Offline Reinforcement Learning with Closed-Form Policy Improvement Operators

Jiachen Li[*]
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
`jiachen_li@cs.ucsb.edu`

Edwin Zhang[*]
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
`ete@cs.ucsb.edu`

Ming Yin
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
`ming_yin@cs.ucsb.edu`

Qinxun Bai
Horizon Robotics Inc.
Cupertino, CA, 95014 USA
`qinxun.bai@gmail.com`

Yu-Xiang Wang
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
`yuxiangw@cs.ucsb.edu`

William Yang Wang
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
`william@cs.ucsb.edu`

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 an offline RL algorithm with our novel policy improvement operator and empirically demonstrate its effectiveness over state-of-the-art algorithms on the standard D4RL benchmark.

[*]Equal Contribution

Preprint. Under review.

1 Introduction

Deploying Reinforcement Learning (RL) [1] in the real world is hindered by its massive demand for online data. In domains such as robotics [2] and autonomous driving [3], rolling out a premature policy is prohibitively costly and unsafe. To address this issue, offline RL (a.k.a batch RL) [4, 5] 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 original data-generating behavior policies can suffer from significant extrapolation errors, leading to training divergence [6, 7].

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 [8], behavior constrained policy optimization (BCPO) [6, 7, 9, 10, 11, 12, 13] explicitly regularizes the divergence between learned and behavior policies, while conservative methods [14, 15, 16, 17] 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 [1] is notoriously unstable [18, 19, 20, 8, 21] due to the existence of the deadly-triad [1, 22]. In offline settings, learning becomes even more problematic as environment interaction is restricted, thus preventing the learning from receiving corrective feedback [18]. Consequently, training stability poses a major challenge in offline RL. Although some current approaches [11, 12] 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 aim to design a simple yet effective offline RL algorithm that achieves this goal. 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 [23], 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.

Our policy improvement operators can instantiate an offline RL algorithm in a one-step, multi-step, or iterative fashion. In this paper, we focus on a one-step method inspired by the success of the IQL [11] and OneStep RL [12] to avoid off-policy evaluation. Moreover, our policy improvement operators can also be leveraged to improve the performance of 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 effectiveness of modeling 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.
- A one-step instantiation of our algorithm, which outperforms state-of-the-art (SOTA) algorithms on MuJoCo Gym domain of the standard D4RL benchmark [24].

2 Preliminaries

Reinforcement Learning. RL aims to maximize returns in a Markov Decision Process (MDP) [1] $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, T, \rho_0, \gamma)$, 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 network tends to assign spuriously high values to OOD actions due to extrapolation error, which is well documented in previous literature [6, 7]. Behavior Constrained Policy Optimization (BCPO) methods [6, 7, 9, 10, 11, 12] 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 [10, 25], MSE [9] and MMD [7]). 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 [23], $\bar{Q}(s, a; a_\beta)$ 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 for our method.

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*

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

Proof sketch. (4) can be converted into the QCLP below that be solved in closed form [26].

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

A fulll proof is given in the Appendix B.1. □

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 [26], (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, setting a large τ might send μ out of the support of the behavior policy distribution and break the accuracy guarantee of the first-order Taylor approximation. Thankfully, our method enables a line search of $\tau_{1\ldots K}$ for a reasonably wide range with the cost of one additional forward pass of the Q function, which is an unique virtue of the closed-form tractability and leads to an action selection procedure below

$$\mu_{\text{sg}}^\star = \arg\max\nolimits_\mu Q(s,\mu), \quad \text{s.t.} \quad \mu \in \{\mu_{\text{sg}}(\tau_i) | i = 1\ldots K\} \tag{9}$$

3.2 Gaussian Mixture as a more expressive model

The policy improvement operator in its closed form (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. In practice, the historical datasets are usually collected by heterogeneous policies with different levels of expertise, and a Single Gaussian may fail to capture the whole picture of the underlying distribution. This issue motivates us to use a Gaussian Mixture to represent the behavior policy as

$$\pi_\beta = \sum_{i=1}^{N} \lambda_i \mathcal{N}(\mu_i, \Sigma_i), \tag{10}$$

where $\mu_{i=1\ldots N}$, $\Sigma_{i=1\ldots N}$, and $\lambda_{i=1\ldots N}$ are the corresponding mean, covariance matrix, and mixture weights for the N Gaussian components. However, directly plugging the Gaussian Mixture π_β into (5) breaks its tractability, resulting in a non-convex optimization below

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

We are confronted with two major challenges to solve the optimization problem (11). 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 (11) does not admit a convex form, posing non-trivial optimization difficulties. We leverage the lemma below to tackle the non-convexity of the constraint.

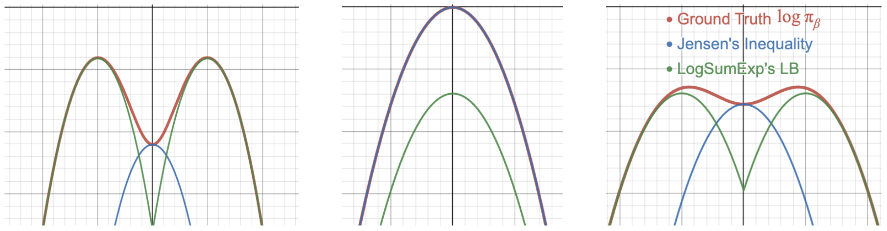


Figure 1: Apply Lemma 3.1 to Gaussian Mixture's log probability $\log \pi_\beta$ at different scenarios. (L) $\log \pi_\beta$ has multiple modes. LogSumExp's LB preserves multimodality. (M) $\log \pi_\beta$ reduces to Single Gaussian. Jensen's inequality becomes equality. (R) $\log \pi_\beta$ is similar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

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 (11) 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* (11)*, we can derive an optimization problem that lower bounds* (11)

$$\begin{aligned} &\max_\mu \quad \mu^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta} \\ &\textit{s.t.} \quad \max_i \left\{ -\frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1}(\mu-\mu_i) - \frac{1}{2}\log\det(2\pi\Sigma_i) + \log\lambda_i \right\} \geq -\delta, \end{aligned} \tag{12}$$

and the closed-form solution to (12) *is given by*

$$\begin{aligned} &\mu_{lse}(\tau) = \operatorname*{arg\,max}_{\bar{\mu}_i(\delta)} \quad \bar{\mu}_i^T \left[\nabla_a Q(s,a)\right]_{a=\mu_i}, \quad \textit{s.t.} \quad \delta = \frac{1}{2}\min_i \left\{\log \lambda_i \det(2\pi\Sigma_i)\right\} + \log\tau \\ &\textit{where} \quad \bar{\mu}_i(\delta) = \mu_i + \sqrt{\frac{2(\delta+\log\lambda_i) - \log\det(2\pi\Sigma_i)}{\left[\nabla_a Q(s,a)\right]_{a=\mu_i}^T \Sigma_i \left[\nabla_a Q(s,a)\right]_{a=\mu_i}}}\, \Sigma_i \left[\nabla_a Q(s,a)\right]_{a=\mu_i} \end{aligned} \tag{13}$$

Proposition 3.3. *By applying the second inequality of Lemma 3.1 to the constraint of* (11)*, we can derive an optimization problem that lower bounds* (11)

$$\begin{aligned} &\max_\mu \quad \mu^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta} \\ &\textit{s.t.} \quad \sum_{i=1}^N \lambda_i \left(-\frac{1}{2}\log\det(2\pi\Sigma_i) - \frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1}(\mu-\mu_i) \right) \geq -\delta \end{aligned} \tag{14}$$

and the closed-form solution to (14) *is given by*

$$\begin{aligned} &\mu_{jensen}(\tau) = \bar{\mu} + \sqrt{\frac{2\log\tau - \sum_{i=1}^N \lambda_i \mu_i^T \Sigma_i^{-1}\mu_i + \bar{\mu}^T \overline{\Sigma}^{-1}\bar{\mu}}{\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}^T \overline{\Sigma} \left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}}}\, \overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}, \\ &\textit{where } \overline{\Sigma} = \left(\sum_{i=1}^N \lambda_i \Sigma_i^{-1}\right)^{-1}, \quad \bar{\mu} = \overline{\Sigma}\left(\sum_{i=1}^N \lambda_i \Sigma_i^{-1}\mu_i\right), \quad \delta = \log\tau + \frac{1}{2}\sum_{i=1}^N \lambda_i \log\det(2\pi\Sigma_i) \end{aligned} \tag{15}$$

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 B.2 and B.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

