd et al., 2004). A fulll proof is given in the Appendix A.1.

$$\max_{\mu} \quad \mu^T \left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}, \quad \text{s.t.} \quad \frac{1}{2}(\mu-\mu_\beta)^T \Sigma_\beta^{-1}(\mu-\mu_\beta) \le \delta - \frac{1}{2}\log\det(2\pi\Sigma_\beta) \tag{7}$$

□

Although we still have to tune τ as tuning α in (2) for conventional BCPO methods, we have a transparent interpretation of τ's effect on the action selection thanks to the tractability of (5). Due to the KKT conditions (Boyd et al., 2004), (6) always returns an action μ_{sg} with the following property

$$-\log \pi_\beta(\mu|s) = \delta = -\log \frac{1}{\tau}\pi_\beta(\mu_\beta|s) \quad \Longleftrightarrow \quad \pi_\beta(\mu_{\text{sg}}|s) = \frac{1}{\tau}\pi_\beta(\mu_\beta|s) \tag{8}$$

While setting $\tau = 1$ will always return the mean of π_β, a large τ might send μ_{sg} out of the support of π_β, breaking the accuracy guarantee of the first-order Taylor approximation.

3.2 Gaussian Mixture as a more expressive model

Performing policy improvement with (6) enjoys favorable computational efficiency and avoids the potential instability caused by SGD. However, its tractability relies on the Single Gaussian assumption of the behavior policy π_β. In practice, the historical datasets are usually collected by heterogeneous policies with different levels of expertise. A Single Gaussian may fail to capture the whole picture of the underlying distribution, motivating the use of a Gaussian Mixture to represent π_β.

$$\pi_\beta = \sum_{i=1}^{N} \lambda_i \mathcal{N}(\mu_i, \Sigma_i), \quad \sum_{i=1}^{N} \lambda_i = 1 \tag{9}$$

However, directly plugging the Gaussian Mixture π_β into (5) breaks its tractability, resulting in a non-convex optimization below

$$\begin{aligned} \max_{\mu} \quad & \mu^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta}, \\ \text{s.t.} \quad & \log \sum_{i=1}^{N} \left(\lambda_i \det(2\pi\Sigma_i)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1} (\mu-\mu_i) \right) \right) \geq -\delta \end{aligned} \tag{10}$$

We are confronted with two major challenges to solve the optimization problem (10). First, it is unclear how to choose a proper a_β while we need to ensure that the solution μ lies within a small neighborhood of a_β. Second, the constraint of (10) does not admit a convex form, posing non-trivial optimization difficulties. We leverage the lemma below to tackle the non-convexity of the constraint.

Lemma 3.1. $\log \sum_{i=1}^{N} \lambda_i \exp(x_i)$ *admits the following inequality:*

1. *(LogSumExp's LB)* $\log \sum_{i=1}^{N} \lambda_i \exp(x_i) \geq \max_i \{x_i + \log \lambda_i\}$
2. *(Jensen's Inequality)* $\log \sum_{i=1}^{N} \lambda_i \exp(x_i) \geq \sum_{i=1}^{N} \lambda_i x_i$

Next, we show that applying each inequality in Lemma 3.1 to the constraint of (10) respectively resolves the intractability and leads to natural choices of a_β.

Proposition 3.2. *By applying the first inequality of Lemma 3.1 to the constraint of* (10)*, we can derive an optimization problem that lower bounds* (10)

$$\begin{aligned} \max_{\mu} \quad & \mu^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta}, \\ \textit{s.t.} \quad & \max_i \left\{ -\frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1} (\mu-\mu_i) - \frac{1}{2}\log\det(2\pi\Sigma_i) + \log\lambda_i \right\} \geq -\delta, \end{aligned} \tag{11}$$

and the closed-form solution to (11) *is given by*

$$\begin{aligned} \mu_{lse}(\tau) = & \arg\max_{\bar{\mu}_i(\delta)} \ \bar{\mu}_i^T \left[\nabla_a Q(s,a)\right]_{a=\mu_i}, \ \textit{s.t.} \ \delta = \min_i \{ \frac{1}{2}\log\det(2\pi\Sigma_i) - \log\lambda_i \} + \log\tau \\ \textit{where} \quad & \bar{\mu}_i(\delta) = \mu_i + \sqrt{\frac{2(\delta + \log\lambda_i) - \log\det(2\pi\Sigma_i)}{\left[\nabla_a Q(s,a)\right]_{a=\mu_i}^T \Sigma_i \left[\nabla_a Q(s,a)\right]_{a=\mu_i}}} \Sigma_i \left[\nabla_a Q(s,a)\right]_{a=\mu_i} \end{aligned} \tag{12}$$

Proposition 3.3. *By applying the second inequality of Lemma 3.1 to the constraint of* (10)*, we can derive an optimization problem that lower bounds* (10)

$$\max_{\mu} \ \mu^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta}, \ \textit{s.t.} \quad \sum_{i=1}^{N} \lambda_i \left(-\frac{1}{2}\log\det(2\pi\Sigma_i) - \frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1} (\mu-\mu_i) \right) \geq -\delta \tag{13}$$

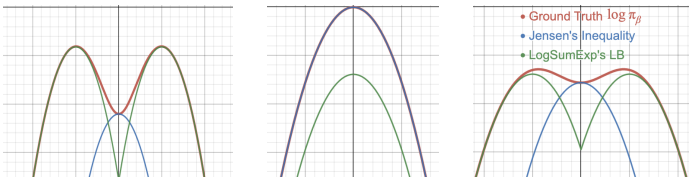


Figure 1: Apply Lemma 3.1 to Gaussian Mixture's log probability $\log \pi_\beta$ at different scenarios. (L) $\log \pi_\beta$ has multiple modes. LogSumExp's LB preserves multimodality. (M) $\log \pi_\beta$ reduces to Single Gaussian. Jensen's inequality becomes equality. (R) $\log \pi_\beta$ is similar to a uniform distribution.

and the closed-form solution to (13) *is given by*

$$\mu_{jensen}(\tau) = \bar{\mu} + \sqrt{\frac{2\log\tau - \sum_{i=1}^{N}\lambda_i \mu_i^T \Sigma_i^{-1}\mu_i + \bar{\mu}^T \bar{\Sigma}^{-1}\bar{\mu}}{[\nabla_a Q(s,a)]_{a=\bar{\mu}}^T \bar{\Sigma}[\nabla_a Q(s,a)]_{a=\bar{\mu}}}}\ \bar{\Sigma}[\nabla_a Q(s,a)]_{a=\bar{\mu}},$$
$$\textit{where } \bar{\Sigma} = \left(\sum_{i=1}^{N}\lambda_i \Sigma_i^{-1}\right)^{-1}, \quad \bar{\mu} = \bar{\Sigma}\left(\sum_{i=1}^{N}\lambda_i \Sigma_i^{-1}\mu_i\right), \quad \delta = \log\tau + \frac{1}{2}\sum_{i=1}^{N}\lambda_i \log\det(2\pi\Sigma_i) \tag{14}$$

We defer the detailed proof of Proposition 3.2 and Proposition 3.3 as well as how we choose a_β for each optimization problem to Appendix A.2 and A.3, respectively.

Indeed, these two optimization problems have their own assets and liabilities. When π_β exhibits an obvious multimodality as is shown in Fig. 1 (L), the lower bound of $\log \pi_\beta$ constructed by Jensen's Inequality cannot capture different modes due to its concavity, losing the advantage of modeling π_β as a Gaussian Mixture. In this case, the optimization problem (11) can serve as a reasonable surrogate problem of (10), as LogSumExp's LB still preserves the multimodality of $\log \pi_\beta$.

When π_β is reduced to a Single Gaussian, the approximation with the Jensen's Inequality becomes equality as is shown in Fig. 1 (M). Thus μ_{jensen} returned by (14) exactly solves the optimization problem (10). However, in this case, the tightness of LogSumExp's LB largely depends on the weights $\lambda_{i=1\ldots N}$. If each Gaussian component is distributed and weighted identically, the lower bound will be $\log N$ lower than the actual value. Moreover, there also exists the scenario (Fig. 1 (R)) when both (11) and (13) can serve as reasonable surrogates to the original problem (10).

Fortunately, we can combine the best of both worlds and design a policy improvement operator accounting for all the above scenarios. As both Proposition 3.2 and 3.3 have closed-form solutions, the operator returns a policy that selects the higher-valued action from μ_{lse} and μ_{jensen}

$$\mu_{\text{mg}}(\tau) = \arg\max_{\mu} Q(s,\mu), \quad \text{s.t.} \quad \mu \in \{\mu_{\text{lse}}(\tau), \mu_{\text{jensen}}(\tau)\} \tag{15}$$

3.3 Algorithm Template

We have derived two CFPI operators that map the behavior policy to a higher-valued policy. When the behavior policy π_β is a Single Gaussian, $\mathcal{I}_{\text{SG}}(\pi_\beta, Q; \tau)$ returns a policy with action selected by (6). When π_β is a Gaussian Mixture, $\mathcal{I}_{\text{MG}}(\pi_\beta, Q; \tau)$ returns a policy with action selected by (15). We note that our methods can also work with a non-Gaussian π_β. Appendix D provides the derivations for the corresponding CFPI operators when π_β is modeled as both a deterministic policy and VAE. Algorithm 1 shows that our CFPI operators enable the design of a general offline RL template that can yield one-step, multi-step and iterative methods, where $\mathcal{E}$ is a general policy evaluation operator that returns a value function $\hat{Q}_t$. When setting $T = 0$, we obtain our one-step method. We defer the discussion on multi-step and iterative methods to the Appendix C.

While the design of our CFPI operators is motivated from the behavior constraint, we highlight that they are compatible with general baseline policies π_b besides π_β. Sec. 5.2 and Appendix G.7 show that our CFPI operators can improve policies learned by IQL and CQL (Kumar et al., 2020b).

3.4 Theoretical Guarantees for Closed-Form Policy Improvement

At a high level, Algorithm 1 follows the *approximate policy iteration* (API) (Perkins & Precup, 2002) by iterating over the policy evaluation ($\mathcal{E}$ step, Line 4) and policy improvement ($\mathcal{I}$ step, Line

Algorithm 1 Offline RL with closed-form policy improvement operators (CFPI)

Input: Dataset $\mathcal{D}$, baseline policy $\hat{\pi}_b$, value function $\hat{Q}_{-1}$, HP τ

1: Warm start $\hat{Q}_0 = \text{SARSA}(\hat{Q}_{-1}, \mathcal{D})$ with the SARSA-style algorithm (Sutton & Barto, 2018)
2: Get one-step policy $\hat{\pi}_1 = \mathcal{I}(\hat{\pi}_b, \hat{Q}_0; \tau)$
3: **for** $t = 1 \dots T$ **do**
4: Policy evaluation: $\hat{Q}_t = \mathcal{E}(\hat{Q}_{t-1}, \hat{\pi}_t, \mathcal{D})$
5: Get policy: $\hat{\pi}_{t+1} = \mathcal{I}(\hat{\pi}_b, \hat{Q}_t; \tau)$ (concrete choices of $\mathcal{I}$ includes $\mathcal{I}_{\text{MG}}$ and $\mathcal{I}_{\text{SG}}$)
6: **end for**

5). Therefore, to verify $\mathcal{E}$ provides the improvement, we need to first show policy evaluation $\hat{Q}_t$ is accurate. We employ the Fitted Q-Iteration (Sutton & Barto, 2018) to perform policy evaluation, which is known to be statistically efficient (*e.g.* (Chen & Jiang, 2019)) under the mild condition for the function approximation class. Next, for the performance gap between $J(\hat{\pi}_{t+1}) - J(\hat{\pi}_t)$, we apply the standard performance difference lemma (Kakade & Langford, 2002; Kakade, 2003).

Theorem 3.1. *[Safe Policy Improvement] Assume the state and action spaces are discrete.*[1] *Let $\hat{\pi}_1$ be the policy obtained after the CFPI update (Line 2 of Algorithm 1). Then with probability $1-\delta$,*

$$J(\hat{\pi}_1) - J(\hat{\pi}_\beta) \geq \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\hat{\pi}_1}}\left[\bar{Q}^{\hat{\pi}_\beta}(s, \hat{\pi}_1(s)) - \bar{Q}^{\hat{\pi}_\beta}(s, \hat{\pi}_\beta(s))\right]$$
$$-\frac{2}{1-\gamma}\mathbb{E}_{s\sim d^{\hat{\pi}_1}}\mathbb{E}_{a\sim\hat{\pi}_1(\cdot|s)}\left[\frac{C_{\gamma,\delta}}{\sqrt{\mathcal{D}(s,a)}} + C_{\text{CFPI}}(s,a)\right] := \zeta.$$

For multi-step T iterative update, we similarly have with probability $1-\delta$,

$$J(\hat{\pi}_T) - J(\hat{\pi}_\beta) = \sum_{t=1}^{T} J(\hat{\pi}_t) - J(\hat{\pi}_{t-1}) \geq \sum_{t=1}^{T} \zeta^{(t)},$$

where $\mathcal{D}(s,a)$ denotes number of samples at s, a, $C_{\gamma,\delta}$ denotes the learning coefficient of SARSA and $C_{\text{CFPI}}(s,a)$ denotes the first-order approximation error from (3)*. We defer detailed derivation and the expression of $C_{\gamma,\delta}, \zeta^{(t)}$ and $C_{\text{CFPI}}(s,a)$ in Appendix A.4. When $a = a_\beta$, $C_{\text{CFPI}}(s,a) = 0$.*

By Theorem 3.1, $\hat{\pi}_1$ is a ζ-safe improved policy. The ζ safeness consists of two parts: C_{CFPI} is caused by the first-order approximation, and the $C_{\gamma,\delta}/\sqrt{\mathcal{D}(s,a)}$ term is incurred by the SARSA update. Similarly, $\hat{\pi}_T$ is a $\sum_{t=1}^{T}\zeta^{(t)}$-safe improved policy.

4 Related Work

Our methods belong and are motivated by the successful BCPO paradigm, which imposes constraints as in (2) to prevent from selecting OOD actions. Algorithms from this paradigm may apply different divergence functions, e.g., KL-divergence (Wu et al., 2019; Jaques et al., 2019), MMD (Kumar et al., 2019) or the MSE (Fujimoto & Gu, 2021). All these methods perform policy improvement via SGD. Instead, we perform CFPI by solving a linear approximation of (2). Another line of research enforces the behavior constraint via parameterization. BCQ (Fujimoto et al., 2019) learns a generative model as the behavior policy and a Q function to select the action from a set of perturbed behavior actions. Ghasemipour et al. (2021) further show that the perturbation model can be discarded.

The design of our CFPI operators is inspired by the SOTA online RL algorithm OAC (Ciosek et al., 2019). It treats the evaluation policy as the baseline π_b and obtains an optimistic exploration policy by solving a similar optimization problem as (7). We extend the result to accommodate a multi-modal π_b and overcome the optimization difficulties by leveraging Lemma 3.1. In Appendix H, we further draw connections with prior works that leveraged the Taylor expansion approach to RL.

Recently, one-step (Kostrikov et al., 2021; Brandfonbrener et al., 2021) algorithms have achieved great success. Instead of iteratively performing policy improvement and evaluation, these methods

[1]Note here we assume the discreteness only for the purpose of analysis. For the more general cases, please refer to Appendix A.4. In Theorem 3.1, $\mu_h^\pi(s,a|s_0,a_0) := P^\pi(s_h = s, a_h = a, |s_0 = s, a_0 = a)$.

Table 1: Comparison between our one-step policy and SOTA methods on the Gym-MuJoCo domain of D4RL. Our method uses the same τ for all datasets except Hopper-M-E (detailed in Appendix F.1). We report the mean and standard deviation of our method's performance across 10 seeds. Each seed contains an individual training process and evaluates the policy for 100 episodes. We use Cheetah for HalfCheetah, M for Medium, E for Expert, and R for Replay. We bold the best results for each task.

Dataset	SG-BC	MG-BC	DT	TT	OnestepRL	TD3+BC	CQL	IQL	Our $\mathcal{I}_{\text{MG}}(\hat{\pi}_\beta, \hat{Q}_0)$
Cheetah-M-v2	40.6	40.6	42.6	46.9	**55.6**	48.3	44.0	47.4	52.1 ± 0.3
Hopper-M-v2	53.7	53.9	67.6	61.1	83.3	59.3	58.5	66.2	$\mathbf{86.8 \pm 4.0}$
Walker2d-M-v2	71.9	70.0	74.0	79.8	85.6	83.7	72.5	78.3	$\mathbf{88.3 \pm 1.6}$
Cheetah-M-R-v2	34.9	33.0	36.6	41.9	42.4	44.6	**45.5**	44.2	44.5 ± 0.4
Hopper-M-R-v2	12.4	21.2	82.7	91.5	71.0	60.9	**95.0**	94.7	93.6 ± 7.9
Walker2d-M-R-v2	22.9	22.8	66.6	**82.6**	71.6	81.8	77.2	73.8	78.2 ± 5.6
Cheetah-M-E-v2	46.6	51.7	86.8	95.0	93.5	90.7	91.6	86.7	$\mathbf{97.3 \pm 1.8}$
Hopper-M-E-v2	53.9	69.2	**107.6**	101.9	102.1	98.0	105.4	91.5	104.2 ± 5.1
Walker2d-M-E-v2	92.3	93.2	108.1	110.0	**110.9**	110.1	108.8	109.6	$\mathbf{111.9 \pm 0.3}$
Total	429.1	455.6	672.6	710.1	716.0	677.4	698.5	692.4	$\mathbf{757.0 \pm 27.0}$

Table 2: Comparison between our Iterative $\mathcal{I}_{\text{MG}}$ and SOTA methods on the AntMaze domain. We report the mean and standard deviation across 5 seeds for our method with each seed evaluating for 100 episodes. The performance for all baselines is directly reported from the IQL paper. Our Iterative $\mathcal{I}_{\text{MG}}$ outperforms all baselines on 5 out of 6 tasks and obtains the best overall performance.