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 $\tau$
1: Warm start $\hat{Q}_0 = \text{SARSA}(\hat{Q}_{-1}, \mathcal{D})$ with the SARSA-style algorithm [1, 27]
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
```

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 (12) can serve as a reasonable surrogate problem of (11), 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 (15) exactly solves the optimization problem (11). However, in this case, the tightness of LogSumExp's LB largely depends on the weights $\lambda_{i=1\dots N}$. 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 (12) and (14) can serve as reasonable surrogates to the original problem (11).

Fortunately, we can combine the best of both worlds and design a policy improvement operator accounting for all 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}

$$\mu_{\text{mg}}(\tau) = \arg\max_{\mu} Q(s, \mu), \quad \text{s.t.} \quad \mu \in \{\mu_{\text{lse}}(\tau), \mu_{\text{jensen}}(\tau)\} \tag{16}$$

Note that we can still perform a line search for τ with the Gaussian Mixture behavior policy, though the batch size of the forward pass will scale linearly with the number of Gaussian components N. We include the detailed line search procedures in Appendix C.

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 (16). 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 D

While we motivate the design of $\mathcal{I}_{\text{sg}}$ and $\mathcal{I}_{\text{mg}}$ from the behavior constraint, we highlight that both operators are also compatible with the other baseline policy π_b besides π_β. In Sec. 5.2, we will show that our policy improvement operator can further improve a policy derived by another algorithm.

3.4 Theoretical guarantees for closed-form policy improvement

Before presenting our empirical studies, we perform a statistical analysis to demonstrate the rigor of the policy improvement scheme in Algorithm 1. We first explain the key steps of analysis.

At a high level, Algorithm 1 follows the *approximate policy iteration* (API) [28] 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 [1]. Fortunately, it is known that FQI is statistically efficient (*e.g.* [29]) under the mild condition for the function approximation class. Recently, [30] shows the finite sample convergence guarantee for SARSA under the standard mean square error loss. Next, for the performance gap between $J(\hat{\pi}_{t+1}) - J(\hat{\pi}_t)$, we apply the standard performance difference lemma [31, 32]. In particular, we focus on the first update step and we have the following result (the extension to multi-step case is discussed in Appendix B.4).

Theorem 3.1 (One-step Safe Policy Improvement). *Let $\hat{\pi}_1$ be the policy obtained after the CFPI update (Line 2 of Algorithm 1). Then with probability $1-\delta$, ($\mathcal{D}(s,a)$ is number of samples at s,a)*

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

Here $C_{\gamma,\delta}$ is the learning coefficient of SARSA, and C_{CFPI} is the error from the first-order approximation (3), (4) using CFPI. Their concrete definitions are deferred to Appendix B.4.

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 closed-form approximation, and the $C_{\gamma,\delta}/\sqrt{\mathcal{D}(s,a)}$ term is incurred by the SARSA update. When sample size is large enough ($\mathcal{D}(s,a)$ large), this term goes to zero.

4 Related Work

Our methods belong and are motivated by the successful BCPO paradigm in offline RL, which imposes explicit constraints as in (2) to prevent the learned policy from selecting OOD actions. Algorithms from this paradigm may apply different divergence functions, e.g., KL-divergence[10, 25], MMD [7] or the MSE loss [9]. All these methods perform policy improvement via SGD. Instead, we perform CFPI by solving a linear approximation of (2). We also note that another line of research enforces the behavior constraint via parameterization. BCQ [6] learns a generative model as the behavior policy and a Q function to select the action from a set of perturbed behavior actions. However, EMaQ shows that the perturbation model is actually not necessary and can be discarded [13].

The design of our CFPI operators is inspired by the SOTA online RL algorithm OAC [33]. 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.

Recently, one-step [11, 12] algorithms have also achieved great success. Instead of iteratively performing policy improvement and evaluation, these methods only learn a Q function via SARSA, eliminating the opportunity to bootstrap from OOD action value when learning the Q. These methods further apply an improvement operator [10, 34] to extract a policy. We also instantiate a one-step algorithm with our CFPI operator and evaluate it on the standard offline RL benchmark.

5 Experiments

Our experiments aim to demonstrate the effectiveness of our policy improvement operators from two aspects. Firstly, on the standard offline RL benchmark D4RL [24], we show that instantiating an offline RL algorithm with our CFPI operators performs better than SOTA algorithms (Sec. 5.1). Secondly, we show that our policy improvement operators can further improve a policy learned by other algorithms on AntMaze (Sec. 5.2). We further conduct ablation studies to support our design choices (Sec. 5.3). In particular, we empirically demonstrate that modeling the behavior policy as a Gaussian Mixture provides obvious advantage over Single Gaussian when datasets are generated by heterogeneous policies. We also examine the HP sensitivity of our methods and discuss limitations.

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. Motivated by the success of IQL [11] that learns a higher expectile of the return, we employed the IQN [35] architecture to model the value function Q and trained the $\hat{Q}_0$ with SARSA algorithm [1, 27] to estimate the 70% quantile of the behavior return. Appendix E includes detailed training procedures of $\hat{\pi}_b$ and $\hat{Q}_0$ and full HP settings. We thus 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 [11], OneStepRL [12], BCPO method TD3+BC [9], conservative method CQL [14], and

Table 1: Comparison between our one-step policy and SOTA methods on the Gym-MuJoCo domain of D4RL. All methods are evaluated on the 'v2' datasets. Our policy uses two sets of τ for all datasets (detailed in Appendix E). We report the mean and standard deviation of our method's performance across 10 seeds. Each seed contains individual training process and evaluates the policy for 100K environment steps. We use Cheetah for HalfCheetah, M for Medium, M-E for Medium Expert, M-R for Medium-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	53.7 ± 0.3
Hopper-M-v2	53.7	53.9	67.6	61.1	83.3	59.3	58.5	66.2	$\mathbf{88.8 \pm 4.1}$
Walker2d-M-v2	71.9	70.0	74.0	79.8	**85.6**	83.7	72.5	78.3	$\mathbf{85.0 \pm 5.5}$
Cheetah-M-R-v2	34.9	33.0	36.6	41.9	42.4	44.6	**45.5**	44.2	44.8 ± 0.2
Hopper-M-R-v2	12.4	21.2	82.7	91.5	71.0	60.9	95.0	94.7	$\mathbf{98.3 \pm 3.1}$
Walker2d-M-R-v2	22.9	22.8	66.6	82.6	71.6	**81.8**	77.2	73.8	78.9 ± 4.9
Cheetah-M-E-v2	46.6	51.7	86.8	**95.0**	93.5	90.7	91.6	86.7	$\mathbf{94.3 \pm 3.5}$
Hopper-M-E-v2	53.9	69.2	**107.6**	101.9	102.1	98.0	105.4	91.5	100.3 ± 7.7
Walker2d-M-E-v2	92.3	93.2	108.1	110.0	**110.9**	110.1	108.8	109.6	$\mathbf{111.9} \pm 0.7$
Total	429.1	455.6	672.6	710.1	716.0	677.4	698.5	692.4	$\mathbf{756.03 \pm 30.0}$

trajectory optimization methods DT [36], TT [37]. 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 datasets.

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 two sets of τ. Moreover, we do not claim that our $\mathcal{I}_{\text{mg}}$ is always better than all other existing policy improvement operators. We will perform ablation studies in Sec. 5.3 and carefully revisit the crucial design choices necessary to create a successful one-step algorithm with our CFPI operators.