Dataset	BC	DT	Onestep RL	TD3+BC	CQL	IQL	Iterative $\mathcal{I}_{\text{MG}}$
antmaze-u-v0	54.6	59.2	64.3	78.6	74.0	87.5	$\mathbf{90.2 \pm 3.9}$
antmaze-u-d-v0	45.6	49.3	60.7	71.4	**84.0**	62.2	58.6 ± 15.2
antmaze-m-p-v0	0.0	0.0	0.3	10.6	61.2	71.2	$\mathbf{75.2 \pm 6.9}$
antmaze-m-d-v0	0.0	0.7	0.0	3.0	53.7	70.0	$\mathbf{72.2 \pm 7.3}$
antmaze-l-p-v0	0.0	0.0	0.0	0.2	15.8	39.6	$\mathbf{51.4 \pm 7.7}$
antmaze-l-d-v0	0.0	1.0	0.0	0.0	14.9	47.5	$\mathbf{52.4 \pm 10.9}$
Total	100.2	112.2	125.3	163.8	303.6	378.0	$\mathbf{400.0 \pm 52.0}$

only learn a Q function via SARSA without bootstrapping from OOD action value. These methods further apply an policy improvement operator (Wu et al., 2019; Peng et al., 2019) to extract a policy. We also instantiate a one-step algorithm with our CFPI operator and evaluate on standard benchmarks.

5 Experiments

Our experiments aim to demonstrate the effectiveness of our CFPI operators. Firstly, on the standard offline RL benchmark D4RL (Fu et al., 2020), we show that instantiating offline RL algorithms with our CFPI operators in both one-step and iterative manners outperforms SOTA methods (Sec. 5.1). Secondly, we show that our CFPI operator can improve a policy learned by other algorithms (Sec. 5.2). Ablation studies in Sec. 5.3 further shows our superiority over the other policy improvement operators and demonstrate the benefit of modeling the behavior policy as a Gaussian Mixture.

5.1 Comparison with SOTA offline RL algorithms

We instantiate a one-step offline RL algorithm from Algorithm 1 with our policy improvement operator $\mathcal{I}_{\text{MG}}$. We learned a Gaussian Mixture baseline policy $\hat{\pi}_\beta$ via behavior cloning. We employed the IQN (Dabney et al., 2018a) architecture to model the Q value network for its better generalizability, as we need to estimate out-of-buffer $Q(s, a)$ during policy deployment. We trained the $\hat{Q}_0$ with SARSA algorithm (Sutton & Barto, 2018; Parisotto et al., 2015). Appendix F.1 includes detailed training procedures of $\hat{\pi}_\beta$ and $\hat{Q}_0$ with full HP settings. We obtain our one-step policy as $\mathcal{I}_{\text{MG}}(\hat{\pi}_\beta, \hat{Q}_0; \tau)$.

We evaluate the effectiveness of our one-step algorithm on the D4RL benchmark focusing on the Gym-MuJoCo domain, which contains locomotion tasks with dense rewards. Table 1 compares our one-step algorithm with SOTA methods, including the other one-step actor-critic methods IQL (Kostrikov et al., 2021), OneStepRL (Brandfonbrener et al., 2021), BCPO method TD3+BC (Fujimoto & Gu, 2021), conservative method CQL (Kumar et al., 2020b), and trajectory optimization methods DT (Chen et al., 2021), TT (Janner et al., 2021). We also include the performance of two behavior policies

SG-BC and MG-BC modeled with Single Gaussian and Gaussian Mixture, respectively. We directly report results for IQL, BCQ, TD3+BC, CQL, and DT from the IQL paper, and TT's result from its own paper. Note that OneStepRL instantiates three different algorithms. We only report its (Rev. KL Reg) result because this algorithm follows the BCPO paradigm and achieves the best overall performance. We highlight that OnesteRL reports the results by tuning the HP for each dataset.

Results in Table 1 demonstrate that our one-step algorithm outperforms the other algorithms by a significant margin without training a policy to maximize its Q-value through SGD. We note that we use the same τ for all datasets except Hopper-M-E. In Sec. 5.3, we will perform ablation studies and provide a fair comparison between our CFPI operators and the other policy improvement operators.

We further instantiate an iterative algorithm with $\mathcal{I}_{\text{MG}}$ and evaluate its effectiveness on the challenging AntMaze domain of D4RL. The 6 tasks from AntMaze are more challenging due to their sparse-reward nature and lack of optimal trajectories in the static datasets. Table 2 compares our Iterative $\mathcal{I}_{\text{MG}}$ with SOTA algorithms on the AntMaze domain. Our method uses the same set of HP for all 6 tasks, outperforming all baselines on 5 out of 6 tasks and obtaining the best overall performance. Appendix C.1 presents additional details with pseudo-codes and training curves.

5.2 Improvement over a learned policy

In this section, we show that our CFPI operator $\mathcal{I}_{\text{SG}}$ can further improve the performance of a Single Gaussian policy π_{IQL} learned by IQL (Kostrikov et al., 2021) on the AntMaze domain. We first obtain the IQL policy π_{IQL} and Q_{IQL} by training for 1M gradient steps using the PyTorch Implementation from RLkit (Berkeley). We emphasize that we follow the authors' exact training and evaluation protocols and include all training curves in Appendix G.6. Interestingly, while the running average of the evaluation results during the course of training matches the reported results in the IQL paper, Table 3 shows that the evaluation of the final 1M-step policy π_{IQL} does not match the reported performance on all 6 tasks, **echoing the training instability we are trying to resolve with our CFPI operators**. This demonstrates how drastically performance can fluctuate across just dozens of epochs. Thanks to the tractability of $\mathcal{I}_{\text{SG}}$, we directly obtain an improved policy $\mathcal{I}_{\text{SG}}(\pi_{\text{IQL}}, Q_{\text{IQL}}; \tau)$ that achieves better overall performance than both π_{IQL} (train) and (1M), as shown in Table 3. We tune the HP τ using a small set of seeds for each task following the practice of (Brandfonbrener et al., 2021; Fu et al., 2020) and include more details in Appendix F.2 and G.6.

Table 3: Our $\mathcal{I}_{\text{SG}}(\pi_{\text{IQL}}, Q_{\text{IQL}})$ improves over the policy π_{IQL} learned by IQL on AntMaze. We report the mean and standard deviation 10 seeds. Each seed evaluates for 100 episodes.

Dataset	π_{IQL} (train)	π_{IQL} (1M)	$\mathcal{I}_{\text{SG}}(\pi_{\text{IQL}}, Q_{\text{IQL}})$
antmaze-u-v0	$\mathbf{87.4 \pm 3.2}$	83.6 ± 3.2	85.1 ± 5.3
antmaze-u-d-v0	$\mathbf{59.0 \pm 5.7}$	55.8 ± 7.9	55.0 ± 9.1
antmaze-m-p-v0	71.1 ± 5.43	64.2 ± 13.2	$\mathbf{75.5 \pm 6.1}$
antmaze-m-d-v0	70.0 ± 6.16	66.8 ± 9.4	$\mathbf{79.9 \pm 3.8}$
antmaze-l-p-v0	34.4 ± 6.04	35.6 ± 7.0	$\mathbf{37.7 \pm 7.7}$
antmaze-l-d-v0	39.8 ± 9.09	38.8 ± 7.1	$\mathbf{40.1 \pm 5.6}$
Total	361.7 ± 35.6	344.7 ± 47.8	$\mathbf{373.3 \pm 37.5}$

5.3 Ablation studies

We first provide a fair comparison with the other policy improvement operators, demonstrating the effectiveness of solving the approximated BCPO (4) and modeling the behavior policy as a Gaussian Mixture. Additionally, we examine the sensitivity on τ, ablate the number of Gaussian components, and discuss the limitation by ablating the Q network in Appendix G.2, G.3, G.4, respectively.

Effectiveness of our CFPI operators. In Table 4, we compare our CFPI operators with two policy improvement operators, namely, Easy BCQ (EBCQ) and Rev. KL Reg from OneStepRL (Brandfonbrener et al., 2021). EBCQ doe not require training either, returning a policy by selecting an action that maximizes a learned $\hat{Q}$ from N_{bcq} actions randomly sampled from the behavior policy $\hat{\pi}_\beta$. Rev. KL Reg sets $D(\cdot, \cdot)$ in (2) as the reverse KL divergence and solves the problem via SGD, with α controlling the regularization strength. We omit the comparison with the other learning-based operator Exp. Weight, as Rev. KL Reg achieves the best overall performance in OneStepRL.

For all methods, we present results with $\hat{\pi}_\beta$ modeled by Single Gaussian (SG-) and Gaussian Mixture (MG-). To ensure a fair comparison, we employ the same $\hat{Q}_0$ and $\hat{\pi}_\beta$ modeled and learned in the same way as in Sec. 5.1 for all methods. Moreover, we tune N_{bcq} for EBCQ, α for Rev. KL Reg, and

Table 4: Ablation studies of our Method on the Gym-MuJoCo domain. Again we report the mean and std of 10 seeds, each seed evaluates for 100 episodes.

Dataset	SG-EBCQ	MG-EBCQ	SG-Rev. KL Reg	MG-Rev. KL Reg	$\mathcal{I}_{\text{SG}}$	$\mathcal{I}_{\text{MG}}$
Cheetah-M-v2	$\mathbf{53.3 \pm 0.2}$	51.5 ± 0.2	47.1 ± 0.2	47.0 ± 0.2	51.1 ± 0.1	52.1 ± 0.3
Hopper-M-v2	$\mathbf{86.8 \pm 5.2}$	82.5 ± 1.9	70.3 ± 7.0	76.3 ± 6.9	75.6 ± 3.7	$\mathbf{86.8 \pm 4.0}$
Walker2d-M-v2	85.2 ± 5.1	85.2 ± 2.1	82.4 ± 1.0	82.8 ± 1.8	88.1 ± 1.1	$\mathbf{88.3 \pm 1.6}$
Cheetah-M-R-v2	43.5 ± 0.6	43.0 ± 0.3	44.3 ± 0.4	44.4 ± 0.5	42.8 ± 0.4	$\mathbf{44.5 \pm 0.4}$
Hopper-M-R-v2	88.5 ± 12.2	83.6 ± 10.3	$\mathbf{99.7 \pm 1.0}$	99.4 ± 2.1	87.7 ± 8.7	93.6 ± 7.9
Walker2d-M-R-v2	75.4 ± 4.6	73.1 ± 5.2	63.6 ± 28.5	69.7 ± 30.9	71.3 ± 4.4	$\mathbf{78.2 \pm 5.6}$
Cheetah-M-E-v2	81.8 ± 5.4	84.5 ± 4.6	78.9 ± 9.8	65.0 ± 10.1	91.1 ± 3.1	$\mathbf{97.3 \pm 1.8}$
Hopper-M-E-v2	40.0 ± 5.8	56.1 ± 6.2	76.6 ± 18.3	$\mathbf{79.4 \pm 32.6}$	70.3 ± 8.9	73.0 ± 10.5
Walker2d-M-E-v2	111.1 ± 1.8	111.1 ± 1.0	106.7 ± 4.1	107.1 ± 4.0	111.1 ± 1.1	$\mathbf{111.9 \pm 0.3}$
Total	665.5 ± 41.0	670.6 ± 31.9	669.7 ± 70.3	671.2 ± 89.1	688.9 ± 31.6	$\mathbf{725.8 \pm 32.4}$

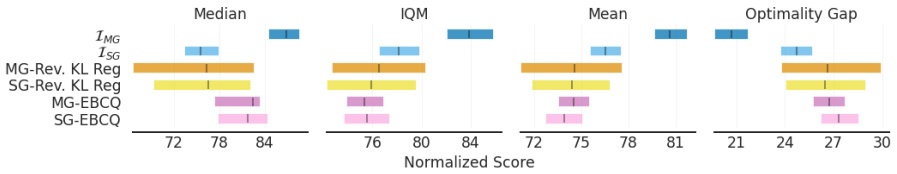


Figure 2: Aggregate metrics (Agarwal et al., 2021) with 95% CIs based on results reported in Table 4. The CIs are estimated using the percentile bootstrap with stratified sampling. Higher median, IQM, and mean scores, and lower Optimality Gap correspond to better performance. Our $\mathcal{I}_{\text{MG}}$ outperforms baselines by a significant margin based on all four metrics. Appendix E includes additional details.

τ for our methods. Each method uses the same set of HP for all datasets. As a result, the Hopper-M-E performance of $\mathcal{I}_{\text{MG}}$ reported in Table 4 is different from Table 1. Appendix F.1 includes further details on the HP tuning and corresponding experiment results in Table 9, 10, 11 and 12.

As is shown in Table 4 and Fig. 2, our $\mathcal{I}_{\text{MG}}$ clearly outperforms all baselines by a significant margin. The learning-based method Rev. KL Reg exhibits a substantial amount of variance, again echoing the training instability we are trying to resolve. Moreover, our CFPI operators outperform their EBCQ counterparts, demonstrating the effectiveness of solving the approximated BCPO.

Effectiveness of Gaussian Mixture. As the three M-E datasets are collected by an expert and medium policy, we should recover an expert performance as long as we can 1) capture the two modes of the action distribution 2) and always select action from the expert mode. In other words, we can leverage the $\hat{Q}_0$ learned by SARSA to select actions from the mean of each Gaussian component, resulting in a mode selection algorithm (MG-MS) that selects its action by

$$\mu_{\text{mode}} = \arg\max_{\hat{\mu}_i} \quad \hat{Q}_0(s, \hat{\mu}_i), \quad \text{s.t.} \quad \{\hat{\mu}_i | \hat{\lambda}_i > \xi\}, \quad \text{where} \quad \sum\nolimits_{i=1:N} \hat{\lambda}_i \mathcal{N}(\hat{\mu}_i, \hat{\Sigma}_i) = \hat{\pi}_\beta, \tag{16}$$

ξ is set to filter out trivial components. Our MG-MS achieves an expert performance on Hopper-M-E (104.2 ± 5.1) and Walker2d-M-E (104.1 ± 6.7), and matches SOTA algorithms in Cheetah-M-E (91.3 ± 2.1). Appendix G.1 includes the full results of MG-MS on the Gym MuJoCo domain.

6 Conclusion and Limitations

Motivated by the behavior constraint in the BCPO paradigm, we propose CFPI operators that perform policy improvement by solving an approximated BCPO in closed form. As practical datasets are usually generated by heterogeneous policies, we use the Gaussian Mixture to model the data-generating policies and overcome extra optimization difficulties by leveraging the LogSumExp's LB and Jensen's Inequality. We instantiate a one-step offline RL algorithm with our CFPI operator and show that it can outperform SOTA algorithms on the Gym-MuJoCo domain of the D4RL benchmark.

Our CFPI operators avoid the training instability incurred by policy improvement through SGD. However, our method still requires learning a good Q function. Specifically, our operators rely on the gradient information provided by the Q, and its accuracy largely impacts the effectiveness of our policy improvement. Therefore, one promising future direction for this work is to investigate ways to robustify the policy improvement given a noisy Q.

REPRODUCIBILITY STATEMENT

We include our codes in the supplementary material.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhihong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. *arXiv preprint arXiv:2202.11566*, 2022.

RAIL Berkeley. Rlkit: Reinforcement learning framework and algorithms implemented in pytorch. URL `https://github.com/rail-berkeley/rlkit`.

Christopher M Bishop and Markus Svensén. Bayesian hierarchical mixtures of experts. *arXiv preprint arXiv:1212.2447*, 2012.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

David Brandfonbrener, William F Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *arXiv preprint arXiv:2106.08909*, 2021.

Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, et al. A framework for data-driven robotics. *arXiv preprint arXiv:1909.12200*, 2019.

James J Callahan. *Advanced calculus: a geometric view*, volume 1. Springer, 2010.

Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International Conference on Machine Learning*, pp. 1042–1051. PMLR, 2019.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34, 2021.

Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor-critic, 2019.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pp. 1096–1105. PMLR, 2018a.

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018b.

Yaqi Duan, Zeyu Jia, and Mengdi Wang. Minimax-optimal off-policy evaluation with linear function approximation. In *International Conference on Machine Learning*, pp. 2701–2709. PMLR, 2020.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

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 and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *CoRR*, abs/1802.09477, 2018. URL http://arxiv.org/abs/1802.09477.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062, 2019.

Scott Fujimoto, David Meger, Doina Precup, Ofir Nachum, and Shixiang Shane Gu. Why should i trust you, bellman? the bellman error is a poor replacement for value error. *arXiv preprint arXiv:2201.12417*, 2022.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, pp. 3682–3691. PMLR, 2021.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018. URL http://arxiv.org/abs/1801.01290.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2021.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Chi Jin, Yuchen Zhang, Sivaraman Balakrishnan, Martin J Wainwright, and Michael I Jordan. Local maxima in the likelihood of gaussian mixture models: Structural results and algorithmic consequences. *Advances in neural information processing systems*, 29, 2016.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pp. 5084–5096. PMLR, 2021.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

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.

Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. *Advances in Neural Information Processing Systems*, 33: 18560–18572, 2020a.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020b.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pp. 45–73. Springer, 2012.

Hoang Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pp. 3703–3712. PMLR, 2019.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *ISAIM*, 4(5):9, 2006.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Xiaoteng Ma, Li Xia, Zhengyuan Zhou, Jun Yang, and Qianchuan Zhao. Dsac: Distributional soft actor critic for risk-sensitive reinforcement learning. *arXiv preprint arXiv:2004.14547*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

Emilio Parisotto, Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *CoRR*, abs/1511.06342, 2015.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Theodore Perkins and Doina Precup. A convergent form of approximate policy iteration. *Advances in neural information processing systems*, 15, 2002.

Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL http://arxiv.org/abs/1502.05477.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. PMLR, 2014.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Yunhao Tang, Michal Valko, and Rémi Munos. Taylor expansion policy optimization. In *International Conference on Machine Learning*, pp. 9397–9406. PMLR, 2020.

Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Lei Xu, Michael Jordan, and Geoffrey E Hinton. An alternative model for mixtures of experts. *Advances in neural information processing systems*, 7, 1994.

Ming Yin and Yu-Xiang Wang. Asymptotically efficient off-policy evaluation for tabular reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3948–3958. PMLR, 2020.

Ming Yin, Yaqi Duan, Mengdi Wang, and Yu-Xiang Wang. Near-optimal offline reinforcement learning with linear representation: Leveraging variance information with pessimism. *International Conference on Learning Representations*, 2022.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

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, 2021.

Ruiqi Zhang, Xuezhou Zhang, Chengzhuo Ni, and Mengdi Wang. Off-policy fitted q-evaluation with differentiable function approximators: Z-estimation and inference theory. *International Conference on Machine Learning*, 2022.