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 [11] on the AntMaze domain. The 6 tasks from AntMaze are more challenging due to their sparse-reward nature and lack of optimal trajectories in the static datasets. We first obtain the IQL policy π_{IQL} and Q_{IQL} by training for 1M gradient steps using the PyTorch Implementation from RLkit [2]. We emphasize that we follow the authors' exact training and evaluation protocol and include all training curves in Appendix E. Interestingly, while the running average of the evaluation results during the course of training matches the reported results in the IQL paper, Table 2 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 2. We tune the HP τ from a small set for each task following the practice of [12, 24] and include more details in Appendix E and F.

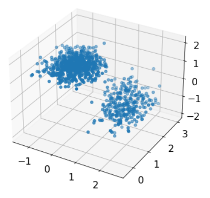
Table 2: Our $\mathcal{I}_{\text{sg}}(\pi_{\text{IQL}}, Q_{\text{IQL}}; \tau)$ improves over the policy π_{IQL} learned by IQL on AntMaze. We report the mean and standard deviation 10 seeds. Each seed for 100K env steps.

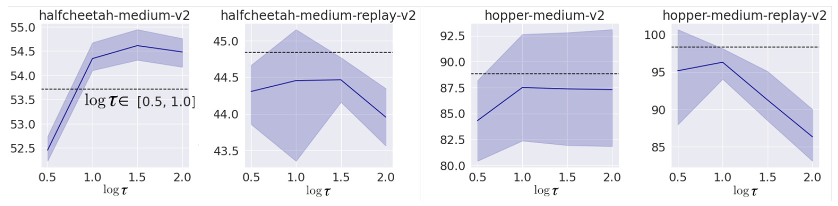
Dataset	π_{IQL} (train)	π_{IQL} (1M)	$\mathcal{I}_{\text{sg}}(\pi_{\text{IQL}}, Q_{\text{IQL}})$
antmaze-u-v0	$\mathbf{87.4 \pm 3.17}$	83.6 ± 3.2	84.3 ± 5.3
antmaze-u-d-v0	$\mathbf{59.0 \pm 5.73}$	55.8 ± 7.9	51.4 ± 8.2
antmaze-m-p-v0	71.1 ± 5.43	64.2 ± 13.2	$\mathbf{77.2 \pm 4.7}$
antmaze-m-d-v0	70.0 ± 6.16	66.8 ± 9.4	$\mathbf{80.6 \pm 5.9}$
antmaze-l-p-v0	34.4 ± 6.04	35.6 ± 7.0	$\mathbf{36.4 \pm 5.6}$
antmaze-l-d-v0	$\mathbf{39.8 \pm 9.09}$	38.8 ± 7.1	39.6 ± 6.6
Total	361.7 ± 35.6	344.7 ± 47.8	$\mathbf{368.9 \pm 36.3}$

[2]`https://github.com/rail-berkeley/rlkit/tree/master/examples/iql`

Table 3: Ablation studies of our Method on the Gym-MuJoCo domain. Again we report the mean and standard deviation 10 seeds and each seed evaluates for 100K environment steps.

Dataset	SG-BC	MG-BC	MG-MS	SG-EBCQ	MG-EBCQ	$\mathcal{I}_{\text{SG}}$	$\mathcal{I}_{\text{MG}}$
Cheetah-M-v2	40.6	40.6	43.5 ± 0.1	53.5 ± 0.3	53.8 ± 0.2	52.8 ± 0.3	$\mathbf{53.7 \pm 0.3}$
Hopper-M-v2	53.7	53.9	58.0 ± 2.0	85.2 ± 5.3	$\mathbf{90.5} \pm 1.8$	81.7 ± 4.8	88.8 ± 4.1
Walker2d-M-v2	71.9	70.0	72.1 ± 3.3	85.6 ± 2.2	81.6 ± 4.0	$\mathbf{89.3} \pm 1.2$	85.0 ± 5.5
Cheetah-M-R-v2	34.9	33.0	42.4 ± 0.4	43.3 ± 0.5	43.8 ± 0.3	42.7 ± 0.6	$\mathbf{44.8 \pm 0.2}$
Hopper-M-R-v2	12.4	21.2	62.4 ± 17.4	94.2 ± 5.7	90.0 ± 6.3	93.1 ± 6.8	$\mathbf{98.3 \pm 3.1}$
Walker2d-M-R-v2	22.9	22.8	64.1 ± 11.1	70.7 ± 6.2	77.4 ± 3.1	73.1 ± 6.4	$\mathbf{78.9 \pm 4.9}$
Cheetah-M-E-v2	46.6	51.7	90.6 ± 2.2	80.9 ± 6.7	81.0 ± 3.0	88.4 ± 7.7	$\mathbf{94.3 \pm 3.5}$
Hopper-M-E-v2	53.9	69.2	$\mathbf{103.1} \pm 5.3$	51.3 ± 5.3	47.3 ± 0.6	54.9 ± 9.1	61.4 ± 9.0
Walker2d-M-E-v2	92.3	93.2	103.8 ± 6.9	111.1 ± 1.2	111.6 ± 0.7	111.5 ± 1.4	$\mathbf{111.9 \pm 0.7}$
Total	429.1	455.6	640.0 ± 48.7	675.7 ± 33.3	677.0 ± 20.9	687.6 ± 38.3	$\mathbf{717.1 \pm 31.3}$


(a)



(b)

Figure 2: (a) Our learned Gaussian Mixture exhibits obvious multi-modality at a specific state on Hopper-M-E-v2 (b) Performance with different $\log \tau$. The dotted line denotes the results when $\log \tau$ is sampled uniformly from $[0.5, 1.0]$. Shaded area represents bootstrapped 95% CI.

5.3 Ablation studies

We first demonstrate the effectiveness of modeling the behavior policy as a Gaussian Mixture. We further ablate the optimization conducted by our methods to evaluate its contributions and examine the HP sensitivity. In Appendix F, we include additional ablations and discuss the limitation of our methods by ablating the Q function network.

Effectiveness of Gaussian Mixture. Fig. 2a shows that our learned Gaussian Mixture behavior policy $\hat{\pi}_\beta$ can indeed capture the multimodality of the underlying action distribution of the Hopper-M-E dataset. 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

$$\mu_{\text{mode}} = \arg\max_{\hat{\mu}_i} \quad \hat{Q}_0(s, \hat{\mu}_i), \quad \text{s.t.} \quad \{\hat{\mu}_i | \hat{\lambda}_i > \xi\}, \quad \text{where} \quad \sum_{i=1}^{N} \hat{\lambda}_i \mathcal{N}(\hat{\mu}_i, \hat{\Sigma}_i) = \hat{\pi}_\beta, \tag{17}$$

ξ is a threshold to filter out trivial components. As shown in Table 3, MG-MS can indeed achieve expert performance on the Hopper- and Walker2d-M-E datasets, along with performance on par with SOTA algorithms in Cheetah-M-E.

Moreover, Table 3 show that with the same $\hat{Q}_0$, $\mathcal{I}_{\text{MG}}$ with Gaussian Mixture behavior $\hat{\pi}_\beta$ outperforms $\mathcal{I}_{\text{SG}}$ with Single Gaussian behavior $\hat{\pi}_\beta$ on all M-R and M-E datasets. The $\hat{Q}_0$ is modeled and learned in the same way as in Sec. 5.1. Here, we slightly abuse the notation and learn both $\hat{\pi}_\beta$ via behavior cloning. We highlight that we tune the HP τ separately for $\mathcal{I}_{\text{MG}}$ and $\mathcal{I}_{\text{SG}}$ and each method use the same set of HP for all 9 datasets. As a result, the Hopper-M-E performance of $\mathcal{I}_{\text{MG}}$ reported in Table 3 is different from the one reported in Table 1.

Effectiveness of solving the approximated BCPO. We show that our policy improvement operator returns a better policy than its Easy BCQ (EBCQ) [12] counterpart. EBCQ returns a policy by selecting an action that maximizes a learned $\hat{Q}$ from a set of N_{bcq} randomly sampled actions from the behavior policy distribution $\hat{\pi}_\beta$. We present the EBCQ results with $\hat{\pi}_\beta$ modeled by Single Gaussian (SG-EBCQ) and Gaussian Mixture (MG-EBCQ), respectively. We tune the N_{bcq} separately for each method and fix its value for all datasets. Results in Table 3 reveal that our $\mathcal{I}_{\text{MG}}$ and $\mathcal{I}_{\text{SG}}$ outperform their EBCQ counterparts, demonstrating the effectiveness of solving the approximated BCPO.

Effect of the HP τ. Fig. 2b shows the effect of τ on the performance of $\mathcal{I}_{\text{MG}}(\hat{\pi}_\beta, \hat{Q}_0; \tau)$. While τ has different impact on different tasks, conducting a simple line search on the τ yields robust performance and improves over a single τ on some tasks. We provide complete results and further analysis in the Appendix F.

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 shows that it can outperform SOTA algorithms on the Gym-MuJoCo domain of the D4RL benchmark.

Our policy improvement operators provide a way to perform policy improvement in closed form, avoiding the learning instability incurred by policy learning 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.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In the Supplementary.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] We include full proofs in the Appendix (supplemental material).
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See supplemental material.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See supplemental material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] It is Apache 2.0.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include our codes.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We use simulated data.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We use simulated data.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

Outline of the Appendix

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

- Appendix A describes the potential broader impact of our work.
- Appendix B presents the missing proofs for Proposition 3.1, 3.2, 3.3 and Theorem 3.1 in the main paper.
- Appendix C justify one HP setting for Equation 16, and illustrate the corresponding line search algorithms.
- Appendix D discusses how to instantiate multi-step and iterative algorithms from our algorithm template Algorithm 1.
- Appendix E gives experiment details and hyper-parameter settings
- Appendix F provides additional experiment results.
- Appendix G lists the computing infrastructures and approximiated training time.

A Broader Impact

Our work lies in the broad context of Offline Reinforcement Learning, where we aim to learn a performant policy from a pre-collected dataset. Since Offline RL eases the requirement to perform online interactions during policy learning, it eliminates the safety risk and financial cost caused by rolling out a pre-mature policy in real-life applications, paving a new way for ubiquitous deployment of RL agents in our daily life. However, learning the policy through stochastic gradient descent consumes extensive computation, while collecting the datasets may still require human labor and non-trivial amounts of computational resources. Finally, it is extremely difficult if not impossible to predict the consequences of new technology. Thus the potential impact of our work, as does almost all work in RL, depends largely on the intentions of the end user. We acknowledge that both potentially large positive and negative impact can result from progress in RL.

The closed-form policy improvement (CFPI) operators proposed in our paper are easy to use and enable us to perform policy improvement without learning, which significantly reduces the computation cost due to the policy improvement compared with existing methods. Moreover, our CFPI operators prevent us from the training instability caused by the unstable learning procedures. In summary, we believe our CFPI operators can benefit the community by reducing the computational cost during training and facilitating the stability of an offline RL algorithm.

B Proofs and Theoretical Results

B.1 Proof of Proposition 3.1

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

$$\mu_{sg}(\tau) = \mu_\beta + \frac{\sqrt{2\log\tau}\,\Sigma_\beta\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}}{\sqrt{\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}^T \Sigma_\beta \left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}}}, \quad \textit{where} \;\; \delta = \frac{1}{2}\log\det(2\pi\Sigma_\beta) + \log\tau \tag{18}$$

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

$$\max_{\mu} \quad \mu^T\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}, \quad \text{s.t.} \quad \frac{1}{2}(\mu-\mu_\beta)^T\Sigma_\beta^{-1}(\mu-\mu_\beta) \le \delta - \frac{1}{2}\log\det(2\pi\Sigma_\beta) \tag{19}$$

Following a similar procedure as is in OAC [33], we first derive the Lagrangian below:

$$L = \mu^T\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta} - \eta\left(\frac{1}{2}(\mu-\mu_\beta)^T\Sigma_\beta^{-1}(\mu-\mu_\beta) - \delta + \frac{1}{2}\log\det(2\pi\Sigma_\beta)\right) \tag{20}$$

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

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

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

$$\mu = \mu_\beta + \frac{1}{\eta}\Sigma_\beta\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta} \tag{22}$$

To satisfy the the KKT conditions [26], we have $\eta > 0$ and

$$(\mu-\mu_\beta)^T\Sigma_\beta^{-1}(\mu-\mu_\beta) = 2\delta - \log\det(2\pi\Sigma_\beta) \tag{23}$$

Finally with (22) and (23), we get

$$\eta = \sqrt{\frac{\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}^T \Sigma_\beta \left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}}{2\delta - \log\det(2\pi\Sigma_\beta)}} \tag{24}$$

By setting $\delta = \frac{1}{2}\log\det(2\pi\Sigma_\beta) + \log\tau$ and plugging (24) to (22), we obtain the final solution as

$$\mu_{\text{sg}}(\tau) = \mu_\beta + \frac{\sqrt{2\log\tau}\,\Sigma_\beta\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}}{\sqrt{\left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}^T \Sigma_\beta \left[\nabla_a Q(s,a)\right]_{a=\mu_\beta}}}, \tag{25}$$

which completes the proof. □

B.2 Proof of Proposition 3.2

Proposition 3.2. *By applying the first inequality of Lemma 3.1 to the constraint of* (11)*, we can derive an optimization problem that lower bounds* (11)

$$\begin{aligned} \max_{\mu} \quad & \mu^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta} \\ \textit{s.t.} \quad & \max_i \left\{ -\frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1}(\mu-\mu_i) - \frac{1}{2}\log\det(2\pi\Sigma_i) + \log\lambda_i \right\} \geq -\delta, \end{aligned} \tag{26}$$

and the closed-form solution to (12) *is given by*

$$\begin{aligned} & \mu_{lse}(\tau) = \operatorname*{arg\,max}_{\bar{\mu}_i(\delta)} \quad \bar{\mu}_i^T \left[\nabla_a Q(s,a)\right]_{a=\mu_i}, \quad \textit{s.t.} \quad \delta = \frac{1}{2}\min_i \left\{\log\lambda_i \det(2\pi\Sigma_i)\right\} + \log\tau \\ & \textit{where} \quad \bar{\mu}_i(\delta) = \mu_i + \sqrt{\frac{2(\delta+\log\lambda_i) - \log\det(2\pi\Sigma_i)}{\left[\nabla_a Q(s,a)\right]_{a=\mu_i}^T \Sigma_i \left[\nabla_a Q(s,a)\right]_{a=\mu_i}}} \Sigma_i \left[\nabla_a Q(s,a)\right]_{a=\mu_i} \end{aligned} \tag{27}$$

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

$$\pi_\beta = \sum_{i=1}^{N} \lambda_i \mathcal{N}(\mu_i, \Sigma_i), \tag{28}$$

We first divide the optimization problem (26) into N sub-problems, with each sub-problem i given by

$$\begin{aligned} \max_{\mu} \quad & \mu^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta} \\ \text{s.t.} \quad & -\frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1}(\mu-\mu_i) - \frac{1}{2}\log\det(2\pi\Sigma_i) + \log\lambda_i \geq -\delta, \end{aligned} \tag{29}$$

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 $\breve{\mu} \notin \{\bar{\mu}_i(\delta) | i = 1 \ldots N\}$, s.t., $\breve{\mu}$ is the maximizer of (26). We can show this by contradiction. Suppose there exists a $\breve{\mu} \notin \{\bar{\mu}_i(\delta) | i = 1 \ldots N\}$ maximizing (26), there exists at least one $j \in \{1, \ldots, N\}$ s.t.

$$-\frac{1}{2}(\breve{\mu}-\mu_j)^T \Sigma_j^{-1}(\breve{\mu}-\mu_j) - \frac{1}{2}\log\det(2\pi\Sigma_j) + \log\lambda_j \geq -\delta. \tag{30}$$

Since $\breve{\mu}$ is the maximizer of (26), 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 (26) has to be given by

$$\operatorname{arg\,max}_{\bar{\mu}_i} \quad \bar{\mu}_i^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta} \quad \text{where} \quad \bar{\mu}_i \in \{\bar{\mu}_i(\delta) | i = 1 \ldots N\} \tag{31}$$

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

$$\begin{aligned} \max_{\mu} \quad & \mu^T \left[\nabla_a Q(s,a)\right]_{a=\mu_i} \\ \text{s.t.} \quad & \frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1}(\mu-\mu_i) \leq \delta - \frac{1}{2}\log\det(2\pi\Sigma_i) + \log\lambda_i, \end{aligned} \tag{32}$$

Note that problem (32) is also a QCLP similar to the problem (5). Therefore, we can derive its solution by following similar procedures as in Appendix B.1, resulting in

$$\bar{\mu}_i(\delta) = \mu_i + \sqrt{\frac{2(\delta+\log\lambda_i) - \log\det(2\pi\Sigma_i)}{\left[\nabla_a Q(s,a)\right]_{a=\mu_i}^T \Sigma_i \left[\nabla_a Q(s,a)\right]_{a=\mu_i}}} \Sigma_i \left[\nabla_a Q(s,a)\right]_{a=\mu_i}. \tag{33}$$

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

B.3 Proof of Proposition 3.3

Proposition 3.3. *By applying the second inequality of Lemma 3.1 to the constraint of* (11)*, we can derive an optimization problem that lower bounds* (11)

$$\begin{aligned} \max_{\mu} \quad & \mu^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta} \\ \textit{s.t.} \quad & \sum_{i=1}^{N} \lambda_i \left(-\frac{1}{2}\log\det(2\pi\Sigma_i) - \frac{1}{2}(\mu-\mu_i)^T \Sigma_i^{-1} (\mu-\mu_i) \right) \geq -\delta \end{aligned} \tag{34}$$

and the closed-form solution to (14) *is given by*

$$\begin{aligned} &\mu_{\textit{jensen}}(\tau) = \bar{\mu} + \sqrt{\frac{2\log\tau - \sum_{i=1}^{N}\lambda_i \mu_i^T \Sigma_i^{-1}\mu_i + \bar{\mu}^T \overline{\Sigma}^{-1}\bar{\mu}}{[\nabla_a Q(s,a)]_{a=\bar{\mu}}^T \overline{\Sigma}[\nabla_a Q(s,a)]_{a=\bar{\mu}}}}\ \overline{\Sigma}[\nabla_a Q(s,a)]_{a=\bar{\mu}}, \\ &\textit{where } \overline{\Sigma} = \left(\sum_{i=1}^{N}\lambda_i \Sigma_i^{-1}\right)^{-1}, \quad \bar{\mu} = \overline{\Sigma}\left(\sum_{i=1}^{N}\lambda_i\Sigma_i^{-1}\mu_i\right), \quad \delta = \log\tau + \frac{1}{2}\sum_{i=1}^{N}\lambda_i \log\det(2\pi\Sigma_i) \end{aligned} \tag{35}$$

Proof. Note that problem (34) is also a QCLP. Before deciding the value of a_β, we first derive its Lagrangian with a general a_β below

$$L = \mu^T \left[\nabla_a Q(s,a)\right]_{a=a_\beta} - \eta \left(\sum_{i=1}^{N} \lambda_i \left(\frac{1}{2}\log\det(2\pi\Sigma_i) + \frac{1}{2}(\mu-\mu_i)^T\Sigma_i^{-1}(\mu-\mu_i) \right) - \delta \right) \tag{36}$$

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

$$\nabla_\mu L = \left[\nabla_a Q(s,a)\right]_{a=a_\beta} - \eta \left(\sum_{i=1}^{N} \lambda_i \left(\Sigma_i^{-1}(\mu-\mu_i) \right) \right) \tag{37}$$

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