Shaofeng Zou, Tengyu Xu, and Yingbin Liang. Finite-sample analysis for sarsa with linear function approximation. *Advances in neural information processing systems*, 32, 2019.

Appendix

OUTLINE OF THE APPENDIX

In this Appendix, we organize the content in the following ways:

- Appendix A presents the missing proofs for Proposition 3.1, 3.2, 3.3 and Theorem 3.1 in the main paper.
- Appendix B justify one HP setting for Equation 15.
- Appendix C discusses how to instantiate multi-step and iterative algorithms from our algorithm template Algorithm 1.
- Appendix D provides the derivation of a new CFPI operator that can work with both deterministic and VAE policy.
- Appendix E conducted a reliable evaluation to demonstrate the statistical significance of our methods and address statistical uncertainty.
- Appendix F gives experiment details, HP settings and corresponding experiment results.
- Appendix G provides additional ablation studies and experiment results.
- Appendix H includes additional related work and discusses the relationship between our method and prior literature that leverage the Taylor expansion approach.

Our experiments are conducted on various types of 8GPUs machines. Different machines may have different GPU types, such as NVIDIA GA100 and TU102. Training a behavior policy for 500K gradient steps takes around 40 minutes, while training a Q network for 500K gradient steps takes around 50 minutes.

A PROOFS AND THEORETICAL RESULTS

A.1 PROOF OF PROPOSITION 3.1

Proposition 3.1. *The optimization problem* (5) *has a closed-form solution that is given by*

$$\mu_{sg}(\tau) = \mu_\beta + \frac{\sqrt{2\log\tau}\,\Sigma_\beta\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}}{\sqrt{\left[\nabla_a Q(s,a)\right]^T_{a=\mu_\beta}\Sigma_\beta\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}}}, \quad \textit{where} \;\; \delta = \frac{1}{2}\log\det(2\pi\Sigma_\beta) + \log\tau \tag{17}$$

Proof. The optimization problem (5) can be converted into the QCLP

$$\max_{\mu} \quad \mu^T\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}, \quad \text{s.t.} \quad \frac{1}{2}(\mu-\mu_\beta)^T\Sigma_\beta^{-1}(\mu-\mu_\beta) \le \delta - \frac{1}{2}\log\det(2\pi\Sigma_\beta) \tag{18}$$

Following a similar procedure as is in OAC (Ciosek et al., 2019), we first derive the Lagrangian below:

$$L = \mu^T\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta} - \eta\left(\frac{1}{2}(\mu-\mu_\beta)^T\Sigma_\beta^{-1}(\mu-\mu_\beta) - \delta + \frac{1}{2}\log\det(2\pi\Sigma_\beta)\right) \tag{19}$$

Taking the derivatives w.r.t μ, we get

$$\nabla_\mu L = \left[\nabla_a Q(s,a)\right]_{a=\mu_\beta} - \eta\Sigma_\beta^{-1}(\mu-\mu_\beta) \tag{20}$$

By setting $\nabla_\mu L = 0$, we get

$$\mu = \mu_\beta + \frac{1}{\eta}\Sigma_\beta\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta} \tag{21}$$

To satisfy the the KKT conditions (Boyd et al., 2004), we have $\eta > 0$ and

$$(\mu-\mu_\beta)^T\Sigma_\beta^{-1}(\mu-\mu_\beta) = 2\delta - \log\det(2\pi\Sigma_\beta) \tag{22}$$

Finally with (21) and (22), we get

$$\eta = \sqrt{\frac{\left[\nabla_a Q(s,a)\right]^T_{a=\mu_\beta}\Sigma_\beta\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}}{2\delta - \log\det(2\pi\Sigma_\beta)}} \tag{23}$$

By setting $\delta = \frac{1}{2}\log\det(2\pi\Sigma_\beta) + \log\tau$ and plugging (23) to (21), we obtain the final solution as

$$\mu_{\text{sg}}(\tau) = \mu_\beta + \frac{\sqrt{2\log\tau}\,\Sigma_\beta\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}}{\sqrt{\left[\nabla_a Q(s,a)\right]^T_{a=\mu_\beta}\Sigma_\beta\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}}}, \tag{24}$$

which completes the proof. □

A.2 PROOF OF PROPOSITION 3.2

Proposition 3.2. *By applying the first inequality of Lemma 3.1 to the constraint of* (10)*, we can derive an optimization problem that lower bounds* (10)

$$\begin{aligned} &\max_{\mu} \quad \mu^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta} \\ &\textit{s.t.} \quad \max_i \left\{ -\frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1}(\mu-\mu_i) - \frac{1}{2}\log\det(2\pi\Sigma_i) + \log\lambda_i \right\} \geq -\delta, \end{aligned} \tag{25}$$

and the closed-form solution to (11) *is given by*

$$\begin{aligned} &\mu_{lse}(\tau) = \arg\max_{\bar{\mu}_i(\delta)} \quad \bar{\mu}_i^T \left[\nabla_a Q(s,a)\right]_{a=\mu_i}, \quad \textit{s.t.} \quad \delta = \frac{1}{2}\min_i \left\{\log\lambda_i \det(2\pi\Sigma_i)\right\} + \log\tau \\ &\textit{where} \quad \bar{\mu}_i(\delta) = \mu_i + \sqrt{\frac{2(\delta+\log\lambda_i) - \log\det(2\pi\Sigma_i)}{\left[\nabla_a Q(s,a)\right]_{a=\mu_i}^T \Sigma_i \left[\nabla_a Q(s,a)\right]_{a=\mu_i}}} \Sigma_i \left[\nabla_a Q(s,a)\right]_{a=\mu_i} \end{aligned} \tag{26}$$

Proof. Recall that the Gaussian Mixture behavior policy is constructed by

$$\pi_\beta = \sum_{i=1}^{N} \lambda_i \mathcal{N}(\mu_i, \Sigma_i), \tag{27}$$

We first divide the optimization problem (25) into N sub-problems, with each sub-problem i given by

$$\begin{aligned} &\max_{\mu} \quad \mu^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta} \\ &\text{s.t.} \quad -\frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1}(\mu-\mu_i) - \frac{1}{2}\log\det(2\pi\Sigma_i) + \log\lambda_i \geq -\delta, \end{aligned} \tag{28}$$

which is equivalent to solving problem (5) for each Gaussian component with an additional constant term $\log\lambda_i$, and thus has a *unique* closed-form solution.

Define the maximizer for each sub-problem i as $\bar{\mu}_i(\delta)$, though $\bar{\mu}_i(\delta)$ does not always exist. Whenever $-\frac{1}{2}\log\det(2\pi\Sigma_i) + \log\lambda_i < -\delta$, there will be no μ satisfying the constraint as $\frac{1}{2}(\mu-\mu_i)^T\Sigma_i^{-1}(\mu-\mu_i)$ is always greater than 0. We thus set $\bar{\mu}_i(\delta)$ to be *None* in this case. Next, we will show that there does not exist any $\breve{\mu} \notin \{\bar{\mu}_i(\delta) | i = 1 \dots N\}$, s.t., $\breve{\mu}$ is the maximizer of (25). We can show this by contradiction. Suppose there exists a $\breve{\mu} \notin \{\bar{\mu}_i(\delta) | i = 1 \dots N\}$ maximizing (25), there exists at least one $j \in \{1, \dots, N\}$ s.t.

$$-\frac{1}{2}(\breve{\mu}-\mu_j)^T \Sigma_j^{-1}(\breve{\mu}-\mu_j) - \frac{1}{2}\log\det(2\pi\Sigma_j) + \log\lambda_j \geq -\delta. \tag{29}$$

Since $\breve{\mu}$ is the maximizer of (25), it should also be maximizer of the sub-problem j. However, the maximizer for sub-problem j is given by $\bar{\mu}_j(\delta) \neq \breve{\mu}$, contradicting with the fact that $\breve{\mu}$ is the maximizer of the sub-problem j. Therefore, the optimal solution to (25) has to be given by

$$\arg\max_{\bar{\mu}_i} \quad \bar{\mu}_i^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta} \quad \text{where} \quad \bar{\mu}_i \in \{\bar{\mu}_i(\delta) | i = 1 \dots N\} \tag{30}$$

To solve each sub-problem i, it is natural to set $a_\beta = \mu_i$, which reformulate the sub-problem i as below

$$\begin{aligned} &\max_{\mu} \quad \mu^T \left[\nabla_a Q(s,a)\right]_{a=\mu_i} \\ &\text{s.t.} \quad \frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1}(\mu-\mu_i) \leq \delta - \frac{1}{2}\log\det(2\pi\Sigma_i) + \log\lambda_i, \end{aligned} \tag{31}$$

Note that problem (31) is also a QCLP similar to the problem (5). Therefore, we can derive its solution by following similar procedures as in Appendix A.1, resulting in

$$\bar{\mu}_i(\delta) = \mu_i + \sqrt{\frac{2(\delta+\log\lambda_i) - \log\det(2\pi\Sigma_i)}{\left[\nabla_a Q(s,a)\right]_{a=\mu_i}^T \Sigma_i \left[\nabla_a Q(s,a)\right]_{a=\mu_i}}} \Sigma_i \left[\nabla_a Q(s,a)\right]_{a=\mu_i}. \tag{32}$$

We complete the proof by further setting $\delta = \frac{1}{2}\min_i \left\{\log\lambda_i \det(2\pi\Sigma_i)\right\} + \log\tau$. □

A.3 PROOF OF PROPOSITION 3.3

Proposition 3.3. *By applying the second inequality of Lemma 3.1 to the constraint of* (10)*, we can derive an optimization problem that lower bounds* (10)

$$\begin{aligned} \max_{\mu} \quad & \mu^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta} \\ \textit{s.t.} \quad & \sum_{i=1}^{N} \lambda_i \left(-\frac{1}{2}\log\det(2\pi\Sigma_i) - \frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1}(\mu-\mu_i) \right) \geq -\delta \end{aligned} \tag{33}$$

and the closed-form solution to (13) *is given by*

$$\begin{aligned} &\mu_{jensen}(\tau) = \bar{\mu} + \sqrt{\frac{2\log\tau - \sum_{i=1}^{N}\lambda_i \mu_i^T \Sigma_i^{-1}\mu_i + \bar{\mu}^T \overline{\Sigma}^{-1}\bar{\mu}}{\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}^T \overline{\Sigma} \left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}}}\ \overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}, \\ &\textit{where } \overline{\Sigma} = \left(\sum_{i=1}^{N}\lambda_i \Sigma_i^{-1}\right)^{-1}, \quad \bar{\mu} = \overline{\Sigma}\left(\sum_{i=1}^{N}\lambda_i \Sigma_i^{-1}\mu_i\right), \quad \delta = \log\tau + \frac{1}{2}\sum_{i=1}^{N}\lambda_i \log\det(2\pi\Sigma_i) \end{aligned} \tag{34}$$

Proof. Note that problem (33) is also a QCLP. Before deciding the value of a_β, we first derive its Lagrangian with a general a_β below

$$L = \mu^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta} - \eta\left(\sum_{i=1}^{N}\lambda_i\left(\frac{1}{2}\log\det(2\pi\Sigma_i) + \frac{1}{2}(\mu-\mu_i)^T\Sigma_i^{-1}(\mu-\mu_i)\right) - \delta\right) \tag{35}$$

Taking the derivatives w.r.t μ, we get

$$\nabla_\mu L = \left[\nabla_a Q(s,a)\right]_{a=a_\beta} - \eta\left(\sum_{i=1}^{N}\lambda_i\left(\Sigma_i^{-1}(\mu-\mu_i)\right)\right) \tag{36}$$

By setting $\nabla_\mu L = 0$, we get

$$\begin{aligned} \mu &= \left(\sum_{i=1}^{N}\lambda_i\Sigma_i^{-1}\right)^{-1}\left(\sum_{i=1}^{N}\lambda_i\Sigma_i^{-1}\mu_i\right) + \frac{1}{\eta}\left(\sum_{i=1}^{N}\lambda_i\Sigma_i^{-1}\right)\left[\nabla_a Q(s,a)\right]_{a=a_\beta} \\ &= \bar{\mu} + \frac{1}{\eta}\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=a_\beta}, \\ \text{where} \quad \overline{\Sigma} &= \left(\sum_{i=1}^{N}\lambda_i\Sigma_i^{-1}\right)^{-1}, \quad \bar{\mu} = \overline{\Sigma}\left(\sum_{i=1}^{N}\lambda_i\Sigma_i^{-1}\mu_i\right), \end{aligned} \tag{37}$$

Equation 37 shows that the final solution to the problem (33) will be a shift from the pseudo-mean $\bar{\mu}$. Therefore, setting $a_\beta = \bar{\mu}$ becomes a natural choice.

Furthermore, by satisfying the KKT conditions, we have $\eta > 0$ and

$$\sum_{i=1}^{N}\lambda_i\left(\mu-\mu_i\right)^T\Sigma_i^{-1}\left(\mu-\mu_i\right) = 2\delta - \sum_{i=1}^{N}\lambda_i\log\det(2\pi\Sigma_i) \tag{38}$$

Plugging (33) into (38) gives the equation below

$$\begin{aligned} &\sum_{i=1}^{N}\lambda_i\left(\bar{\mu} + \frac{1}{\eta}\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}} - \mu_i\right)^T \Sigma_i^{-1}\left(\bar{\mu} + \frac{1}{\eta}\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}} - \mu_i\right) \\ &\quad = 2\delta - \sum_{i=1}^{N}\lambda_i\log\det(2\pi\Sigma_i). \end{aligned} \tag{39}$$

The LHS of (39) can be reformulated as

$$
\begin{aligned}
& \sum_{i=1}^{N} \lambda_i \left(\bar{\mu} + \frac{1}{\eta}\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}} - \mu_i\right)^T \Sigma_i^{-1} \left(\bar{\mu} + \frac{1}{\eta}\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}} - \mu_i\right) \\
= \quad & \frac{1}{\eta^2}\sum_{i=1}^{N} \lambda_i \left(\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}\right)^T \Sigma_i^{-1} \left(\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}\right) \\
& + \frac{2}{\eta}\sum_{i=1}^{N} \lambda_i \left(\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}\right)^T \Sigma_i^{-1}\left(\bar{\mu} - \mu_i\right) \\
& + \sum_{i=1}^{N} \lambda_i \left(\bar{\mu} - \mu_i\right)^T \Sigma_i^{-1} \left(\bar{\mu} - \mu_i\right)
\end{aligned} . \tag{40}
$$

We note that the second line of (40)'s RHS can be reduced to

$$
\begin{aligned}
& \frac{2}{\eta}\sum_{i=1}^{N} \lambda_i \left(\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}\right)^T \Sigma_i^{-1}\left(\bar{\mu} - \mu_i\right) \\
= \quad & \frac{2}{\eta}\left(\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}\right)^T \left(\left(\sum_{i=1}^{N} \lambda_i \Sigma_i^{-1}\right)\bar{\mu} - \sum_{i=1}^{N} \lambda_i \Sigma_i^{-1}\mu_i\right) \\
= \quad & \frac{2}{\eta}\left(\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}\right)^T \left(\overline{\Sigma}^{-1}\bar{\mu} - \overline{\Sigma}^{-1}\left(\overline{\Sigma}\sum_{i=1}^{N} \lambda_i \Sigma_i^{-1}\mu_i\right)\right) \\
= \quad & \frac{2}{\eta}\left(\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}\right)^T \left(\overline{\Sigma}^{-1}\bar{\mu} - \overline{\Sigma}^{-1}\bar{\mu}\right) \\
= \quad & 0
\end{aligned} . \tag{41}
$$

Therefore, (40) can be further reformulated as

$$
\begin{aligned}
& \sum_{i=1}^{N} \lambda_i \left(\bar{\mu} + \frac{1}{\eta}\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}} - \mu_i\right)^T \Sigma_i^{-1} \left(\bar{\mu} + \frac{1}{\eta}\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}} - \mu_i\right) \\
= \quad & \frac{1}{\eta^2}\sum_{i=1}^{N} \lambda_i \left(\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}\right)^T \Sigma_i^{-1} \left(\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}\right) \\
& + \sum_{i=1}^{N} \lambda_i \left(\bar{\mu} - \mu_i\right)^T \Sigma_i^{-1} \left(\bar{\mu} - \mu_i\right) \\
= \quad & \frac{1}{\eta^2}\left(\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}\right)^T \left(\sum_{i=1}^{N} \lambda_i \Sigma_i^{-1}\right)\left(\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}\right) \\
& + \sum_{i=1}^{N} \lambda_i \left(\bar{\mu} - \mu_i\right)^T \Sigma_i^{-1} \left(\bar{\mu} - \mu_i\right) \\
= \quad & \frac{1}{\eta^2}\left(\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}\right)^T \overline{\Sigma}^{-1}\left(\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}\right) \\
& + \sum_{i=1}^{N} \lambda_i \left(\bar{\mu} - \mu_i\right)^T \Sigma_i^{-1} \left(\bar{\mu} - \mu_i\right) \\
= \quad & \frac{1}{\eta^2}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}^T \overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}} + \sum_{i=1}^{N} \lambda_i \left(\bar{\mu} - \mu_i\right)^T \Sigma_i^{-1} \left(\bar{\mu} - \mu_i\right)
\end{aligned} . \tag{42}
$$

To this point, (39) can be reformulated as

$$\begin{aligned} & \frac{1}{\eta^2}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}^T \overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}} + \sum_{i=1}^{N} \lambda_i \left(\bar{\mu}-\mu_i\right)^T \Sigma_i^{-1}\left(\bar{\mu}-\mu_i\right) \\ = \quad & 2\delta - \sum_{i=1}^{N} \lambda_i \log\det(2\pi\Sigma_i) \end{aligned} \tag{43}$$

We can thus express η as below

$$\eta = \sqrt{\frac{\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}^T \overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}}{2\delta - \sum_{i=1}^{N} \lambda_i \log\det(2\pi\Sigma_i) - \sum_{i=1}^{N} \lambda_i \left(\bar{\mu}-\mu_i\right)^T \Sigma_i^{-1}\left(\bar{\mu}-\mu_i\right)}} \tag{44}$$

By setting $\delta = \frac{1}{2}\sum_{i=1}^{N} \lambda_i \log\det(2\pi\Sigma_i) + \log\tau$, we have

$$\begin{aligned} \eta &= \sqrt{\frac{\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}^T \overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}}{2\log\tau - \sum_{i=1}^{N} \lambda_i \left(\bar{\mu}-\mu_i\right)^T \Sigma_i^{-1}\left(\bar{\mu}-\mu_i\right)}} \\ &= \sqrt{\frac{\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}^T \overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}}{2\log\tau - \sum_{i=1}^{N} \lambda_i \bar{\mu}^T\Sigma_i^{-1}\bar{\mu} + 2\bar{\mu}^T\sum_{i=1}^{N} \lambda_i \Sigma_i^{-1}\mu_i - \sum_{i=1}^{N} \lambda_i \mu_i^T\Sigma_i^{-1}\mu_i}} \\ &= \sqrt{\frac{\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}^T \overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}}{2\log\tau - \sum_{i=1}^{N} \bar{\mu}^T\overline{\Sigma}^{-1}\bar{\mu} + 2\bar{\mu}^T\overline{\Sigma}^{-1}\bar{\mu} - \sum_{i=1}^{N} \lambda_i \mu_i^T\Sigma_i^{-1}\mu_i}} \\ &= \sqrt{\frac{\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}^T \overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}}{2\log\tau + \bar{\mu}^T\overline{\Sigma}^{-1}\bar{\mu} - \sum_{i=1}^{N} \lambda_i \mu_i^T\Sigma_i^{-1}\mu_i}} \end{aligned}. \tag{45}$$

Finally, plugging (45) into (37), with $a_\beta = \bar{\mu}$, we have

$$\mu_{\text{jensen}}(\tau) = \bar{\mu} + \sqrt{\frac{2\log\tau - \sum_{i=1}^{N} \lambda_i \mu_i^T\Sigma_i^{-1}\mu_i + \bar{\mu}^T\overline{\Sigma}^{-1}\bar{\mu}}{\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}^T \overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}}}\ \overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}, \tag{46}$$

which completes the proof. □

A.4 PROOF OF THEOREM 3.1

In this section we prove the *safe policy improvement* presented in Section 3.3. Algorithm 1 follows the *approximate policy iteration* (API) (Perkins & Precup, 2002) by iterating over the policy evaluation ($\mathcal{E}$ step, Line 4) and policy improvement ($\mathcal{I}$ step, Line 5). Therefore, to verify $\mathcal{E}$ provides the improvement, we need to first show policy evaluation $\hat{Q}_t$ is accurate. In particular, we focus on the SARSA updates (Line 2), which is a form of on-policy Fitted Q-Iteration (Sutton & Barto, 2018). Fortunately, it is known that FQI is statistically efficient (*e.g.* (Chen & Jiang, 2019)) under the mild condition for the function approximation class. Its linear counterpart, least-square value iteration, is also shown to be efficient for offline reinforcement learning (Jin et al., 2021; Yin et al., 2022). Recently, (Zou et al., 2019) shows the finite sample convergence guarantee for SARSA under the standard the mean square error loss.

Next, to show the performance improvement, we leverage the performance difference lemma to show our algorithm achieves the desired goal.

Lemma A.1 (Performance Difference Lemma)**.** *For any policy π, π', it holds that*

$$J(\pi) - J(\pi') = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi}}\left[\mathbb{E}_{a\sim\pi(\cdot|s)}A^{\pi'}(s,a)\right],$$

where $A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$ is the advantage function.

Similar to (Kumar et al., 2020b), we focus on the discrete case where the number of states $|\mathcal{S}|$ and actions $|\mathcal{A}|$ are finite (note in the continuous case, the $\mathcal{D}(s,a)$ would be 0 for most locations, and thus the bound becomes less interesting). The adaptation to the continuous space can leverage standard techniques like *state abstraction* (Li et al., 2006) and covering arguments.

Next, we define the learning coefficient $C_{\gamma,\delta}$ of SARSA as

$$|\hat{Q}^{\hat{\pi}_\beta}(s,a) - Q^{\hat{\pi}_\beta}(s,a)| \leq \frac{C_{\gamma,\delta}}{\sqrt{\mathcal{D}(s,a)}}, \quad \forall s,a \in \mathcal{S}\times\mathcal{A}.$$

Define the first-order approximation error as

$$\bar{Q}^{\hat{\pi}_\beta}(s,a) := (a-a_\beta)^T\left[\nabla_a \hat{Q}^{\hat{\pi}_\beta}(s,a)\right]_{a=a_\beta} + \hat{Q}^{\hat{\pi}_\beta}(s,a_\beta),$$

then the approximation error is defined as:

$$C_{\text{CFPI}}(s,a) := |\bar{Q}^{\hat{\pi}_\beta}(s,a) - \hat{Q}^{\hat{\pi}_\beta}(s,a)| = \left|(a-a_\beta)^T\left[\nabla_a \hat{Q}^{\hat{\pi}_\beta}(s,a)\right]_{a=a_\beta} + \hat{Q}^{\hat{\pi}_\beta}(s,a_\beta) - \hat{Q}^{\hat{\pi}_\beta}(s,a)\right|.$$

Under the constraint $D\left(\pi(\cdot \mid s), \hat{\pi}_\beta(\cdot \mid s)\right) \leq \delta$ (4) (or equivalently action a is close to a_β), the first-order approximation provides a good estimation for the $\hat{Q}^{\pi_\beta}$.

Theorem A.1 (Restatement of Theorem 3.1)**.** *Assume the state and action spaces are discrete. Let $\hat{\pi}_1$ be the policy obtained after the CFPI update (Line 2 of Algorithm 1). Then with probability $1-\delta$,*

$$\begin{aligned} J(\hat{\pi}_1) - J(\hat{\pi}_\beta) \geq & \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\hat{\pi}_1}}\left[\bar{Q}^{\hat{\pi}_\beta}(s,\hat{\pi}_1(s)) - \bar{Q}^{\hat{\pi}_\beta}(s,\hat{\pi}_\beta(s))\right] \\ & - \frac{2}{1-\gamma}\mathbb{E}_{s\sim d^{\hat{\pi}_1}}\mathbb{E}_{a\sim\hat{\pi}_1(\cdot|s)}\left[\frac{C_{\gamma,\delta}}{\sqrt{\mathcal{D}(s,a)}} + C_{\text{CFPI}}(s,a)\right] := \zeta. \end{aligned}$$

For multi-step T iterative update, we similarly have with probability $1-\delta$,

$$J(\hat{\pi}_T) - J(\hat{\pi}_\beta) = \sum_{t=1}^{T} J(\hat{\pi}_t) - J(\hat{\pi}_{t-1}) \geq \sum_{t=1}^{T}\zeta^{(t)},$$

where $\mathcal{D}(s,a)$ denotes number of samples at s,a, the learning coefficient of SARSA is defined as $C_{\gamma,\delta} = \max_{s_0,a_0}\sqrt{2\ln(12SA/\delta)}\cdot\sqrt{\sum_{h=0}^{\infty}\sum_{s,a}\gamma^{2h}\cdot\mu_h^{\hat{\pi}_\beta}(s,a|s_0,a_0)^2\,\mathrm{Var}\left[V^{\hat{\pi}_\beta}(s') \mid s,a\right]}$, and $C_{CFPI}(s,a)$ denotes the error from the first-order approximation (3), (4) using CFPI, i.e. $C_{\text{CFPI}}(s,a) := \left|(a-a_\beta)^T\left[\nabla_a \hat{Q}^{\hat{\pi}_\beta}(s,a)\right]_{a=a_\beta} + \hat{Q}^{\hat{\pi}_\beta}(s,a_\beta) - \hat{Q}^{\hat{\pi}_\beta}(s,a)\right|$. When $a = a_\beta$, $C_{\text{CFPI}}(s,a) = 0$.

proof of Theorem 3.1. We focus on the first update, which is from $\hat{\pi}_b$ to $\hat{\pi}_1$. According to the Sarsa update, we have $|\hat{Q}^{\hat{\pi}_\beta}(s,a) - Q^{\hat{\pi}_\beta}(s,a)| \leq \frac{C_{\gamma,\delta}}{\sqrt{\mathcal{D}(s,a)}}, \quad \forall s,a \in \mathcal{S} \times \mathcal{A}$ with probability $1-\delta$ and this is due to previous on-policy evaluation result (*e.g.* (Zou et al., 2019)). Also denote $\hat{\pi}_1 := \arg\max_\pi \bar{Q}^{\hat{\pi}_\beta}$.

By Lemma A.1,

$$
\begin{aligned}
J(\hat{\pi}_1) - J(\hat{\pi}_\beta) =& \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\hat{\pi}_1}}\left[\mathbb{E}_{a\sim\hat{\pi}_1(\cdot|s)} A^{\hat{\pi}_\beta}(s,a)\right] \\
=& \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\hat{\pi}_1}}\left[\mathbb{E}_{a\sim\hat{\pi}_1(\cdot|s)}[Q^{\hat{\pi}_\beta}(s,a) - V^{\hat{\pi}_\beta}(s)]\right] \\
=& \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\hat{\pi}_1}}\left[\mathbb{E}_{a\sim\hat{\pi}_1(\cdot|s)}[Q^{\hat{\pi}_\beta}(s,a) - Q^{\hat{\pi}_\beta}(s,\hat{\pi}_\beta(s))]\right] \\
\geq& \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\hat{\pi}_1}}\left[\mathbb{E}_{a\sim\hat{\pi}_1(\cdot|s)}[\hat{Q}^{\hat{\pi}_\beta}(s,a) - \hat{Q}^{\hat{\pi}_\beta}(s,\hat{\pi}_\beta(s))]\right] - \frac{2}{1-\gamma}\mathbb{E}_{s\sim d^{\hat{\pi}_1}}\mathbb{E}_{a\sim\hat{\pi}_1(\cdot|s)}\left[\frac{C_{\gamma,\delta}}{\sqrt{\mathcal{D}(s,a)}}\right] \\
\geq& \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\hat{\pi}_1}}\left[\bar{Q}^{\hat{\pi}_\beta}(s,\hat{\pi}_1(s)) - \bar{Q}^{\hat{\pi}_\beta}(s,\hat{\pi}_\beta(s))\right] - \frac{2}{1-\gamma}\mathbb{E}_{s\sim d^{\hat{\pi}_1}}\mathbb{E}_{a\sim\hat{\pi}_1(\cdot|s)}\left[\frac{C_{\gamma,\delta}}{\sqrt{\mathcal{D}(s,a)}} + C_{\mathrm{CFPI}}(s,a)\right] \\
:=& \zeta^{(1)}
\end{aligned}
$$

where the first inequality uses $|\hat{Q}^{\hat{\pi}_\beta}(s,a) - Q^{\hat{\pi}_\beta}(s,a)| \leq \frac{C_{\gamma,\delta}}{\sqrt{\mathcal{D}(s,a)}}$ and the last inequality uses $\hat{\pi}_1 := \arg\max_\pi \bar{Q}^{\hat{\pi}_\beta}$. Here

$$
C_{\gamma,\delta} = \max_{s_0,a_0}\sqrt{2\ln(12SA/\delta)} \cdot \sqrt{\sum_{h=0}^{\infty}\sum_{s,a}\gamma^{2h}\cdot\mu_h^{\hat{\pi}_\beta}(s,a|s_0,a_0)\,\mathrm{Var}\left[V^{\hat{\pi}_\beta}(s') \mid s,a\right]}
$$

Similarly, if the number of iteration $t > 1$, then Denote

$$
C_{\gamma,\delta}^{(t)} := \max_{s_0,a_0}\sqrt{2\ln(12SA/\delta)} \cdot \sqrt{\sum_{h=0}^{\infty}\sum_{s,a}\gamma^{2h}\cdot\frac{\mu_h^{\hat{\pi}_t}(s,a|s_0,a_0)^2}{\mu_h^{\hat{\pi}_{t-1}}(s,a|s_0,a_0)}\,\mathrm{Var}\left[V^{\hat{\pi}_t}(s') \mid s,a\right]},
$$

then we have with probability $1-\delta$, by the Corollary 1 of Duan et al. (2020), the OPE estimation follows

$$
|\hat{Q}^{\hat{\pi}_\beta}(s,a) - Q^{\hat{\pi}_\beta}(s,a)| \leq \frac{C_{\gamma,\delta}^{(t)}}{\sqrt{\mathcal{D}(s,a)}}
$$

and

$$
\begin{aligned}
J(\hat{\pi}_t) - J(\hat{\pi}_{t-1}) \geq& \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\hat{\pi}_t}}\left[\bar{Q}^{\hat{\pi}_{t-1}}(s,\hat{\pi}_t(s)) - \bar{Q}^{\hat{\pi}_{t-1}}(s,\hat{\pi}_{t-1}(s))\right] \\
&- \frac{2}{1-\gamma}\mathbb{E}_{s\sim d^{\hat{\pi}_t}}\mathbb{E}_{a\sim\hat{\pi}_t(\cdot|s)}\left[\frac{C_{\gamma,\delta}^{(t)}}{\sqrt{\mathcal{D}(s,a)}} + C_{\mathrm{CFPI}}(s,a)\right] := \zeta^{(t)},
\end{aligned}
$$

then for multi-step iterative algorithm, by a union bound, we have with probability $1-\delta$

$$
J(\hat{\pi}_T) - J(\hat{\pi}_\beta) = \sum_{t=1}^{T} J(\hat{\pi}_t) - J(\hat{\pi}_{t-1}) \geq \sum_{t=1}^{T}\zeta^{(t)}.
$$

□

On the learning coefficient of SARSA. The learning of SARSA is known to be statistically efficient from existing off-policy evaluation (OPE) literature, for instance (Duan et al., 2020; Yin & Wang, 2020). This is due to the on-policy SARSA scheme is just a special case of OPE task by choosing $\pi = \hat{\pi}_\beta$.

Concretely, we can translate the finite sample error bound in Corollary 1 of (Duan et al., 2020) to the infinite horizon discounted setting as: for any initial state,action s_0, a_0, with probability $1-\delta$,

$$|\hat{Q}^{\hat{\pi}_\beta}(s_0,a_0) - Q^{\hat{\pi}_\beta}(s_0,a_0)| \leq \frac{1}{\sqrt{\mathcal{D}(s_0,a_0)}}\sqrt{2\ln(12/\delta)} \cdot \sqrt{\sum_{h=0}^{\infty}\sum_{s,a}\gamma^{2h}\cdot\mu_h^{\hat{\pi}_\beta}(s,a|s_0,a_0)\,\mathrm{Var}\left[V^{\hat{\pi}_\beta}(s') \mid s,a\right]}$$

Note the original statement in (Duan et al., 2020) is for $v^{\hat{\pi}_\beta} - \hat{v}^{\hat{\pi}_\beta}$, here we conduct the version for $\hat{Q}^{\hat{\pi}_\beta} - Q^{\hat{\pi}_\beta}$ instead and this can be readily obtained by fixing the initial state action s_0, a_0 for v^π. As a result, by a union bound (over S, A) it is valid to define

$$C_{\gamma,\delta} = \max_{s_0,a_0}\sqrt{2\ln(12SA/\delta)} \cdot \sqrt{\sum_{h=0}^{\infty}\sum_{s,a}\gamma^{2h}\cdot\mu_h^{\hat{\pi}_\beta}(s,a|s_0,a_0)\,\mathrm{Var}\left[V^{\hat{\pi}_\beta}(s') \mid s,a\right]}$$

and this makes sure the statistical guarantee in Theorem 3.1 follows through.

Similarly, for the multi-step case, the OPE estimator hold with the corresponding coefficient

$$C_{\gamma,\delta}^{(t)} := \max_{s_0,a_0}\sqrt{2\ln(12SA/\delta)} \cdot \sqrt{\sum_{h=0}^{\infty}\sum_{s,a}\gamma^{2h}\cdot\frac{\mu_h^{\hat{\pi}_t}(s,a|s_0,a_0)^2}{\mu_h^{\hat{\pi}_{t-1}}(s,a|s_0,a_0)}\,\mathrm{Var}\left[V^{\hat{\pi}_t}(s') \mid s,a\right]}.$$

Lastly, even the assumption on the state-action space to be finite is not essential for Theorem 3.1 since, for more general function approximations, recent literature for OPE (Zhang et al., 2022) shows SARSA update in Algorithm 1 is still statistically efficient.