$$\begin{aligned} \mu &= \left(\sum_{i=1}^{N}\lambda_i\Sigma_i^{-1}\right)^{-1} \left(\sum_{i=1}^{N}\lambda_i\Sigma_i^{-1}\mu_i\right) + \frac{1}{\eta}\left(\sum_{i=1}^{N}\lambda_i\Sigma_i^{-1}\right)\left[\nabla_a Q(s,a)\right]_{a=a_\beta} \\ &= \bar{\mu} + \frac{1}{\eta}\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=a_\beta}, \\ \text{where} \quad \overline{\Sigma} &= \left(\sum_{i=1}^{N}\lambda_i\Sigma_i^{-1}\right)^{-1}, \quad \bar{\mu} = \overline{\Sigma}\left(\sum_{i=1}^{N}\lambda_i\Sigma_i^{-1}\mu_i\right), \end{aligned} \tag{38}$$

Equation 38 shows that the final solution to the problem (34) will be a shift from the pseudo-mean $\bar{\mu}$. Therefore, setting $a_\beta = \bar{\mu}$ becomes a natural choice.

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

$$\sum_{i=1}^{N}\lambda_i \left(\mu-\mu_i\right)^T \Sigma_i^{-1}\left(\mu-\mu_i\right) = 2\delta - \sum_{i=1}^{N}\lambda_i\log\det(2\pi\Sigma_i) \tag{39}$$

Plugging (34) into (39) gives the equation below

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

The LHS of (40) can be reformulated as

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

We note that the second line of (41)'s RHS can be reduced to

$$\begin{array}{rl} & \frac{2}{\eta} \sum_{i=1}^{N} \lambda_i \left(\overline{\Sigma} \left[\nabla_a Q(s,a) \right]_{a=\bar{\mu}} \right)^T \Sigma_i^{-1} \left(\bar{\mu} - \mu_i \right) \\ = & \frac{2}{\eta} \left(\overline{\Sigma} \left[\nabla_a Q(s,a) \right]_{a=\bar{\mu}} \right)^T \left(\left(\sum_{i=1}^{N} \lambda_i \Sigma_i^{-1} \right) \bar{\mu} - \sum_{i=1}^{N} \lambda_i \Sigma_i^{-1} \mu_i \right) \\ = & \frac{2}{\eta} \left(\overline{\Sigma} \left[\nabla_a Q(s,a) \right]_{a=\bar{\mu}} \right)^T \left(\overline{\Sigma}^{-1} \bar{\mu} - \overline{\Sigma}^{-1} \left(\overline{\Sigma} \sum_{i=1}^{N} \lambda_i \Sigma_i^{-1} \mu_i \right) \right) \\ = & \frac{2}{\eta} \left(\overline{\Sigma} \left[\nabla_a Q(s,a) \right]_{a=\bar{\mu}} \right)^T \left(\overline{\Sigma}^{-1} \bar{\mu} - \overline{\Sigma}^{-1} \bar{\mu} \right) \\ = & 0 \end{array}. \tag{42}$$

Therefore, (41) can be further reformulated as

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

To this point, (40) can be reformulated as

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

We can thus express η as below

$$\eta = \sqrt{\frac{\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}^T \overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}}{2\delta - \sum_{i=1}^{N}\lambda_i \log\det(2\pi\Sigma_i) - \sum_{i=1}^{N}\lambda_i\left(\bar{\mu}-\mu_i\right)^T\Sigma_i^{-1}\left(\bar{\mu}-\mu_i\right)}} \tag{45}$$

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

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

Finally, plugging (46) into (38), with $a_\beta = \bar{\mu}$, we have

$$\mu_{\text{jensen}}(\tau) = \bar{\mu} + \sqrt{\frac{2\log\tau - \sum_{i=1}^{N}\lambda_i\mu_i^T\Sigma_i^{-1}\mu_i + \bar{\mu}^T\overline{\Sigma}^{-1}\bar{\mu}}{\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}^T\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}}}\,\overline{\Sigma}\left[\nabla_a Q(s,a)\right]_{a=\bar{\mu}}, \tag{47}$$

which completes the proof. □

B.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) [28] 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 [1]. Fortunately, it is known that FQI is statistically efficient (*e.g.* [29]) 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 [38, 39]. Recently, [30] 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 B.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 [14], 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* [40] and covering arguments.

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

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

Define the first-order approximation $\bar{Q}^{\pi_\beta}(s,a) := a^T\left[\nabla_\zeta \hat{Q}^{\pi_\beta}(s,\zeta)\right]_{\zeta=a_\beta}$, then the approximation error is defined as:

$$C_{\mathrm{CFPI}}(s,a) := |\bar{Q}^{\pi_\beta}(s,a) - \hat{Q}^{\pi_\beta}(s,a)| = \left|a^T\left[\nabla_\zeta \hat{Q}^{\pi_\beta}(s,\zeta)\right]_{\zeta=a_\beta} - \hat{Q}^{\pi_\beta}(s,a)\right|.$$

Under the constraint $D\left(\pi(\cdot\mid s), \pi_\beta(\cdot\mid s)\right) \le \delta$ (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 B.1 (Restatement of Theorem 3.1)**.** *Let $\hat{\pi}_1$ be the policy obtained after the CFPI update (Line 2 of Algorithm 1). Then with probability $1-\delta$, ($\mathcal{D}(s,a)$ is number of samples at s,a)*

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

Here $C_{\gamma,\delta}$ is the learning coefficient of SARSA, and C_{CFPI} is the error from the first-order approximation (3), (4) using CFPI. Their concrete definitions are deferred to Appendix B.4.

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 $|\hat{Q}^{\hat{\pi}_\beta}(s,a) - Q^{\hat{\pi}_\beta}(s,a)| \le \frac{C_{\gamma,\delta}}{\sqrt{\mathcal{D}(s,a)}}$, $\forall s,a \in \mathcal{S}\times\mathcal{A}$ with probability $1-\delta$ and this is due to previous on-policy evaluation result (*e.g.* [30]). Also denote $\hat{\pi}_1 := \arg\max_\pi \bar{Q}^{\hat{\pi}_\beta}$.

By Lemma B.1,

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

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 $\hat{\pi}_1 := \arg\max_\pi \bar{Q}^{\hat{\pi}_\beta}$.

□

C Detailed Procedures to obtain Equation 16

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,

$$\begin{aligned} \delta_{\text{lse}}(\tau) &= \log\tau + \min_i \left\{ \frac{1}{2}\log\det(2\pi\Sigma_i) - \log\lambda_i \right\}, \quad \text{(Proposition 3.2)} \\ \delta_{\text{jensen}}(\tau) &= \log\tau + \frac{1}{2}\sum_{i=1}^{N}\lambda_i \log\det(2\pi\Sigma_i), \quad \text{(Proposition 3.3)} \end{aligned} \tag{48}$$

We note that the δ defined in Proposition 3.2 **is a typo** (Line 150 of the main paper), but it will not lead to an incorrect derivation. We corrected its definition in (48) and make sure that our implementation is consistent with the definition in (48).

In the rest of this section, we will first provide intuition for the design choices (48). Secondly, we will describe the line search algorithm to select the $\log\tau \in \{\log\tau_j \mid k = 1\ldots K\}$ for the generation of $\mu_{\text{mg}}(\tau)$, where each $\log\tau_k$ is sampled from a reasonably wide range.

C.1 Justify the design of Equation 48

Recall that the Gaussian Mixture behavior policy is constructed by

$$\pi_\beta = \sum_{i=1}^{N}\lambda_i \mathcal{N}(\mu_i, \Sigma_i). \tag{49}$$

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

$$\breve{\pi}_i(\mu_i) = \lambda_i\pi_i(a) = \lambda_i \det(2\pi\Sigma_i)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(a-\mu_i)^T\Sigma_i^{-1}(a-\mu_i)\}, \tag{50}$$

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

$$\begin{aligned} \delta_{\text{lse}}(\tau) &= \log\tau + \min_i \left\{ \frac{1}{2}\log\det(2\pi\Sigma_i) - \log\lambda_i \right\} \\ &= -\left(\max_i \left\{ \log\lambda_i - \frac{1}{2}\log\det(2\pi\Sigma_i) \right\} - \log\tau \right). \\ &= -\max_i \left\{ \log\frac{1}{\tau}\breve{\pi}_i(\mu_i) \right\}. \end{aligned} \tag{51}$$

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 (29) due to the KKT conditions, $\forall j \in \{1,\ldots,N\}$.

$$\begin{aligned} -\frac{1}{2}(\bar{\mu}_j-\mu_j)^T\Sigma_j^{-1}(\bar{\mu}_j-\mu_j) - \frac{1}{2}\log\det(2\pi\Sigma_j) + \log\lambda_j = -\delta_{\text{lse}}(\tau) \\ \iff \quad \log\breve{\pi}_j(\bar{\mu}_j) = \max_i\left\{\log\frac{1}{\tau}\breve{\pi}_i(\mu_i)\right\} \quad \iff \quad \breve{\pi}_j(\bar{\mu}_j) = \frac{1}{\tau}\max_i\{\breve{\pi}_i(\mu_i)\} \end{aligned} \tag{52}$$

To elaborate the design of $\delta_{\text{jensen}}(\tau)$, we first recall that the constraint of problem (14) is given by

$$\sum_{i=1}^{N}\lambda_i\left(-\frac{1}{2}\log\det(2\pi\Sigma_i) - \frac{1}{2}(\mu-\mu_i)^T\Sigma_i^{-1}(\mu-\mu_i)\right) \geq -\delta_{\text{jensen}}(\tau). \tag{53}$$

Note that the LHS of (53) is a concave function w.r.t μ. Thus, we can obtain its maximum by setting its derivatives (54) to zero

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

Algorithm 2 Action selection of $\mu^\star_{\text{mg}}$ with a line search on $\log \tau$