B DETAILED PROCEDURES TO OBTAIN EQUATION 15

We first highlight that we set the HP δ differently for Proposition 3.2 and 3.3. With the same τ, we generate the two different δ for the two different settings. Specifically,

$$\begin{aligned} \delta_{\text{lse}}(\tau) &= \log\tau + \min_i \left\{ \frac{1}{2}\log\det(2\pi\Sigma_i) - \log\lambda_i \right\}, \quad \text{(Proposition 3.2)} \\ \delta_{\text{jensen}}(\tau) &= \log\tau + \frac{1}{2}\sum_{i=1}^{N} \lambda_i \log\det(2\pi\Sigma_i), \quad \text{(Proposition 3.3)} \end{aligned} \tag{47}$$

We next provide intuition for the design choices (47). Recall that the Gaussian Mixture behavior policy is constructed by

$$\pi_\beta = \sum_{i=1}^{N} \lambda_i \mathcal{N}(\mu_i, \Sigma_i). \tag{48}$$

With the mixture weights $\lambda_{i=1\dots N}$, we define the scaled probability $\breve{\pi}_i(a)$ of the i-th Gaussian component evaluated at a

$$\breve{\pi}_i(\mu_i) = \lambda_i \pi_i(a) = \lambda_i \det(2\pi\Sigma_i)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(a-\mu_i)^T \Sigma_i^{-1}(a-\mu_i)\}, \tag{49}$$

where $\pi_i(a) = \mathcal{N}(a; \mu_i, \Sigma_i)$ denotes the probability of the i-th Gaussian component evaluated at a. Therefore, we can have $\log\breve{\pi}_i(\mu_i) = \log\lambda_i - \frac{1}{2}\log\det(2\pi\Sigma_i)$, which implies that

$$\begin{aligned} \delta_{\text{lse}}(\tau) &= \log\tau + \min_i \left\{ \frac{1}{2}\log\det(2\pi\Sigma_i) - \log\lambda_i \right\} \\ &= -\left(\max_i \left\{ \log\lambda_i - \frac{1}{2}\log\det(2\pi\Sigma_i) \right\} - \log\tau \right). \\ &= -\max_i \left\{ \log\frac{1}{\tau}\breve{\pi}_i(\mu_i) \right\}. \end{aligned} \tag{50}$$

By setting $\delta_{\text{lse}}(\tau)$ in this way, $\bar{\mu}_j = \bar{\mu}_j(\delta_{\text{lse}}(\tau))$ will satisfy the following condition whenever $\bar{\mu}_j$ is a valid solution to the sub-problem j (28) due to the KKT conditions, $\forall j \in \{1, \dots, N\}$.

$$\begin{aligned} & -\frac{1}{2}(\bar{\mu}_j - \mu_j)^T \Sigma_j^{-1}(\bar{\mu}_j - \mu_j) - \frac{1}{2}\log\det(2\pi\Sigma_j) + \log\lambda_j = -\delta_{\text{lse}}(\tau) \\ \iff \quad & \log\breve{\pi}_j(\bar{\mu}_j) = \max_i \left\{ \log\frac{1}{\tau}\breve{\pi}_i(\mu_i) \right\} \quad \iff \quad \breve{\pi}_j(\bar{\mu}_j) = \frac{1}{\tau}\max_i \{\breve{\pi}_i(\mu_i)\} \end{aligned} \tag{51}$$

To elaborate the design of $\delta_{\text{jensen}}(\tau)$, we first recall that the constraint of problem (13) is given by

$$\sum_{i=1}^{N} \lambda_i \left(-\frac{1}{2}\log\det(2\pi\Sigma_i) - \frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1}(\mu-\mu_i) \right) \geq -\delta_{\text{jensen}}(\tau). \tag{52}$$

Note that the LHS of (52) is a concave function w.r.t μ. Thus, we can obtain its maximum by setting its derivatives (53) to zero

$$\begin{aligned} & \nabla_\mu \left(\sum_{i=1}^{N} \lambda_i \left(-\frac{1}{2}\log\det(2\pi\Sigma_i) - \frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1}(\mu-\mu_i) \right) \right) \\ = & -\sum_{i=1}^{N} \lambda_i \Sigma_i^{-1}(\mu-\mu_i) = -\overline{\Sigma}^{-1}\mu + \overline{\Sigma}^{-1}\bar{\mu} \end{aligned}. \tag{53}$$

Interestingly, we can find that the solution is given by $\mu = \bar{\mu}$. Plugging $\mu = \bar{\mu}$ into the LHS of (52), we can obtain its maximum as below

$$\begin{aligned} & -\frac{1}{2}\sum_{i=1}^{N} \lambda_i \log\det(2\pi\Sigma_i) - \frac{1}{2}\sum_{i=1}^{N} \lambda_i (\bar{\mu}-\mu_i)^T \Sigma_i^{-1}(\bar{\mu}-\mu_i) \\ \leq & \sum_{i=1}^{N} \lambda_i \left(-\frac{1}{2}\log\det(2\pi\Sigma_i) \right) = \sum_{i=1}^{N} \lambda_i \log\pi_i(\mu_i) \end{aligned} \tag{54}$$

The inequality holds as the covariance matrix Σ_i is a positive semi-definite matrix for $i \in \{1 \dots N\}$. Therefore, our choice of $\delta_{\text{jensen}}(\tau)$ can be interpreted as

$$\delta_{\text{jensen}}(\tau) = \log \tau + \frac{1}{2} \sum_{i=1}^{N} \lambda_i \log \det(2\pi\Sigma_i) = -(\sum_{i=1}^{N} \lambda_i \log \pi_i(\mu_i) - \log \tau) \tag{55}$$

Algorithm 2 Iterative $\mathcal{I}_{\text{MG}}$

1: **Input**: Learned behavior policy $\hat{\pi}_\beta$, Q network parameters ϕ_1, ϕ_2, target Q network parameters $\phi_{\text{targ},1}$, $\phi_{\text{targ},2}$, dataset $\mathcal{D}$, parameter τ
2: **repeat**
3: Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
4: Compute target actions

$$a'(s') = \text{clip}\left(\mathcal{I}_{\text{MG}}(\hat{\pi}_\beta, \hat{Q}; \tau)(s') + \text{clip}(\epsilon, -c, c), a_{Low}, a_{High}\right),$$
$$\text{where } \hat{Q} = \min(Q_{\phi_1}, Q_{\phi_2}), \text{ and } \epsilon \sim \mathcal{N}(0, \sigma)$$

5: Compute targets

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', a'(s'))$$

6: Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d)\in B} \left(Q_{\phi_i}(s, a) - y(r, s', d)\right)^2 \qquad \text{for } i = 1, 2$$

7: Update target networks with

$$\phi_{\text{targ},i} \leftarrow \rho\phi_{\text{targ},i} + (1 - \rho)\phi_i \qquad \text{for } i = 1, 2$$

8: **until** convergence
9: **Output**: $\mathcal{I}_{\text{MG}}(\hat{\pi}_\beta, \hat{Q}; \tau)$

C Multi-step and iterative algorithms

By setting $T > 0$, we can derive multi-step and iterative algorithms. Thanks to the tractability of our CFPI operators $\mathcal{I}_{\text{SG}}$ and $\mathcal{I}_{\text{MG}}$, we can always perform the policy improvement step in-closed form. Therefore, there is no significant gap between multi-step and iterative algorithms with our CFPI operators. One can differentiate our multi-step and iterative algorithms by whether an algorithm trains the policy evaluation step $\mathcal{E}(\hat{Q}_{t-1}, \hat{\pi}_t, \mathcal{D})$ to convergence or not.

As for the policy evaluation operator $\mathcal{E}$, the fitted Q evaluation (Ernst et al., 2005; Le et al., 2019; Fujimoto et al., 2022) with a target network (Mnih et al., 2015) has been demonstrated to be an effective and successful paradigm to perform policy evaluation (Kumar et al., 2019; Fujimoto & Gu, 2021; Haarnoja et al., 2018; Lillicrap et al., 2015; Fujimoto et al., 2018) in deep (offline) RL. When instantiating a multi-step or iterative algorithm from Algorithm 1, one can also consider the other policy evaluation operators by incorporating more optimization techniques.

In the rest of this section, we will instantiate an iterative algorithm with our CFPI operators performing the policy improvement step and evaluate its effectiveness on the challenging AntMaze domains.

C.1 Iterative algorithm with our CFPI operators

In Sec. 5.1, we instantiate an iterative algorithm *Iterative* $\mathcal{I}_{MG}$ with our CFPI operator $\mathcal{I}_{\text{MG}}$. Algorithm 2 presents the corresponding pseudo-codes that learn a set of Q-function networks for simplicity. Without loss of generality, we can easily generalize the algorithm to learn the action-value distribution $Z(s, a)$ as is defined in (58).

For each task, we learn a Gaussian Mixture behavior policy $\hat{\pi}_\beta$ with behavior cloning. Similar to Sec. 5.1, we employed the IQN (Dabney et al., 2018a) architecture to model the Q-value network for its better generalizability. As our CFPI operator $\mathcal{I}_{\text{MG}}$ returns a deterministic policy, we follow the TD3 (Fujimoto et al., 2018) to perform policy smoothing by adding noise to the action $a'(s')$ in Line 4. After convergence, Algorithm 2 outputs an improved policy $\mathcal{I}_{\text{MG}}(\hat{\pi}_\beta, \hat{Q}; \tau)$.

Table 5 compares our Iterative $\mathcal{I}_{\text{MG}}$ with SOTA algorithms on the AntMaze domain. The performance for all baseline methods is directly reported from the IQL paper (Kostrikov et al., 2021). Our method

Table 5: Comparison between our iterative algorithm and SOTA methods on the AntMaze domain of D4RL. We report the mean and standard deviation across 5 seeds for our methods. Our Iterative $\mathcal{I}_{\text{MG}}$ outperforms all baselines on 5 out of 6 tasks and obtaining the best overall performance, demonstrating the effectiveness of our CFPI operator when instantiating an iterative algorithm. "u" stands for umaze, "m" stands for medium, "l" stands for large, "p" stands for play, and "d" stands for diverse.

Dataset	BC	DT	Onestep RL	TD3+BC	CQL	IQL	Iterative $\mathcal{I}_{\text{MG}}$
antmaze-u-v0	54.6	59.2	64.3	78.6	74.0	87.5	$\mathbf{90.2 \pm 3.9}$
antmaze-u-d-v0	45.6	49.3	60.7	71.4	**84.0**	62.2	58.6 ± 15.2
antmaze-m-p-v0	0.0	0.0	0.3	10.6	61.2	71.2	$\mathbf{75.2 \pm 6.9}$
antmaze-m-d-v0	0.0	0.7	0.0	3.0	53.7	70.0	$\mathbf{72.2 \pm 7.3}$
antmaze-l-p-v0	0.0	0.0	0.0	0.2	15.8	39.6	$\mathbf{51.4 \pm 7.7}$
antmaze-l-d-v0	0.0	1.0	0.0	0.0	14.9	47.5	$\mathbf{52.4 \pm 10.9}$
Total	100.2	112.2	125.3	163.8	303.6	378.0	$\mathbf{400.0 \pm 52.0}$

outperforms all baseline methods on on 5 out of 6 tasks and obtaining the best overall performance. The training curves are shown in Fig. 3 with the HP settings detailed in Table 6. We did not perform much HP tuning, and thus one should expect a performance improvement after conducting fine-grained HP tuning.

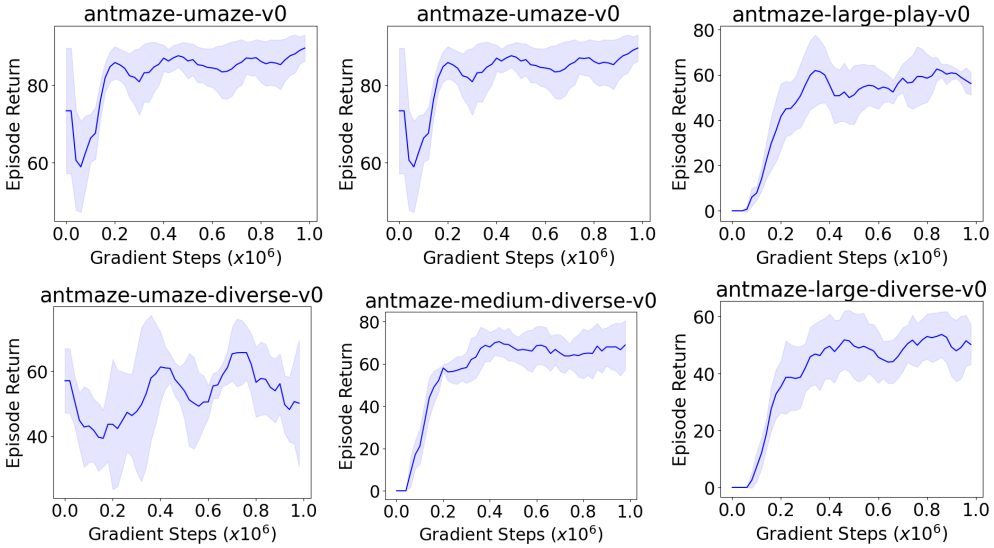


Figure 3: Iterative $\mathcal{I}_{\text{MG}}$ training results on AntMaze. Shaded area denotes one standard deviation.

	Hyperparameter	Value
Shared HP	Optimizer	Adam (Kingma & Ba, 2014)
	Normalize states	False
	activation function	ReLU
	Mini-batch size	256
MG-BC	Gaussian components (N)	8
	Number of gradient steps	500K
	Policy architecture	MLP
	Policy learning rate	1e-4
	Policy hidden layers	3
	Policy hidden dim	256
	Threshold ξ in (16)	0.05
Iterative $\mathcal{I}_{\text{MG}}$	Number of gradient steps	1M
	Critic architecture	IQN (Dabney et al., 2018a)
	Critic hidden dim	256
	Critic hidden layers	3
	Critic learning rate	3e-4
	Number of quantiles N_{q}	8
	Number of cosine basis elements	64
	Discount factor	0.99
	Target update rate	5e-3
	Target update period	1
	$\log \tau$	1.5

Table 6: Hyperparameters for our Iterative $\mathcal{I}_{\text{MG}}$.

D CFPI BEYOND GAUSSIAN POLICIES

In the main paper, we mainly discuss the scenario when the behavior policy π_β is from the Gaussian family and develop two CFPI operators. However, our methods can also work with a non-Gaussian π_β. Next, we derive a new CFPI operator $\mathcal{I}_{\text{DET}}$ that can work with deterministic π_β. We then show that $\mathcal{I}_{\text{DET}}$ can also be leveraged to improve a general stochastic policy π_β without knowing its actual expression, as long as we can sample from it.

D.1 DETERMINISTIC BEHAVIOR POLICY

When modeling both $\pi = \mu$ and $\pi_\beta = \mu_\beta$ as deterministic policies, we can derive the following BCPO from the problem (4) by setting $D(\cdot, \cdot)$ as the mean squared error.

$$\max_{\mu} \quad \mu^T \left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}, \quad \text{s.t.} \quad \frac{1}{2}\|\mu - \mu_\beta\|^2 \leq \delta. \tag{56}$$

Problem (56) has a similar form as the problem (18). We can thus obtain its closed-form solution $\mu = \mu_{\text{det}}(\delta)$ as below

$$\mu_{\text{det}}(\delta) = \mu_\beta + \frac{\sqrt{2\delta}}{\| \left[\nabla_a Q(s,a)\right]_{a=\mu_\beta} \|} \left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}. \tag{57}$$

Therefore, we can derive a new CFPI operator $\mathcal{I}_{\text{DET}}(\pi_\beta, Q; \delta)$ that returns a policy with action selected by (57).

We further note that the problem (56) can be seen as a linear approximation of the objectives used in TD3 + BC (Fujimoto & Gu, 2021).

D.2 BEYOND DETERMINISTIC BEHAVIOR POLICY

Though we assume π_β to be a deterministic policy during the derivation of $\mathcal{I}_{\text{DET}}$, we can indeed leverage $\mathcal{I}_{\text{DET}}$ to tackle the more general case when we can only sample from π_β without knowing its actual expression.

Algorithm 3 Policy improvement of $\mathcal{I}_{\text{DET}}$ with a stochastic π_β

Input: State s, stochastic policy π_β, value function $\hat{Q}$, δ, number of candidate actions to sample M

1: Sample candidate actions $\{a_1, \ldots, a_M\}$ from π_β
2: Obtain the EBCQ policy π_{EBCQ} with action selected by $\pi_{\text{EBCQ}}(s) = \arg\max_{m=1\ldots M} \hat{Q}(s, a_m)$
3: Return $\mathcal{I}_{\text{DET}}(\pi_{\text{EBCQ}}, \hat{Q}; \delta)$ by calculating (57)

Table 7: $\mathcal{I}_{\text{DET}}$ results on the Gym-MuJoCo domain. We report the mean and standard deviation 5 seeds and each seed evaluates for 100 episodes.

Dataset	DET-BC	VAE-BC	VAE-EBCQ	$\mathcal{I}_{\text{DET}}$ with π_{det}	$\mathcal{I}_{\text{DET}}$ with π_{vae}
Walker2d-Medium-v2	71.2 ± 2.0	70.6 ± 3.0	70.6 ± 3.4	79.5 ± 12.9	$\mathbf{86.5 \pm 6.3}$
Walker2d-Medium-Replay-v2	19.5 ± 12.6	19.4 ± 2.9	33.5 ± 7.3	57.1 ± 11.6	$\mathbf{62.6 \pm 7.1}$
Walker2d-Medium-Expert-v2	74.4 ± 0.4	74.9 ± 7.6	82.7 ± 11.9	$\mathbf{111.2 \pm 1.8}$	$\mathbf{111.1 \pm 0.9}$

Algorithm 3 details the procedures to perform the policy improvement step for a stochastic behavior policy π_β. We first obtain its EBCQ policy π_{EBCQ} in Line 1-2. As π_{EBCQ} is deterministic, we further plug it in $\mathcal{I}_{\text{DET}}$ in Line 3 to return an improved policy.

D.3 Experiment results

To evaluate the performance of $\mathcal{I}_{\text{DET}}$, we first learn two behavior policies with two different models. Specifically, we model π_{det} with a three-layer MLP that outputs a deterministic policy and π_{vae} with the Variational auto-encoder (VAE) (Kingma & Welling, 2013) from BCQ (Fujimoto et al., 2019). Moreover, we reused the same value function $\hat{Q}_0$ as in Section 5.1. We present the results in Table 7. DET-BC and VAE denote the performance of π_{det} and π_{vae}, respectively. VAE-EBCQ denotes the EBCQ performance of π_{vae} with $M = 50$ candidate actions. Since π_{det} is deterministic, its EBCQ performance is the same as DET-BC. As for our two methods, we set $\delta = 0.1$ for all datasets. We can observe that both our $\mathcal{I}_{\text{DET}}$ with π_{det} and $\mathcal{I}_{\text{DET}}$ with π_{vae} largely improve over the baseline methods. Moreover, $\mathcal{I}_{\text{DET}}$ with π_{vae} outperforms VAE-EBCQ by a significant margins on all three datasets, demonstrating the effectiveness of our CFPI operator.

Indeed, our method benefits from an accurate and expressive behavior policy, as $\mathcal{I}_{\text{DET}}$ with π_{vae} achieves a higher average performance compared to $\mathcal{I}_{\text{DET}}$ with π_{det}, while maintaining a lower standard deviation on all three datasets.

We also note that we did not spend too much effort optimizing the HP, e.g., the VAE architectures, learning rates, and the value of τ.

E RELIABLE EVALUATION TO ADDRESS THE STATISTICAL UNCERTAINTY

Figure 4: Comparison between our methods and baselines using reliable evaluation methods proposed in (Agarwal et al., 2021). We re-examine the results in Table 4 on the 9 tasks from the D4RL MuJoCo Gym domain. Each metric is calculated with a 95% CI bootstrap based on 9 tasks and 10 seeds for each task. Each seed further evaluates each method for 100 episodes. The interquartile mean (IQM) discards the top and bottom 25% data points and calculates the mean across the remaining 50% runs. The IQM is more robust as an estimator to outliers than the mean while maintaining less variance than the median. Higher median, IQM, mean scores, and lower Optimality Gap correspond to better performance. Our $\mathcal{I}_{\text{MG}}$ outperforms the baseline methods by a significant margin based on all four metrics.

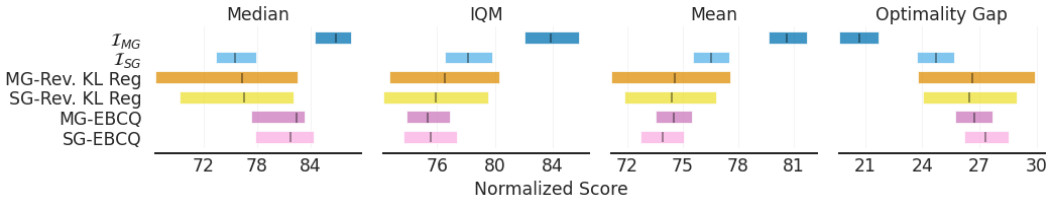
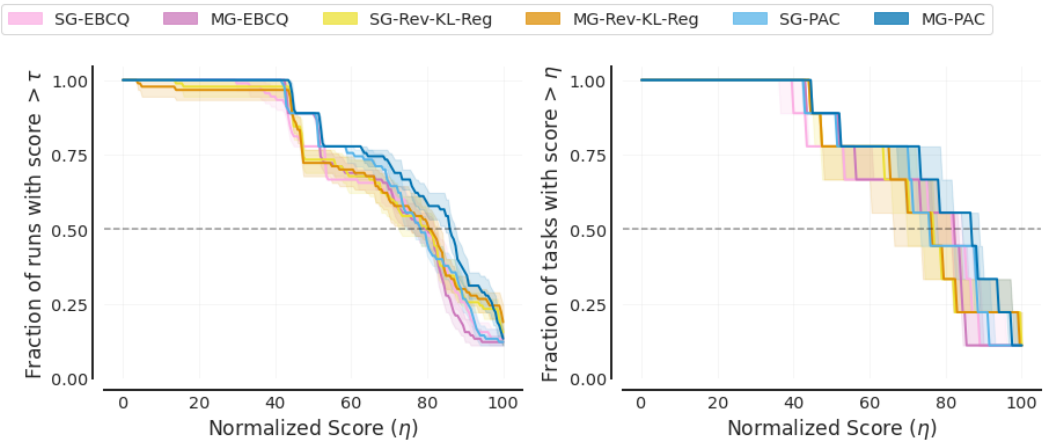


Figure 5: Performance profiles (score distributions) for all methods on the 9 tasks from the D4RL MuJoCo Gym domain. The average score is calculated by averaging all runs within one task. Each task contains 10 seeds, and each seed evaluates for 100 episodes. Shaded area denotes 95% confidence bands based on percentile bootstrap and stratified sampling (Agarwal et al., 2021). The η value where the curves intersect with the dashed horizontal line $y = 0.5$ corresponds to the median, while the area under the performance curves corresponds to the mean.



To demonstrate the superiority of our methods over the baselines and provide reliable evaluation results, we follow the evaluation protocols proposed in (Agarwal et al., 2021) to re-examine the results in Table 4. Specifically, we adopt the evaluation methods for all methods with $N_{\text{tasks}} \times N_{\text{seeds}}$ runs in total.

Moreover, we obtain the performance profile of each method, revealing its score distribution and variability. In particular, the score distribution shows the fraction of runs above a certain threshold η and is given by

$$\hat{F}(\eta) = \hat{F}\left(\eta; x_{1:N_{\text{tasks}},1:N_{\text{seeds}}}\right) = \frac{1}{N_{\text{tasks}}} \sum_{m=1}^{N_{\text{tasks}}} \frac{1}{N_{\text{seeds}}} \sum_{n=1}^{N_{\text{seeds}}} \mathbb{1}\left[x_{m,n} \geq \eta\right]$$

Evaluation results in Fig. 4 and Fig. 5 demonstrate that our $\mathcal{I}_{\text{MG}}$ outperforms the baseline methods by a significant margin based on all four reliable metrics.

F HYPER-PARAMETER SETTINGS AND TRAINING DETAILS

For all methods we proposed in Table 1, Table 3, and Table 4, we obtain the mean and standard deviation of each method across 10 seeds. Each seed contains individual training process and evaluates the policy for 100 episodes.

F.1 HP AND TRAINING DETAILS FOR METHODS IN TABLE 1 AND TABLE 4

Table 8 includes the HP of methods evaluated on the Gym-MuJoCo domain. We use the Adam (Kingma & Ba, 2014) optimizer for all learning algorithms and normalize the states in each dataset following the practice of TD3+BC (Fujimoto & Gu, 2021). Note that our one-step offline RL algorithms presented in Table 1 (Our $\mathcal{I}_{\text{MG}}$) and Table 4 ($\mathcal{I}_{\text{MG}}$, $\mathcal{I}_{\text{SG}}$, MG-EBCQ, SG-EBCQ, MG-MS) require learning a behavior policy and the value function $\hat{Q}^0$. Therefore, we will first describe the detailed procedures for learning Single Gaussian (SG-BC) and Gaussian Mixture (MG-BC) behavior policies. We next describe our SARSA-style training procedures to estimate $\hat{Q}^0$. Finally, we will present the details for each one-step algorithm.

	Hyperparameter	Value
Shared HP	Optimizer	Adam (Kingma & Ba, 2014)
	Normalize states	True
	Policy architecture	MLP
	Policy learning rate	1e-4
	Policy hidden layers	3
	Policy hidden dim	256
	Policy activation function	ReLU
	Threshold ξ in (16)	0.05
MG-BC	Gaussian components (N)	4
	Number of gradient steps	500K
	Mini-batch size	256
SG-BC	Number of gradient steps	500K
	Mini-batch size	512
SARSA	Number of gradient steps	Table 16
	Critic architecture	IQN (Dabney et al., 2018a)
	Critic hidden dim	256
	Critic hidden layers	3
	Critic activation function	ReLU
	Number of quantiles N_{q}	8
	Number of cosine basis elements	64
	Discount factor	0.99
	Target update rate	5e-3
	Target update period	1
Our $\mathcal{I}_{\text{MG}}$ (Table 1)	$\log \tau$	0 for Hopper-M-E; 0.5 for the others
$\mathcal{I}_{\text{MG}}$ & $\mathcal{I}_{\text{SG}}$(Table 4)	$\log \tau$	0.5 for all tasks
MG-EBCQ	Number of candidate actions N_{bcq}	5
SG-EBCQ	Number of candidate actions N_{bcq}	10
MG-Rev. KL Reg & SG-Rev. KL Reg	α Number of gradient steps	3.0 100K

Table 8: Hyperparameters for our methods in Table 1 and Table 4.

MG-BC. We parameterize the policy as a 3-layer MLP, which outputs the tanh of a Gaussian Mixture with $N = 4$ Gaussian components. For each Gaussian component, we learn the state-dependent diagonal covariance matrix. While existing methods suggest learning Gaussian Mixture

via expectation maximization (Jordan & Jacobs, 1994; Xu et al., 1994; Jin et al., 2016) or variational Bayes (Bishop & Svensén, 2012), we empirically find that directly minimizing the negative log-likelihood of actions sampled from the offline datasets achieves satisfactory performance, as is shown in Table 1. We train the policy for 500K gradient steps. We emphasize that we do not aim to propose a better algorithm for learning a Gaussian Mixture behavior policy. Instead, future work may use a more advanced algorithm to capture the underlying behavior policy better.

SG-BC. We parameterize the policy as a 3-layer MLP, which outputs the tanh of a Single Gaussian with the state-dependent diagonal covariance matrix (Fu et al., 2020; Haarnoja et al., 2018). We train the policy for 500K gradient steps.

SARSA. We parameterize the value function with the IQN (Dabney et al., 2018a) architecture and train it to model the distribution $Z^\beta : \mathcal{S} \times \mathcal{A} \to \mathcal{Z}$ of the behavior return via quantile regression, where $\mathcal{Z}$ is the action-value distributional space (Ma et al., 2020) defined as

$$\mathcal{Z} = \{Z : \mathcal{S} \times \mathcal{A} \to \mathscr{P}(\mathbb{R}) \mid \mathbb{E}\left[|Z(s,a)|^p\right] < \infty, \forall (s,a), p \geq 1\}. \tag{58}$$

We define the CDF function of Z^β as $F_{Z^\beta}(z) = Pr(Z^\beta < z)$, leading to the quantile function (Müller, 1997) $F^{-1}_{Z^\beta}(\rho) := \inf\{z \in \mathbb{R} : \rho \leq F_{Z^\beta}(z)\}$ as the inverse CDF function, where ρ denotes the quantile fraction. We further denote $Z^\beta_\rho = F^{-1}_{Z^\beta}(\rho)$ to ease the notation.

To obtain Z^β, we leverage the empirical distributional bellman operator $\hat{\mathcal{T}}^\beta_D : \mathcal{Z} \to \mathcal{Z}$ defined as

$$\hat{\mathcal{T}}^\beta_D Z(s,a) :\overset{D}{=} r + \gamma Z\left(s', a'\right) \mid (s,a,r,s',a') \sim \mathcal{D}, \tag{59}$$

where $A :\overset{D}{=} B$ implies the random variables A and B are governed by the same distribution. We note that $\hat{\mathcal{T}}^\beta_D$ helps to construct a Huber quantile regression loss (Dabney et al., 2018a; Ma et al., 2020; Dabney et al., 2018b), and we can finally learn Z^β by minimizing the quantile regression loss following a similar procedures as in (Ma et al., 2020).

To achieve the goal, we approximate Z^β by N_q quantile fractions $\{\rho_i \in [0,1] \mid i = 0 \ldots N_\text{q}\}$ with $\rho_0 = 0$, $\rho_{N_\text{q}} = 1$ and $\rho_i < \rho_j, \forall i < j$. We further denote $\hat{\rho}_i = (\rho_i + \rho_{i+1})/2$, and use random sampling (Dabney et al., 2018a) to generate the quantile fractions. By further parameterizing $Z^\beta_\rho(s,a)$ as $\hat{Z}^\beta_\rho(s,a;\theta)$ with parameter θ, we can derive the loss function $J_Z(\theta)$ as

$$\begin{aligned} & J_Z(\theta) = \mathbb{E}_{(s,a,r,s',a')\sim\mathcal{D}}\left[\sum_{i=0}^{N_\text{q}-1}\sum_{j=0}^{N_\text{q}-1}\left(\rho_{i+1}-\rho_i\right) l_{\hat{\rho}_j}\left(\delta_{ij}\right)\right], \\ \text{where} \quad & \delta_{ij} = \delta_{ij}\left(s,a,r,s',a'\right) = r + \gamma Z_{\hat{\rho}_i}\left(s',a';\bar{\theta}\right) - Z_{\hat{\rho}_j}\left(s,a;\theta\right) \\ \text{and} \quad & l_\rho\left(\delta_{ij}\right) = \left|\rho - \mathbb{I}\left\{\delta_{ij} < 0\right\}\right| \mathcal{L}\left(\delta_{ij}\right), \text{ with } \mathcal{L}\left(\delta_{ij}\right) = \begin{cases} \frac{1}{2}\delta_{ij}^2, & \text{if } |\delta_{ij}| \leq 1 \\ |\delta_{ij}| - \frac{1}{2}, & \text{otherwise.} \end{cases} \end{aligned} . \tag{60}$$

$\bar{\theta}$ is the parameter of the target network (Lillicrap et al., 2015) given by the Polyak averaging of θ. We refer interested readers to (Dabney et al., 2018a; Ma et al., 2020) for further details.

The training procedures above returns $\hat{Z}^\beta_\rho, \forall \rho \in [0,1]$. With the learned $\hat{Z}^\beta_\rho$, our one-step methods presented in Table 1 and Table 4 extract the value function by setting $\hat{Q}_0 = \mathbb{E}_\rho[\hat{Z}^\beta_\rho] = \hat{Q}^\beta$ as the expectation of $\hat{Z}^\beta_\rho$, which is equivalent to the conventional action-value function $\hat{Q}^\beta$. Specifically, we use $N = 32$ fixed quantile fractions with $\rho_i = i/N,\ i = 0 \ldots N$. Given a state-action pair (s,a), we calculate $\hat{Q}_0(s,a) = \hat{Q}^\beta(s,a)$ as

$$\hat{Q}_0(s,a) = \hat{Q}^\beta(s,a) = \frac{1}{N}\sum_{i=1}^{N} \hat{Z}^\beta_{\hat{\rho}_i}(s,a), \quad \hat{\rho}_i = \frac{\rho_i + \rho_{i-1}}{2}. \tag{61}$$

Since our methods still need to query out-of-buffer action values during rollout, we employed the conventional double Q-learning (Fujimoto et al., 2018) technique to prevent potential overestimation without clipping. Specifically, we initialize $\hat{Q}^1_0$ and $\hat{Q}^2_0$ differently and train them to minimize (60). With the learned $\hat{Q}^1_0$ and $\hat{Q}^2_0$, we set the value of $\hat{Q}_0(s,a)$ as

$$\hat{Q}_0(s,a) = \min_{k=1,2} \hat{Q}^k_0(s,a) \tag{62}$$

Table 9: HP search for MG-EBCQ. We report the mean and std of 10 seeds, and each seed evaluates for 100 episodes.

Dataset	$N_{\text{bcq}} = 2$	$N_{\text{bcq}} = 5$	$N_{\text{bcq}} = 10$	$N_{\text{bcq}} = 20$	$N_{\text{bcq}} = 50$	$N_{\text{bcq}} = 100$
Cheetah-M-v2	47.2 ± 0.3	51.5 ± 0.2	53.3 ± 0.3	54.4 ± 0.3	55.3 ± 0.4	55.8 ± 0.4
Hopper-M-v2	63.3 ± 2.3	82.5 ± 1.9	88.3 ± 4.6	90.8 ± 6.9	92.1 ± 7.6	91.3 ± 9.4
Walker2d-M-v2	78.6 ± 1.4	85.2 ± 2.1	81.0 ± 6.3	73.4 ± 11.0	67.6 ± 14.8	62.7 ± 15.8
Cheetah-M-R-v2	38.8 ± 0.6	43.0 ± 0.3	44.2 ± 0.3	44.7 ± 0.5	44.8 ± 0.8	44.6 ± 0.8
Hopper-M-R-v2	58.4 ± 6.9	83.6 ± 10.3	82.8 ± 14.9	82.3 ± 15.9	77.5 ± 17.7	76.0 ± 16.7
Walker2d-M-R-v2	55.1 ± 3.4	73.1 ± 5.2	75.6 ± 5.3	77.6 ± 5.4	78.0 ± 5.6	78.5 ± 4.5
Cheetah-M-E-v2	75.2 ± 3.2	84.5 ± 4.6	82.7 ± 5.2	77.6 ± 7.3	73.4 ± 6.4	68.8 ± 5.9
Hopper-M-E-v2	73.6 ± 7.5	56.1 ± 6.2	44.9 ± 4.6	37.3 ± 3.6	29.8 ± 2.9	25.3 ± 3.3
Walker2d-M-E-v2	107.1 ± 1.8	111.1 ± 1.0	111.4 ± 1.5	111.4 ± 2.5	109.6 ± 4.0	107.2 ± 6.0
Total	597.2 ± 27.4	670.6 ± 31.9	664.1 ± 43.1	649.5 ± 53.5	628.0 ± 60.2	610.2 ± 62.9

Table 10: HP search for SG-EBCQ. We report the mean and std of 10 seeds, and each seed evaluates for 100 episodes.

Dataset	$N_{\text{bcq}} = 2$	$N_{\text{bcq}} = 5$	$N_{\text{bcq}} = 10$	$N_{\text{bcq}} = 20$	$N_{\text{bcq}} = 50$	$N_{\text{bcq}} = 100$
Cheetah-M-v2	47.1 ± 0.2	51.5 ± 0.1	53.3 ± 0.2	54.4 ± 0.3	55.3 ± 0.3	55.8 ± 0.4
Hopper-M-v2	60.7 ± 2.4	78.6 ± 4.0	86.8 ± 5.2	89.1 ± 7.7	89.8 ± 8.8	89.8 ± 9.8
Walker2d-M-v2	78.5 ± 2.8	86.9 ± 1.8	85.2 ± 5.1	81.5 ± 9.3	76.6 ± 11.8	72.4 ± 13.8
Cheetah-M-R-v2	37.8 ± 0.7	42.3 ± 0.6	43.5 ± 0.6	44.3 ± 0.7	44.1 ± 1.1	43.6 ± 0.9
Hopper-M-R-v2	58.7 ± 5.8	85.2 ± 9.0	88.5 ± 12.2	89.1 ± 11.7	83.9 ± 15.0	82.1 ± 16.1
Walker2d-M-R-v2	54.0 ± 7.2	72.2 ± 5.2	75.4 ± 4.6	77.7 ± 4.8	77.5 ± 5.8	74.9 ± 6.2
Cheetah-M-E-v2	71.8 ± 2.2	81.9 ± 4.8	81.8 ± 5.4	77.6 ± 6.9	71.5 ± 7.5	68.2 ± 6.5
Hopper-M-E-v2	66.4 ± 4.8	49.8 ± 6.2	40.0 ± 5.8	34.9 ± 6.2	29.0 ± 5.7	25.2 ± 4.8
Walker2d-M-E-v2	106.6 ± 1.6	111.0 ± 0.9	111.1 ± 1.8	110.0 ± 3.7	107.2 ± 7.8	106.0 ± 9.0
Total	581.6 ± 27.7	659.4 ± 32.7	665.5 ± 41.0	658.7 ± 51.3	634.7 ± 63.9	618.1 ± 67.5

for every (s, a) pair. Note that the double Q-learning technique is only used during policy evaluation.

As for deciding the number of gradient steps, we detail our procedures in Appendix G.5. And the number of gradient steps for each dataset can be found in Table 16.

Our $\mathcal{I}_{\text{MG}}$ (Table 1). Recall that our CFPI operator $\mathcal{I}_{\text{MG}}(\hat{\pi}_\beta, \hat{Q}_0; \tau)$ requires to learn a Gaussian Mixture behavior policy $\hat{\pi}_\beta$ and a value function $\hat{Q}_0$. We train $\hat{\pi}_\beta$ and $\hat{Q}_0$ according to the procedures listed in **MG-BC** and **SARSA**, respectively. By following the practice of (Brandfonbrener et al., 2021; Fu et al., 2020), we perform a grid search on $\log \tau \in \{0, 0.5, 1.0, 1.5, 2.0\}$ using 3 seeds. We note that we manually reduce $\mathcal{I}_{\text{MG}}$ to MG-MS when $\log \tau = 0$ by only considering the mean of each non-trivial Gaussian component. Our results show that setting $\log \tau = 0.5$ achieves the best overall performance while Hopper-M-E requires an extremely small $\log \tau$ to perform well as is shown in Appendix G.2. Therefore, we decide to set $\log \tau = 0$ for Hopper-M-E and $\log \tau = 0.5$ for the other 8 datasets. We then obtain the results for the other 7 seeds with these HP settings and report the results on the 10 seeds in total.