```
Input: State $s$, Baseline policy $\hat{\pi}_b = \sum_{i=1}^N \hat{\lambda}_i \mathcal{N}(\hat{\mu}_i, \hat{\Sigma}_i)$, value function $\hat{Q}$, $\log \tau$ range $[\underline{\tau}, \overline{\tau}]$, number of $\tau$ to sample $K$
1: Uniformly sample $\{\log \tau_1, \ldots, \log \tau_K\}$ from $[\underline{\tau}, \overline{\tau}]$
2: Initialize a candidate action set $\mathcal{C} \leftarrow \{\}$
3: for each $\log \tau_j \in \{\tau_1, \ldots, \tau_K\}$ do
4:     Calculate $\mu_{\text{lse}}(\tau_j)$ with Equation 13, $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mu_{\text{lse}}(\tau_j)\}$
5:     Calculate $\mu_{\text{jensen}}(\tau_j)$ with Equation 15, $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mu_{\text{jensen}}(\tau_j)\}$
6: end for
7: Return the action $\mu^\star_{\text{mg}} = \arg\max_{a \in \mathcal{C}} \hat{Q}(s, a)$
```

Algorithm 3 Action selection of $\mu^\star_{\text{mg}}$ with a line search on $\log \tau$ (practice)

```
Input: State $s$, Baseline policy $\hat{\pi}_b = \sum_{i=1}^N \hat{\lambda}_i \mathcal{N}(\hat{\mu}_i, \hat{\Sigma}_i)$, value function $\hat{Q}$, $\log \tau$ range $[\underline{\tau}, \overline{\tau}]$, number of $\tau$ to sample $K$
1: Uniformly sample $\{\log \tau_1, \ldots, \log \tau_K\}$ from $[\underline{\tau}, \overline{\tau}]$
2: Initialize a candidate action set $\mathcal{C} \leftarrow \{\}$
3: for each $\log \tau_j \in \{\tau_1, \ldots, \tau_K\}$ do
4:     Calculate $\{\bar{\mu}_{i,j}\} = \{\bar{\mu}_i(\delta(\tau_j)) \mid i = 1 \ldots N, \lambda_i > \xi\}$ with Equation 13, $\mathcal{C} \leftarrow \mathcal{C} \cup \{\bar{\mu}_{i,j}\}$
5:     Calculate $\mu_{\text{jensen}}(\tau_j)$ with Equation 15, $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mu_{\text{jensen}}(\tau_j)\}$
6: end for
7: Return the action $\mu^\star_{\text{mg}} = \arg\max_{a \in \mathcal{C}} \hat{Q}(s, a)$
```

Interestingly, we can find that the solution is given by $\mu = \bar{\mu}$. Plugging $\mu = \bar{\mu}$ into the LHS of (53), we can obtain its maximum as below

$$\begin{aligned} &-\frac{1}{2}\sum_{i=1}^N \lambda_i \log\det(2\pi\Sigma_i) - \frac{1}{2}\sum_{i=1}^N \lambda_i (\bar{\mu} - \mu_i)^T \Sigma_i^{-1} (\bar{\mu} - \mu_i) \\ \le &\sum_{i=1}^N \lambda_i \left(-\frac{1}{2}\log\det(2\pi\Sigma_i)\right) = \sum_{i=1}^N \lambda_i \log \pi_i(\mu_i) \end{aligned} \tag{55}$$

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

$$\delta_{\text{jensen}}(\tau) = \log \tau + \frac{1}{2}\sum_{i=1}^N \lambda_i \log\det(2\pi\Sigma_i) = -(\sum_{i=1}^N \lambda_i \log \pi_i(\mu_i) - \log \tau) \tag{56}$$

C.2 Line search algorithm

Algorithm 2 gives the detailed procedures to obtain $\mu^\star_{\text{mg}}$ with a line search on $\log \tau$. It augments the action selection performed in Equation 16, and can be easily adapted to our CFPI operator $\mathcal{I}_{\text{mg}}$.

In practice, for each $\log \tau_j$, we instead add the N candidate actions $\bar{\mu}_i(\delta(\tau_j))$ to the candidate set $\mathcal{C}$ when calculating the $\mu_{\text{lse}}(\tau_j)$ with Equation 13 (Line 4 of Algorithm 2), where

$$\delta(\tau_j) = \frac{1}{2}\min_i \{\log \lambda_i \det(2\pi\Sigma_i)\} + \log \tau_j. \tag{57}$$

Each $\bar{\mu}_i(\delta(\tau_j))$ is the solution for the corresponding sub-problem (29). Therefore, we can derive a practical version of Algorithm 2 as is shown in Algorithm 3. Note that (Line 3 - 7 of Algorithm 3) can be performed in parallel, thus the additional computation cost is a single forward pass of the Q function with the batch size scaled linearly with the number of Gaussian components N.

D Multi-step and iterative algorithms

We first correct a typo in Line 179 of the main paper. Algorithm 1 should yield an one-step algorithm when $T = 0$ instead of $T = 1$.

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 [41, 42, 43] with a target network [19] has been demonstrated to be an effective and successful paradigm to perform policy evaluation [7, 9, 44, 45, 46] in deep (Offline) RL. When instantiating a multi-step or iterative algorithms from Algorithm 1, one can also consider the other policy evaluation operators by incorporating more optimization techniques.

E Hyper-parameter settings and training details

For all methods we proposed in Table 1, Table 2, and Table 3, we obtain the mean and standard deviation of each method across 10 seeds. Each seed contains individual training process and evaluates the policy for 100K environment steps.

E.1 HP and training details for methods in Table 1 and Table 3

Table 4 includes the HP of methods evaluated on the Gym-MuJoCo domain. We use the Adam [47] optimizer for all learning algorithms and normalize the states in each dataset following the practice of TD3+BC [9]. Note that our one-step offline RL algorithms presented in Table 1 (Our $\mathcal{I}_{\text{MG}}$) and Table 3 ($\mathcal{I}_{\text{MG}}$, $\mathcal{I}_{\text{SG}}$, 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 [47]
	Normalize states	True
	Policy architecture	MLP
	Trivial Gaussian threshold ξ	0.05
MG-BC HP	Gaussian components (N)	4
	Number of gradient steps	500K
	Mini-batch size	256
	Policy learning rate	1e-4
	Policy hidden dim	256
	Policy hidden layers	3
	Policy activation function	ReLU
SG-BC HP	Number of gradient steps	500K
	Mini-batch size	512
	Policy learning rate	1e-4
	Policy hidden dim	256
	Policy hidden layers	3
	Policy activation function	ReLU
SARSA HP	Number of gradient steps	Table 6
	Critic architecture	IQN [35]
	Critic hidden dim	256
	Critic hidden layers	3
	Critic activation function	ReLU
	Number of quantiles N_{q}	8
	Modeled quantile	0.7
	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) HP	Number of sampled τ (K)	10
	$\log \tau$ range $[\underline{\tau}, \overline{\tau}]$	[0, 0] for Hopper-M-E; [0.5, 1.0] for the others
$\mathcal{I}_{\text{MG}}$ (Table 3) & $\mathcal{I}_{\text{SG}}$ (Table 3) HP	Number of sampled τ (K)	10
	$\log \tau$ range $[\underline{\tau}, \overline{\tau}]$	[0.5, 1.0] for all tasks