$\mathcal{I}_{\text{MG}}$ (Table 4) & $\mathcal{I}_{\text{SG}}$ (Table 4). Different from the results in Table 1, we use the same $\log \tau = 0.5$ for all datasets including Hopper-M-E to obtain the performance of $\mathcal{I}_{\text{MG}}$ in Table 4. In this way, we aim to understand the effectiveness of each component of our methods better. To fairly compare $\mathcal{I}_{\text{MG}}$ and $\mathcal{I}_{\text{SG}}$, we tune the τ for $\mathcal{I}_{\text{SG}}$ in a similar way by performing a grid search on $\log \tau \in \{0.5, 1.0, 1.5, 2.0\}$ with 3 seeds and finally set $\log \tau = 0.5$ for all datasets. We then obtain the results for the other 7 seeds and report the results with 10 seeds in total.

MG-EBCQ & SG-EBCQ. We tune the number of candidate actions N_{bcq} from the same range $\{2, 5, 10, 20, 50, 100\}$ as is in (Brandfonbrener et al., 2021). For each N_{bcq}, we obtain its average performance for all tasks across 10 seeds and select the best performing N_{bcq} for each method. We separately tune the N_{bcq} for MG-EBCQ and SG-EBCQ. As a result, we set $N_{\text{bcq}} = 5$ for MG-EBCQ and $N_{\text{bcq}} = 10$ for SG-EBCQ. **Moreover, we highlight that MG-EBCQ (SG-EBCQ) uses the same behavior policy and value function as is in $\mathcal{I}_{\text{MG}}$ ($\mathcal{I}_{\text{SG}}$).** We include the full hyper-parameter search results in Table 9 and Table 10.

Table 11: HP search for MG-Rev. KL Reg. We report the mean and std of 10 seeds, and each seed evaluates for 100 episodes.

Dataset	$\alpha = 0.03$	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 1.0$	$\alpha = 3.0$	$\alpha = 10.0$
Cheetah-M-v2	58.3 ± 1.1	58.1 ± 1.2	55.6 ± 0.5	50.6 ± 0.3	47.0 ± 0.2	44.5 ± 0.2
Hopper-M-v2	14.4 ± 13.6	41.0 ± 31.0	89.4 ± 22.2	99.7 ± 1.1	76.3 ± 6.9	58.5 ± 4.0
Walker2d-M-v2	5.8 ± 4.8	18.4 ± 21.9	34.2 ± 27.3	82.2 ± 7.8	82.8 ± 1.8	76.9 ± 2.0
Cheetah-M-R-v2	46.7 ± 1.8	47.5 ± 1.6	48.1 ± 0.7	46.4 ± 0.6	44.4 ± 0.5	43.1 ± 0.4
Hopper-M-R-v2	70.9 ± 33.8	86.6 ± 26.3	103.1 ± 0.8	101.4 ± 1.1	99.4 ± 2.1	77.6 ± 17.2
Walker2d-M-R-v2	73.7 ± 28.8	65.4 ± 33.8	64.0 ± 39.9	65.4 ± 35.8	69.7 ± 30.9	57.7 ± 22.8
Cheetah-M-E-v2	0.4 ± 2.2	1.2 ± 1.9	4.0 ± 1.9	25.0 ± 6.3	65.0 ± 10.1	86.2 ± 7.1
Hopper-M-E-v2	2.6 ± 1.7	16.2 ± 7.9	22.5 ± 10.7	57.4 ± 23.6	79.4 ± 32.6	86.8 ± 15.7
Walker2d-M-E-v2	10.4 ± 15.3	25.5 ± 38.1	93.5 ± 34.5	109.8 ± 0.6	107.1 ± 4.0	97.4 ± 7.0
Total	283.2 ± 103.0	359.9 ± 163.5	514.3 ± 138.5	637.8 ± 77.2	671.2 ± 89.1	628.6 ± 76.4

Table 12: HP search for SG-Rev. KL Reg. We report the mean and std of 10 seeds, and each seed evaluates for 100 episodes.

Dataset	$\alpha = 0.03$	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 1.0$	$\alpha = 3.0$	$\alpha = 10.0$
Cheetah-M-v2	58.6 ± 1.3	57.9 ± 0.8	55.2 ± 0.5	50.7 ± 0.5	47.1 ± 0.2	44.5 ± 0.3
Hopper-M-v2	18.7 ± 15.6	40.2 ± 24.7	83.2 ± 19.6	98.8 ± 2.0	70.3 ± 7.0	57.2 ± 4.6
Walker2d-M-v2	5.6 ± 3.5	26.2 ± 27.1	37.0 ± 27.6	83.3 ± 7.5	82.4 ± 1.0	77.1 ± 1.2
Cheetah-M-R-v2	46.1 ± 3.6	47.8 ± 1.3	47.8 ± 0.8	46.0 ± 0.5	44.3 ± 0.4	42.5 ± 0.6
Hopper-M-R-v2	77.4 ± 19.1	60.8 ± 27.7	92.0 ± 21.9	100.7 ± 1.0	99.7 ± 1.0	70.3 ± 19.2
Walker2d-M-R-v2	59.5 ± 31.3	72.7 ± 38.8	75.7 ± 30.4	75.1 ± 25.3	63.6 ± 28.5	59.7 ± 21.5
Cheetah-M-E-v2	1.1 ± 3.2	3.4 ± 3.4	7.1 ± 4.3	38.9 ± 18.4	78.9 ± 9.8	89.1 ± 4.0
Hopper-M-E-v2	5.5 ± 4.0	20.1 ± 8.6	24.8 ± 7.9	43.8 ± 23.6	76.6 ± 18.3	67.7 ± 30.6
Walker2d-M-E-v2	1.7 ± 3.7	13.4 ± 33.5	83.2 ± 37.5	109.9 ± 0.7	106.7 ± 4.1	96.8 ± 7.6
Total	274.0 ± 85.3	342.6 ± 165.9	505.9 ± 150.5	647.2 ± 79.5	669.7 ± 70.3	604.9 ± 89.5

MG-Rev. KL Reg & SG-Rev. KL Reg. We tune the regularization strength α from the same range $\{0.03, 0.1, 0.3, 1.0, 3.0, 10.0\}$ as is in (Brandfonbrener et al., 2021). For each α, we obtain its average performance for all tasks across 10 seeds and select the best performing α for each method. We separately tune the α for MG-Rev. KL Reg & SG-Rev. KL Reg, although $\alpha = 3.0$ achieves the best overall performance in both methods. **Moreover, we highlight that MG-Rev. KL Reg (SG-Rev. KL Reg) uses the same behavior policy and value function as is in $\mathcal{I}_{\mathbf{MG}}$ ($\mathcal{I}_{\mathbf{SG}}$).** We include the full hyper-parameter search results in Table 11 and Table 12.

F.2 HP AND TRAINING DETAILS FOR METHODS IN TABLE 3

	Hyperparameter	Value
Shared HP	Normalize states	False
IQL HP	Optimizer	Adam (Kingma & Ba, 2014)
	Number of gradient steps	1M
	Mini-batch size	256
	Policy learning rate	3e-4
	Policy hidden dim	256
	Policy hidden layers	2
	Policy activation function	ReLU
	Critic architecture	MLP
	Critic learning rate	3e-4
	Critic hidden dim	256
	Critic hidden layers	2
	Critic activation function	ReLU
	Target update rate	5e-3
	Target update period	1
	quantile	0.9
	temperature	10.0
$\mathcal{I}_{\text{SG}}(\pi_{\text{IQL}}, Q_{\text{IQL}})$	$\log \tau$	selected from $\{0.1, 0.2, 2.0\}$

Table 13: Hyperparameters for methods in Table 3

Table 13 includes the HP for experiments in Sec. 5.2. The of IQL. We use the same HP for the IQL training as is reported in the IQL paper. We obtain the IQL policy π_{IQL} and Q_{IQL} by training for 1M gradient steps using the PyTorch Implementation from RLkit (Berkeley), a widely used RL library. We emphasize that we follow the authors' exact training and evaluation protocol. We include the training curves for all tasks from the AntMaze domain in Appendix G.6.

Note that IQL (Kostrikov et al., 2021) reported inconsistent offline experiment results on AntMaze in its paper's Table 1, Table 2, Table 5, and Table 6 [2]. We suspect that these results are obtained from different sets of random seeds. In Appendix G.6, we present all these results in Table 17.

To obtain the performance for $\mathcal{I}_{\text{SG}}(\pi_{\text{IQL}}, Q_{\text{IQL}})$, we follow the practice of (Brandfonbrener et al., 2021; Fu et al., 2020) and perform a grid search on $\log \tau \in \{0.1, 0.2, 2.0\}$ using 3 seeds for each dataset. We then evaluate the best choice for each dataset by obtaining corresponding results on the other 7 seeds. We finally report the results with 10 seeds in total.

[2] Link to the IQL paper. IQL's Table 5 & 6 are presented in the supplementary material.

G ADDITIONAL EXPERIMENTS

G.1 COMPLETE EXPERIMENT RESULTS FOR MG-MS

Table 14 provides the results of MG-MS on the 9 tasks from the MuJoCo Gym domain in compensation for the results in Sec. 5.3.

Table 14: Results of MG-MS on the MuJoCo Gym domain. We report the mean and standard deviation across 10 seeds, and each seed evaluates for 100 episodes.

Dataset	MG-MS (16)
Cheetah-M-v2	43.6 ± 0.2
Hopper-M-v2	55.3 ± 6.3
Walker2d-M-v2	73.6 ± 2.2
Cheetah-M-R-v2	42.4 ± 0.4
Hopper-M-R-v2	61.5 ± 15.1
Walker2d-M-R-v2	65.0 ± 10.4
Cheetah-M-E-v2	91.3 ± 2.1
Hopper-M-E-v2	104.2 ± 5.1
Walker2d-M-E-v2	104.1 ± 6.7
Total	641.1 ± 48.5

G.2 COMPLETE EXPERIMENT RESULTS ON THE EFFECT OF THE HP τ

Fig. 6 presents additional results in compensation for the results in Sec. 5.3. We note that Hopper-Medium-Expert-v2 requires a much smaller $\log \tau$ than the other tasks to perform well.

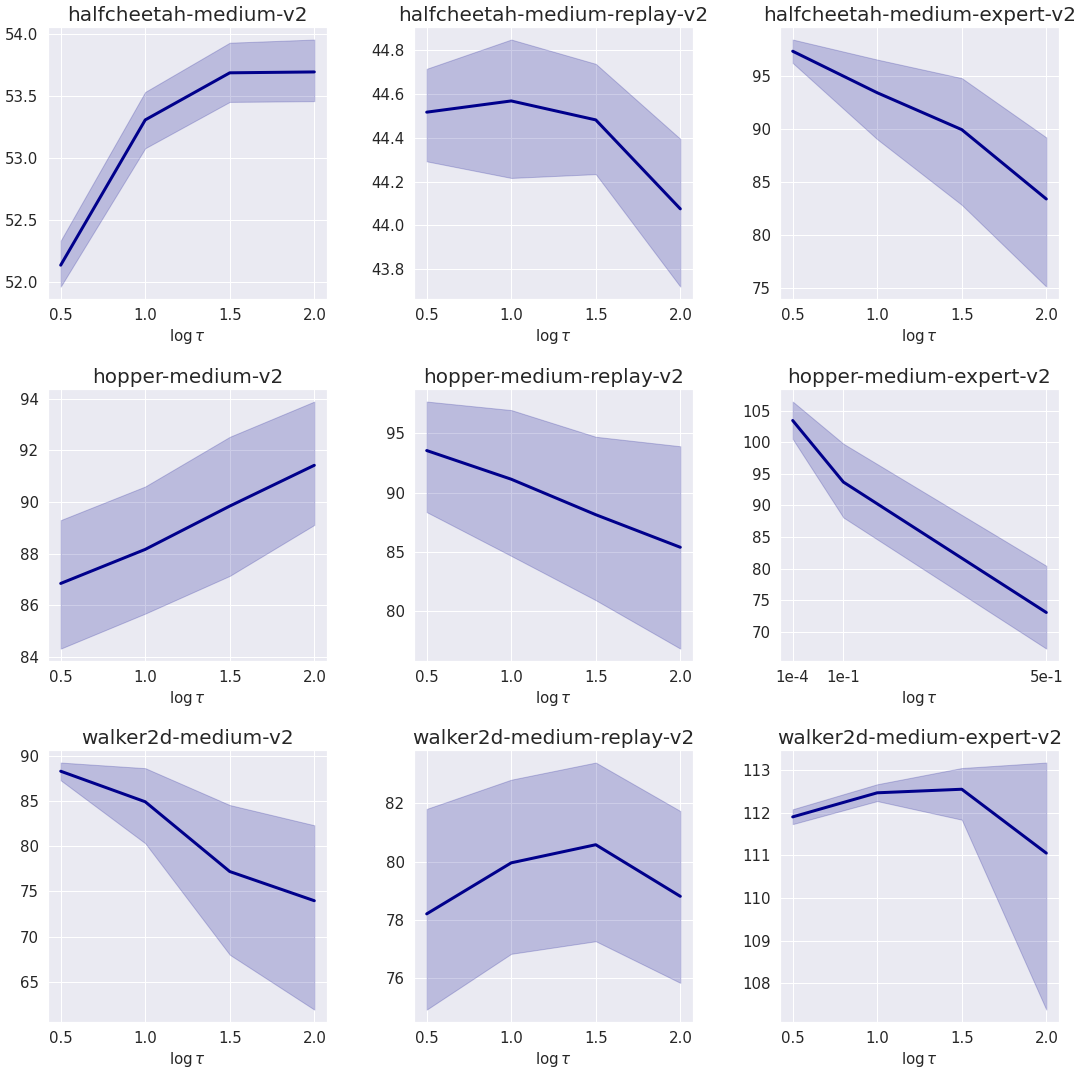


Figure 6: Performance of $\mathcal{I}_{\text{MG}}$ with varying $\log \tau$. The other HP can be found in Table 8. Each variant averages returns over 10 seeds, and each seed contains 100 evaluation episodes. The shaded area denotes bootstrapped 95% CI.

G.3 ABLATION STUDY ON THE NUMBER GAUSSIAN COMPONENTS

In this section, we explore whether increasing the number of Gaussian components will result in a performance boost. We use the same settings as in Table 1 except modeling $\hat{\pi}_\beta$ with 8 Gaussian instead of 4. We hypothesize the performance gain should most likely happen on the three Medium-Replay datasets, as these datasets are collected by diverse policies. However, Table 15 shows that simply increasing the number of Gaussian components from 4 to 8 hardly results in a performance boost, as increasing the number of Gaussian components will induce extra optimization difficulties during behavior cloning (Jin et al., 2016).

Table 15: Comparison between setting the number of Gaussian components to 4 and 8 for our $\mathcal{I}_{\text{MG}}$ on the three Medium-Replay datasets. We report the mean and standard deviation across 10 seeds, and each seed evaluates for 100 episodes.

Dataset	4 components (Table 1)	8 components
Cheetah-M-R-v2	44.5 ± 0.4	44.3 ± 0.3
Hopper-M-R-v2	93.6 ± 7.9	90.6 ± 11.6
Walker2d-M-R-v2	78.2 ± 5.6	79.4 ± 4.5

G.4 MODELING THE VALUE NETWORK WITH CONVENTIONAL MLP

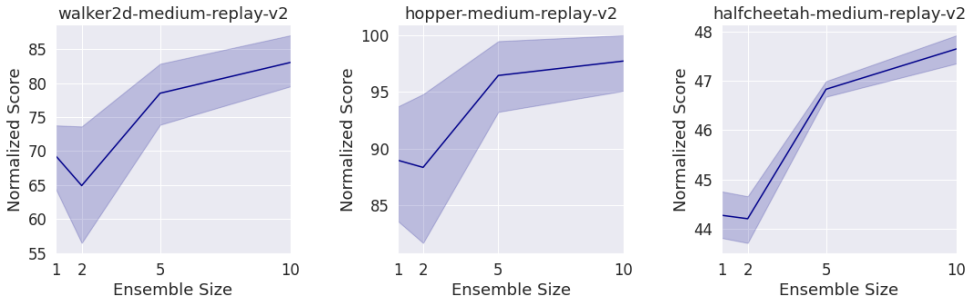


Figure 7: Performance of $\mathcal{I}_{\text{MG}}$ with varying ensemble sizes. Each variant averages returns over 8 seeds, and each seed contains 100 evaluation episode. Each Q-value network is modeled by a 3-layer MLP. The shaded area denotes bootstrapped 95% CI.

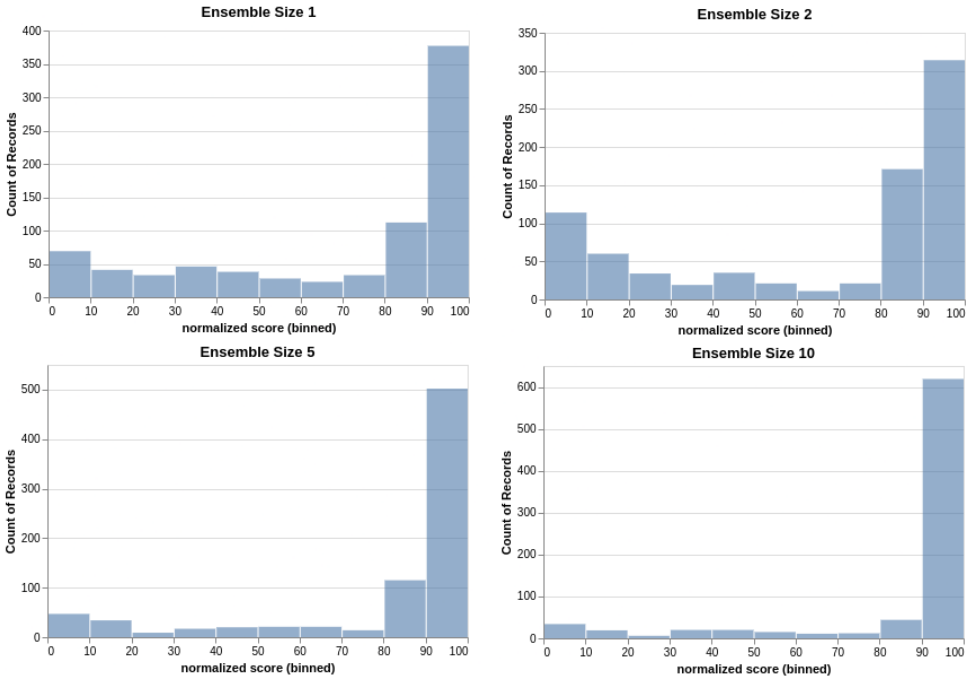


Figure 8: Performance of $\mathcal{I}_{\text{MG}}$ with varying ensemble sizes on Walker2d-Medium-Replay-v2. Each variant aggregates returns over 8 seeds, and each seed evaluates for 100 episodes. Each Q-value network is modeled by a 3-layer MLP. With lower ensemble size, the performance exhibits large variance across different episodes.

Our experiments in Sec. 5.1 rely on learning a Q value function with the IQN (Dabney et al., 2018a) architecture. In this section, we examine the effectiveness of our CFPI operator $\mathcal{I}_{\text{MG}}$ when working with an ensemble of conventional MLP Q-value networks with varying ensemble sizes M.

Each Q-value network $\hat{Q}^{\text{MLP}}_{\theta_k}$ uses ReLU activation and is parameterized with θ_k, including 3 hidden layers of width 256. We train each $\hat{Q}^{\text{MLP}}_{\theta_k}$ by minimizing the bellman error below

$$L(\theta_k) = \mathbb{E}_{(s,a,r,s',a')\sim\mathcal{D}}\left[r + \gamma\hat{Q}^{\text{MLP}}(s',a';\bar{\theta}_k) - \hat{Q}^{\text{MLP}}(s,a;\theta_k)\right], \tag{63}$$

where $\bar{\theta}_k$ is the parameter of a target network given by the Polyak averaging of θ. We set $\hat{Q}^{\text{MLP}}(s,a;\theta_k) = \hat{Q}^{\text{MLP}}_{\theta_k}(s,a)$. We further note that Equation 62 can be reformulated as

$$\begin{aligned}\hat{Q}_0(s,a) = \min_{k=1,2} \hat{Q}^k_0(s,a) &= \frac{1}{2}|\hat{Q}^1_0(s,a) + \hat{Q}^2_0(s,a)| - \frac{1}{2}|\hat{Q}^1_0(s,a) - \hat{Q}^2_0(s,a)| \\ &= \hat{\mu}_Q(s,a) - \hat{\sigma}_Q(s,a),\end{aligned} \tag{64}$$

where $\hat{\mu}_Q$ and $\hat{\sigma}_Q$ calculate the mean and standard deviation of Q value (Ciosek et al., 2019). In the case with an ensemble of Q, we obtain $\hat{Q}_0(s,a)$ by generalizing (64) as below

$$\begin{aligned}\hat{Q}_0(s,a) &= \hat{\mu}^{\text{MLP}}_Q - \sqrt{\frac{1}{M}\sum_{k=1}^{M}\left(\hat{Q}^{\text{MLP}}(s,a;\theta_k) - \hat{\mu}^{\text{MLP}}_Q\right)^2}, \\ \text{where} \quad \hat{\mu}^{\text{MLP}}_Q &= \frac{1}{M}\sum_{k=1}^{M}\hat{Q}^{\text{MLP}}(s,a;\theta_k).\end{aligned} \tag{65}$$

Other than the Q-value network, we applied the same setting as $\mathcal{I}_{\text{MG}}$ in Table 4. Fig. 7 presents the results with different ensemble sizes, showing that the performance generally increases with the ensemble size. Such a phenomenon illustrates a limitation of our CFPI operator $\mathcal{I}_{\text{MG}}$, as it heavily relies on accurate gradient information $\nabla_a[\hat{Q}_0(s,a)]_{a=a_\beta}$.

A large ensemble of Q is more likely to provide accurate gradient information, thus leading to better performance. In contrast, a small ensemble size provides noisy gradient information, resulting in high variance across different rollout, as is shown in Fig. 8.

G.5 How to decide the number of gradient steps for SARSA training?

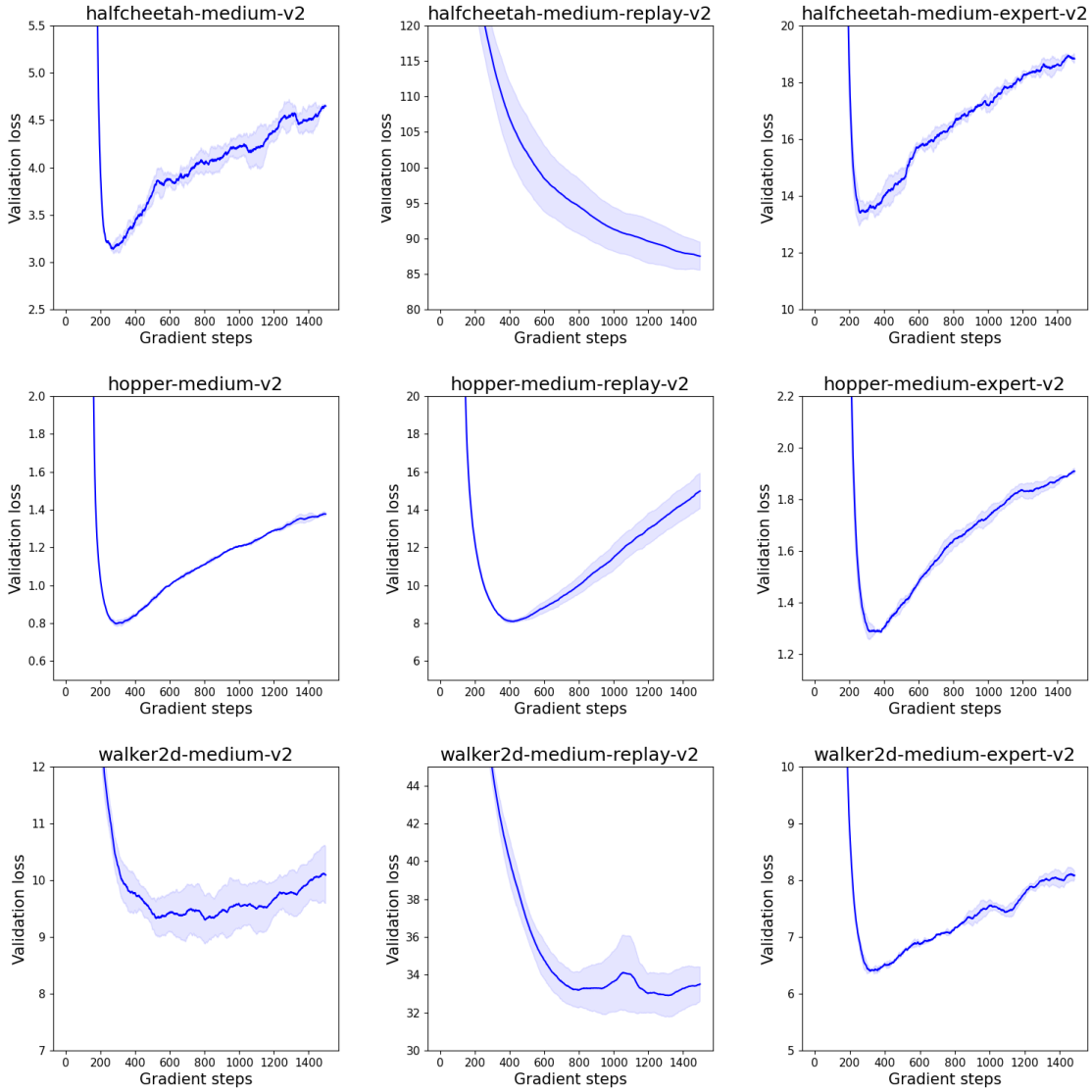


Figure 9: $\mathcal{L}_{\text{val}}$ on each dataset from the Gym-MuJoCo domain. We can observe that the model overfits to the training set when training for too may gradient steps. Each figure averages the validation loss over 2 folds with the same training seed. The shaded area denotes one standard deviation.

Deciding the number of gradient steps is a non-trivial problem in offline RL. While we use a fixed number of gradient steps for behavior cloning, we design a rigorous procedure to decide the gradient steps for SARSA training, inspired by the success of *k-fold validation*.

In our preliminary experiments, we first train a $\hat{Q}^{\beta}_{\text{all}}$ using all data from each dataset for 2M gradient steps. We model the $\hat{Q}^{\beta}_{\text{all}}(s,a)$ as a 3-layer MLP and train following Appendix G.4. By training in this way, we treat $\hat{Q}^{\beta}_{\text{all}}(s,a)$ as the ground truth $Q^{\beta}(s,a)$ for all (s,a) sampled the dataset $\mathcal{D}$. Next, we randomly split the dataset with the ratio 95/5 to create the trainining set $\mathcal{D}_{\text{train}}$ validation set $\mathcal{D}_{\text{val}}$. We then train a new $\hat{Q}^{\beta}$ the SARSA training on $\mathcal{D}_{\text{train}}$. Therefore, we can define the validation loss as

$$\mathcal{L}_{\text{val}} = \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{val}}} ||\hat{Q}^{\beta}_{\text{all}}(s,a) - \hat{Q}^{\beta}(s,a)||^2 \tag{66}$$

Fig. 9 presents the $\mathcal{L}_{\text{val}}$ on each dataset from the Gym-MuJoCo domain. We can clearly observe that $\hat{Q}^{\beta}$ generally overfits the $\mathcal{D}_{\text{train}}$ when training for too many gradient steps. We evaluate over two folds

Table 16: Gradient steps for the SARSA training

Dataset	Gradient steps (K)
HalfCheetah-Medium-v2	200
Hopper-Medium-v2	400
Walker2d-Medium-v2	700
HalfCheetah-Medium-Replay-v2	1500
Hopper-Medium-Replay-v2	300
Walker2d-Medium-Replay-v2	1100
HalfCheetah-Medium-Expert-v2	400
Hopper-Medium-Expert-v2	400
Walker2d-Medium-Expert-v2	400

with one seed. Therefore, we can decide the gradient steps of each dataset for the SARSA training according to the results in Fig. 9 as listed in Table 16.

G.6 OUR REPRODUCED IQL TRAINING CURVES

We use the PyTorch (Paszke et al., 2019) Implementation of IQL from RLkit (Berkeley) to obtain its policy π_{IQL} and value function Q_{IQL}. We do not use the official implementation[3] open-sourced by the authors because our CFPI operators are also based on PyTorch. Fig. 10 presents our reproduced training curves of IQL on the 6 datasets from the AntMaze domain.

We note that the IQL paper[4] does not report consistent results in their paper for the offline experiment performance on the AntMaze, as is shown in Table 17. We suspect that these results are obtained from different sets of random seeds. Therefore, we can conclude that our reproduced results match the results reported in the IQL paper. We believe our reproduction results of IQL are reasonable, even if we do not use the official implementation open-sourced by the authors.

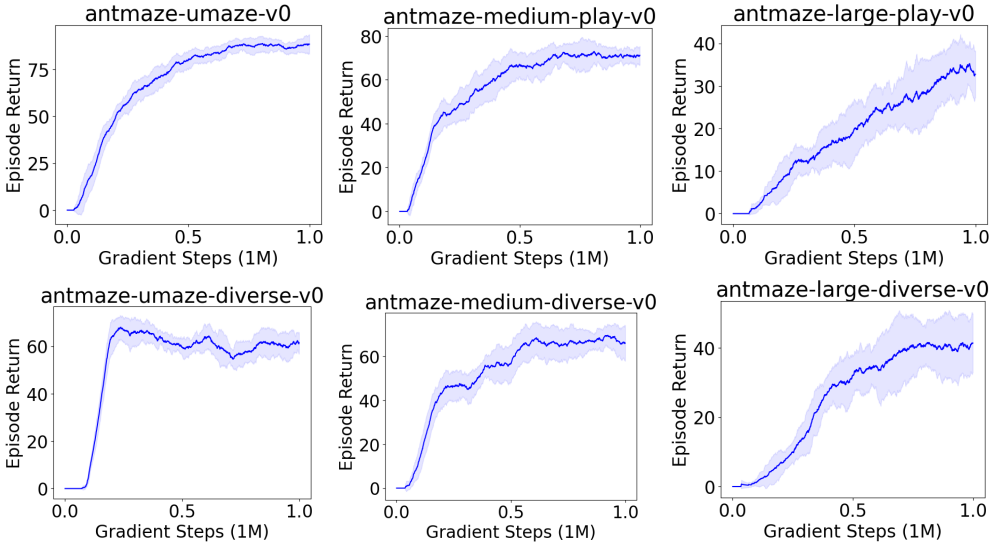


Figure 10: IQL offline training results on AntMaze. Shaded area denotes one standard deviation.

Table 17: Offline experiment results on AntMaze reported in different tables from the IQL paper

Dataset	Table 1 & 6	Table 2	Table 5
antmaze-u-v0	87.5 ± 2.6	88.0	86.7
antmaze-u-d-v0	62.2 ± 13.8	67.0	75.0
antmaze-m-p-v0	71.2 ± 7.3	69.0	72.0
antmaze-m-d-v0	70.0 ± 10.9	71.8	68.3
antmaze-l-p-v0	39.6 ± 5.8	36.8	25.5
antmaze-l-d-v0	47.5 ± 9.5	42.2	42.6
Total	378.0 ± 49.9	374.8	370.1

[3] https://github.com/ikostrikov/implicit_q_learning

[4] Link to the IQL paper. IQL's Table 5 & 6 are presented in the supplementary material.

G.7 IMPROVE THE POLICY LEARNED BY CQL

In this section, we show that our CFPI operators can also improve the policy learned by CQL (Kumar et al., 2020b) on the MuJoCo Gym Domain. We first obtain the CQL policy π_{CQL} and Q_{CQL} by training for 1M gradient steps using the official CQL implementation[5]. We obtain an improved policy $\mathcal{I}_{\text{SG}}(\pi_{\text{CQL}}, Q_{\text{CQL}}; \tau)$ that slightly outperforms π_{CQL} overall, as shown in Table 18. For all 6 tasks, we set $\log \tau = 0.1$.

Table 18: Improving the policy learned by IQL with our CFPI operator $\mathcal{I}_{\text{SG}}$

Dataset	π_{CQL} (1M)	$\mathcal{I}_{\text{SG}}(\pi_{\text{CQL}}, Q_{\text{CQL}})$
HalfCheetah-Medium-v2	45.5 ± 0.3	$\mathbf{47.1 \pm 1.5}$
Hopper-Medium-v2	65.4 ± 3.5	$\mathbf{70.1 \pm 4.9}$
Walker2d-Medium-v2	81.4 ± 0.6	$\mathbf{81.6 \pm 1.1}$
HalfCheetah-Medium-Replay-v2	44.6 ± 0.5	$\mathbf{45.9 \pm 1.7}$
Hopper-Medium-Replay-v2	95.2 ± 2.0	94.6 ± 1.6
Walker2d-Medium-Replay-v2	80.1 ± 2.6	78.8 ± 3.2
Total	412.2 ± 9.4	$\mathbf{418.2 \pm 13.9}$

[5]https://github.com/aviralkumar2907/CQL

H ADDITIONAL RELATED WORK

There has been a history of leveraging the Taylor expansion to construct efficient RL algorithms. Kakade & Langford (2002) proposed the conservative policy iteration that optimizes a mixture of policies towards its policy objective's lower bound, which is constructed by performing first-order Taylor expansion on the mixture coefficient. Later, SOTA deep RL algorithms TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017) extend the results to work with trust region policy constraints and learn a stochastic policy parameterized by a neural network. More recently, Tang et al. (2020) developed a second-order Taylor expansion approach under similar online RL settings.

At a high level, both our works and previous methods propose to create a surrogate of the original policy objective by leveraging the Taylor expansion approach. However, our motivation to use Taylor expansion is fundamentally different from the previous works (Kakade & Langford, 2002; Schulman et al., 2015; 2017; Tang et al., 2020), which leverage the Taylor expansion to construct a lower bound of the policy objective so that optimizing towards the lower bound translates into guaranteed policy improvement. However, these methods do not result in a closed-form solution to the policy and still require iterative policy updates.

On the other hand, our method leverages the Taylor expansion to construct a linear approximation of the policy objective, enabling the derivation of a closed-form solution to the policy improvement step and thus avoiding performing policy improvement via SGD. We highlight that our closed-form policy update cannot be possible without directly optimizing the parameter of the policy distribution. In particular, the parameter should belong to the action space. **We note that this is a significant conceptual difference between our method and previous works.**

Specifically, PDL (Kakade & Langford, 2002) parameterizes the mixture coefficient of a mixture policy as θ. TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017) set θ as the parameter of a neural network that outputs the parameters of a Gaussian distribution. In contrast, our methods learn deterministic policy $\pi(s) = \text{Dirac}(\theta(s))$ and directly optimize the parameter $\theta(s)$. We aim to learn a greedy π by solving $\theta(s) = \arg\max_a Q(s, a)$. However, obtaining a greedy π in continuous control is problematic (Silver et al., 2014). Given the requirement of limited distribution shift in the offline RL, we thus leverage the first-order Taylor expansion to relax the problem into a more tractable form

$$\theta(s) = \arg\max_a \bar{Q}(s, a; a_\beta), \text{ s.t. } -\log \pi_\beta(a|s) \leq \delta, \tag{67}$$

where $\bar{Q}$ is defined in Equation 3. By modeling π_β as a Single Gaussian or Gaussian Mixture, we further transform the problem into a QCLP and thus derive the closed-form solution.

Finally, we note that both the trust region methods TRPO and PPO and our methods constrain the divergence between the learned policy and behavior policy. However, the behavior policy always remains unchanged in our offline RL settings. As TRPO and PPO are designed for the online RL tasks, the updated policy will be used to collect new data and becomes the new behavior policy in future training iteration.