MG-EBCQ (Table 3) & SG-EBCQ (Table 3) HP	Number of candidate actions N_{bcq}	10

Table 4: Hyperparameters for our methods in Table 1 and Table 3.

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 [48, 49, 50] or variational Bayes [51], 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 better capture the underlying behavior policy.

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 [24, 44]. We train the policy for 500K gradient steps.

SARSA. We parameterize the value function with the IQN [35] architecture and train it to model the distribution $Z^\beta : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{Z}$ of the behavior return via quantile regression, where $\mathcal{Z}$ is the action-value distributional space [52] defined as

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

We define the CDF function of Z^β as $F_{Z^\beta}(z) = Pr(Z^\beta < z)$, leading to the quantile function [53] $F^{-1}_{Z^\beta}(\rho) := \inf\{z \in \mathbb{R} : \rho \leq F_{Z^\beta}(z)\}$ 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 $\hat{\mathcal{T}}^\beta_{D} : \mathcal{Z} \rightarrow \mathcal{Z}$ 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 [35, 52, 54], and we can finally learn Z^β by minimizing the quantile regression loss following a similar procedures as in [52].

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 [35] 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 [45] given by the Polyak averaging of θ. We refer interested readers to [35, 52, 54] 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 3 extract the value function by $\hat{Q}_0 = \hat{Z}^\beta_\rho$, and use $\hat{Q}_0$ to guide the action selections. In Appendix F.1, we show that the performance of our $\mathcal{I}_\text{MG}$ (Table 1) is robust to a reasonable range of ρ on the Gym-MuJoCo domain.

Since our methods still need to query out-of-buffer action values during rollout, we employed the conventional double Q-learning [46] 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

$$\hat{Q}_0(s,a) = \min_{k=1,2} \hat{Q}^k_0(s,a) \tag{61}$$

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 F.4. And the number of gradient steps for each dataset can be found in Table 6.

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. When selecting the action, we perform a line search on $\log \tau$ according the Algorithm 3. We note that we manually reduce $\mathcal{I}_{\text{MG}}$ to MG-MS when $\tau = 0$ by only considering the mean of each non-trivial Gaussian component. To decide the range $[\underline{\tau}, \overline{\tau}]$ to perform a line search, we first obtain the performance of $\mathcal{I}_{\text{MG}}$ with a single $\log \tau \in \{0, 0.5, 1.0, 1.5, 2.0\}$. We obtain the average performance across 5 seeds for each setting on the 6 Medium-Replay and Medium tasks as is shown in Appendix F.2. We then repeats the experiments with a linear search on $\log \tau$, and select the best $[\underline{\tau}, \overline{\tau}] \in \{[0, 0], [0.5, 1.0], [0.5, 1.5], [1.0, 1.5]\}$. As a result, we set $[\underline{\tau}, \overline{\tau}] = [0.5, 1.0]$ for all datasets except Hopper-M-E. We note that Hopper-M-E requires a extremely small $\log \tau$ as is shown in Appendix F.2.

$\mathcal{I}_{\text{MG}}$ (Table 3) & $\mathcal{I}_{\text{SG}}$ (Table 3). Different from the results in Table 1, we use the same $[\underline{\tau}, \overline{\tau}] = [0.5, 1.0]$ for all datasets including Hopper-M-E to obtain the performance of $\mathcal{I}_{\text{MG}}$ in Table 3. In this way, we aim to better understand the effectiveness of each component of our methods. To fairly compare $\mathcal{I}_{\text{MG}}$ and $\mathcal{I}_{\text{SG}}$, we tune the range $[\underline{\tau}, \overline{\tau}]$ for $\mathcal{I}_{\text{SG}}$ in a similar way. We first obtain the performance of $\mathcal{I}_{\text{SG}}$ with a single $\log \tau \in \{0, 0.5, 1.0, 1.5, 2.0\}$. We obtain the average performance across 5 seeds for each setting on the 6 Medium-Replay and Medium tasks. We then repeats the experiments with a linear search on $\log \tau$, and select the best $[\underline{\tau}, \overline{\tau}] \in \{[0, 0], [0.5, 1.0], [0.5, 1.5], [1.0, 1.5]\}$. Finally, we pick the best $[\underline{\tau}, \overline{\tau}] = [0.5, 1.0]$ and obtain its performance on all datasets with 10 seeds.

MG-EBCQ (Table 3) & SG-EBCQ (Table 3). We tune the number of candidate actions N_{bcq} from the same range $\{2, 5, 10, 20, 50, 100\}$ as is in [12]. For each N_{bcq}, we obtain its average performance for all tasks across 10 seeds. We highlight that we separately tune the N_{bcq} for MG-EBCQ and SG-EBCQ, although $N_{\text{bcq}} = 10$ achieves the best overall performance for both methods. Moreover, MG-EBCQ (SG-EBCQ) uses the same behavior policy and value function as is in $\mathcal{I}_{\text{MG}}$ ($\mathcal{I}_{\text{SG}}$).

E.2 HP and training details for methods in Table 2

	Hyperparameter	Value
Shared HP	Normalize states	False
IQL HP	Optimizer	Adam [47]
	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
$\mathcal{I}_{\text{sg}}(\pi_{\text{IQL}}, Q_{\text{IQL}})$ HP	Number of sampled τ (K)	100
	$\log \tau$ range $[\underline{\tau}, \overline{\tau}]$	selected from $[0.4, 0.6], [0.5, 0.75], [1, 2], [1.5, 2]$

Table 5: Hyperparameters for methods in Table 2

Table 5 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 [3], 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 F.5.

Note that IQL [11] reported inconsistent offline experiment results on AntMaze in its paper's Table 1, Table 2, Table 5, and Table 6 [4]. We suspect that these results are obtained from different sets of random seeds. In Appendix F.5, we present all these results in our Table 7.

To obtain the performance for $\mathcal{I}_{\text{sg}}(\pi_{\text{IQL}}, Q_{\text{IQL}})$, we first do a HP search over $\log \tau \in \{0, 0.5, 1.0, 1.5, 2.0\}$ with 5 seeds. We then obtain the average performance across 10 seeds by choosing promising ranges according to the first search, finally choosing the best $[\underline{\tau}, \overline{\tau}] \in \{[0.4, 0.6], [0.5, 0.75], [1, 2], [1.5, 2]\}$ for each dataset.

[3] `https://github.com/rail-berkeley/rlkit/tree/master/examples/iql`

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

F Additional Experiments

F.1 Ablation study of the IQN quantile ρ

Recall that we set $\hat{Q}_0 = \hat{Z}^{\beta}_{\rho}$ to obtain the performance for our $\mathcal{I}_{\text{MG}}$ in Table 1, where we set $\rho = 0.7$ following the practice of [11]. In this section, we will examine the effect of ρ on the performance of $\mathcal{I}_{\text{MG}}$. As is shown in Fig. 3, we can see that the performance of our $\mathcal{I}_{\text{MG}}$ is robust to the value of ρ. We evaluate over 5 seeds and each seed for 100K environment steps. Therefore, the results for $\rho = 0.7$ in Fig. 3 are slightly different from Table 1 and Table 3.

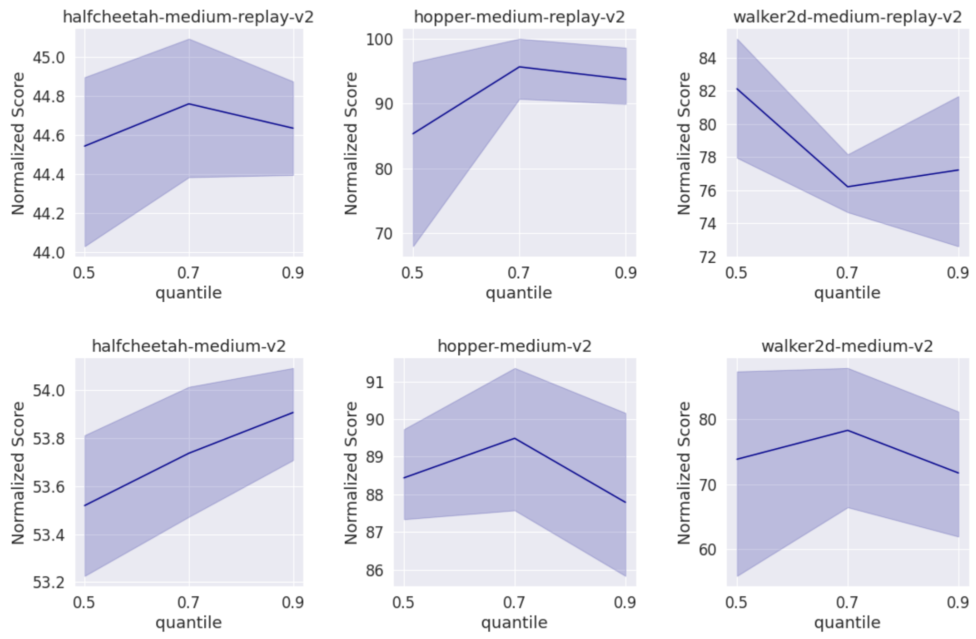


Figure 3: Performance of $\mathcal{I}_{\text{MG}}$ with varying quantiles $\hat{Z}^{\beta}_{\rho}$. We set the line search range $[\underline{\tau}, \overline{\tau}] = [0.5, 1.0]$. The other HP can be found in Table 4. Each variant averages returns over 5 seeds with 100K environment steps for each seed. The shaded area denotes bootstrapped 95% CI.

F.2 Complete experiment results on the effect of the HP τ

Fig. 4 presents additional results in compensation for the results in Sec. 5.3. We note that Hopper-Medium-Expert-v2 requires a much smaller $\log \tau$ to obtain good performance than the other tasks.

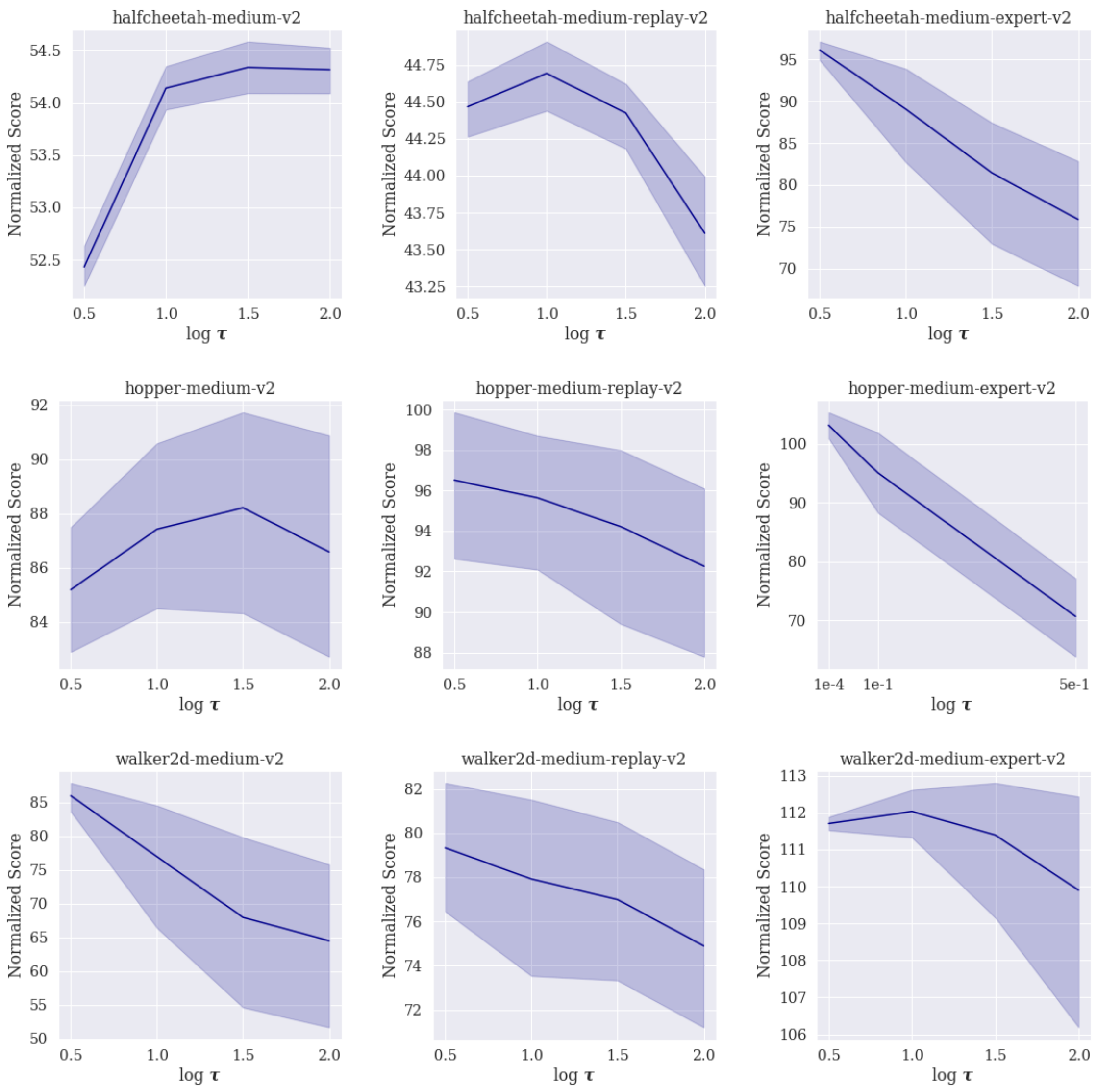


Figure 4: Performance of $\mathcal{I}_{\text{MG}}$ with varying $\log \tau$. We obtain the results without performing line search on $\log \tau$. The other HP can be found in Table 4. Each variant averages returns over 10 seeds with 100K environment steps for each seed. The shaded area denotes bootstrapped 95% CI.

F.3 Modeling the value network with conventional MLP

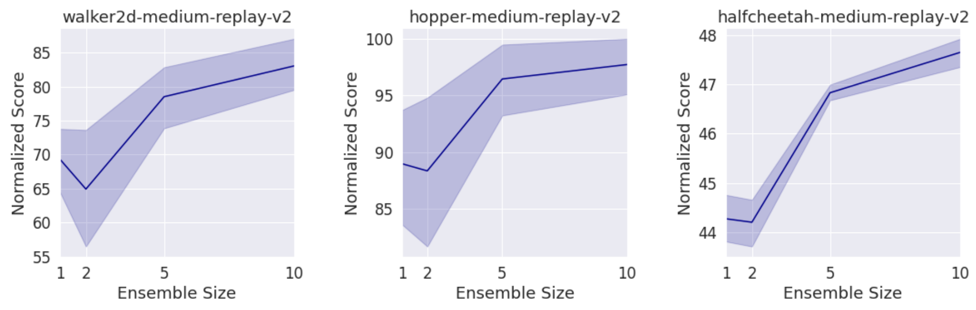


Figure 5: Performance of $\mathcal{I}_{\text{MG}}$ with varying ensemble sizes. Each variant averages returns over 8 seeds with 100K environment steps for each seed. Each Q-value network is modeled by a 3-layer MLP. The shaded area denotes bootstrapped 95% CI.

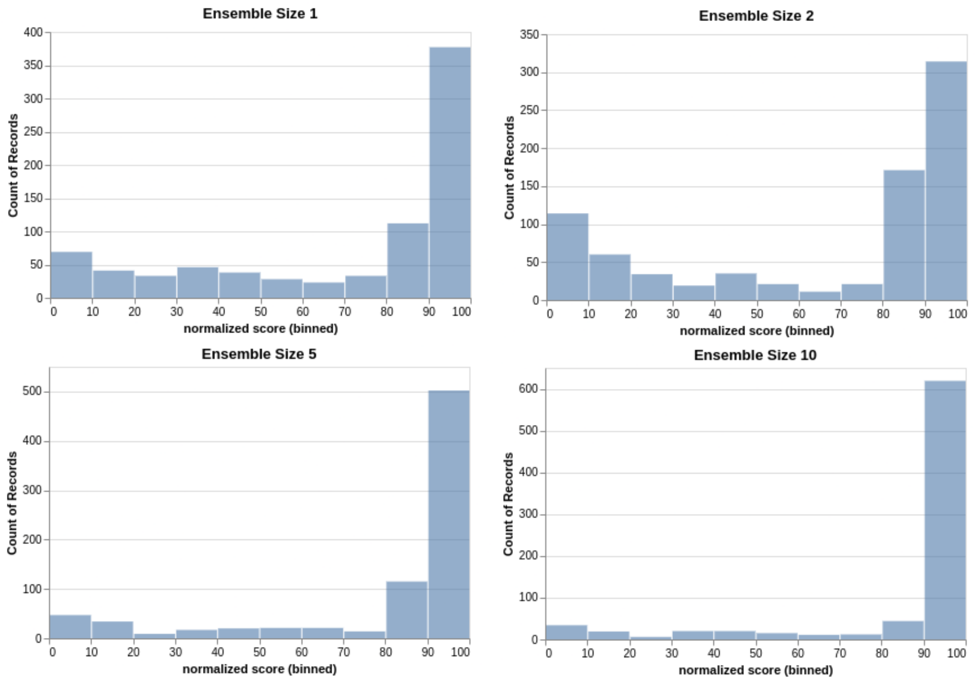


Figure 6: Performance of $\mathcal{I}_{\text{MG}}$ with varying ensemble sizes on Walker2d-Medium-Replay-v2. Each variant aggregates returns over 8 seeds with 100K environment steps for each seed. Each Q-value network is modeled by a 3-layer MLP. With lower ensemble size, the performance exhibits large variance across different episodes.

Our experiments in Sec. 5.1 rely on learning a value function to model the Z_ρ^β with the IQN [35] 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 $\hat{Q}_{\theta_k}^{\text{MLP}}$ uses ReLU activation and is parameterized with θ_k, including 3 hidden layers of width 256. We train each $\hat{Q}_{\theta_k}^{\text{MLP}}$ by minimizing the bellman error below

$$L(\theta_k) = \mathbb{E}_{(s,a,r,s',a')\sim\mathcal{D}}\left[r + \gamma\hat{Q}^{\text{MLP}}(s', a'; \bar{\theta}_k) - \hat{Q}^{\text{MLP}}(s, a; \theta_k)\right], \tag{62}$$

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 61 can be reformulated as

$$\begin{aligned}\hat{Q}_0(s,a) = \min_{k=1,2} \hat{Q}_0^k(s,a) &= \frac{1}{2}|\hat{Q}_0^1(s,a) + \hat{Q}_0^2(s,a)| - \frac{1}{2}|\hat{Q}_0^1(s,a) - \hat{Q}_0^2(s,a)| \\ &= \hat{\mu}_Q(s,a) - \hat{\sigma}_Q(s,a),\end{aligned} \tag{63}$$

where $\hat{\mu}_Q$ and $\hat{\sigma}_Q$ calculate the mean and standard deviation of Q value [33]. In the case with an ensemble of Q, we obtain $\hat{Q}_0(s,a)$ by generalizing (63) as below

$$\begin{aligned}&\hat{Q}_0(s,a) = \hat{\mu}_Q^{\text{MLP}} - \sqrt{\frac{1}{M}\sum_{k=1}^{M}\left(\hat{Q}^{\text{MLP}}(s,a;\theta_k) - \hat{\mu}_Q^{\text{MLP}}\right)^2}, \\ &\text{where} \quad \hat{\mu}_Q^{\text{MLP}} = \frac{1}{M}\sum_{k=1}^{M}\hat{Q}^{\text{MLP}}(s,a;\theta_k).\end{aligned} \tag{64}$$

Other than the Q-value network, we applied the same setting as $\mathcal{I}_{\text{MG}}$ in Table 3. Fig. 5 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. 6.

F.4 How to decide the number of gradient steps for SARSA training?

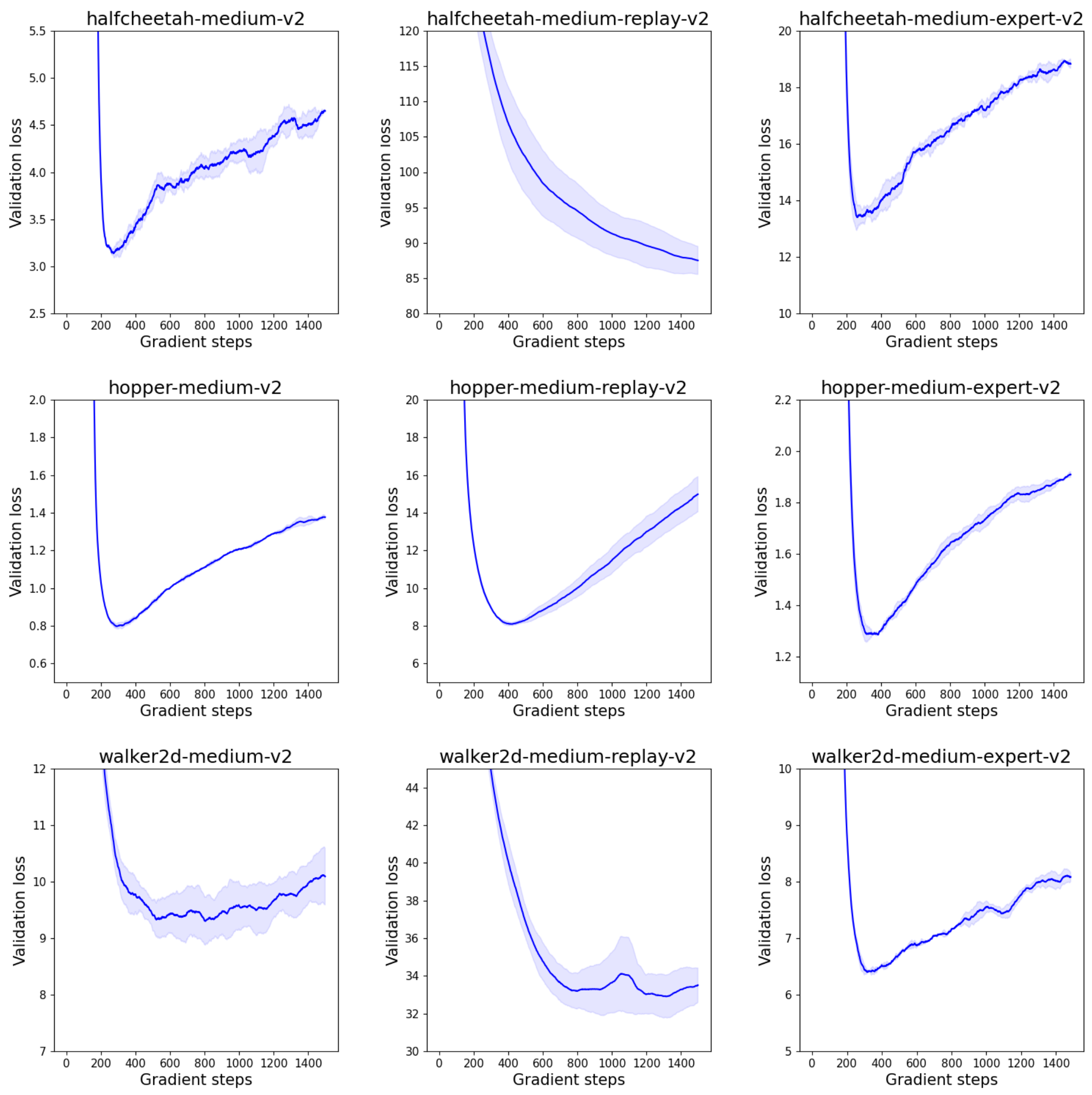


Figure 7: $\mathcal{L}_{\text{val}}$ on each dataset from the Gym-MuJoCo domain. We can observe that the model overfits to the training set when training for too may gradient steps. Each figure averages the validation loss over 2 folds with the same training seed. The shaded area denotes one standard 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 F.3. 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 traininning 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

$$\mathcal{L}_{\text{val}} = \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{val}}} ||\hat{Q}^{\beta}_{\text{all}}(s, a) - \hat{Q}^{\beta}(s, a)||^2 \tag{65}$$

Fig. 7 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 6: Gradient steps for the SARSA training

Dataset	Gradient steps (K)
HalfCheetah-Medium-v2	200
Hopper-Medium-v2	400
Walker2d-Medium-v2	600
HalfCheetah-Medium-Replay-v2	1500
Hopper-Medium-Replay-v2	400
Walker2d-Medium-Replay-v2	1100
HalfCheetah-Medium-Expert-v2	400
Hopper-Medium-Expert-v2	300
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. 7 as is listed in Table 6.

F.5 Our reproduced IQL training curves

We use the PyTorch [55] Implementation of IQL from RLkit [5] to obtain its policy π_{IQL} and value function Q_{IQL}. We do not use the official implementation[6] open-sourced by the authors because our CFPI operators are also based on PyTorch. Fig. 8 presents our reproduced training curves of IQL on the 6 datasets from the AntMaze domain.

We note that the IQL paper[7] does not report consistent results in their paper for the offline experiment performance on the AntMaze, as is shown in Table 7. 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.

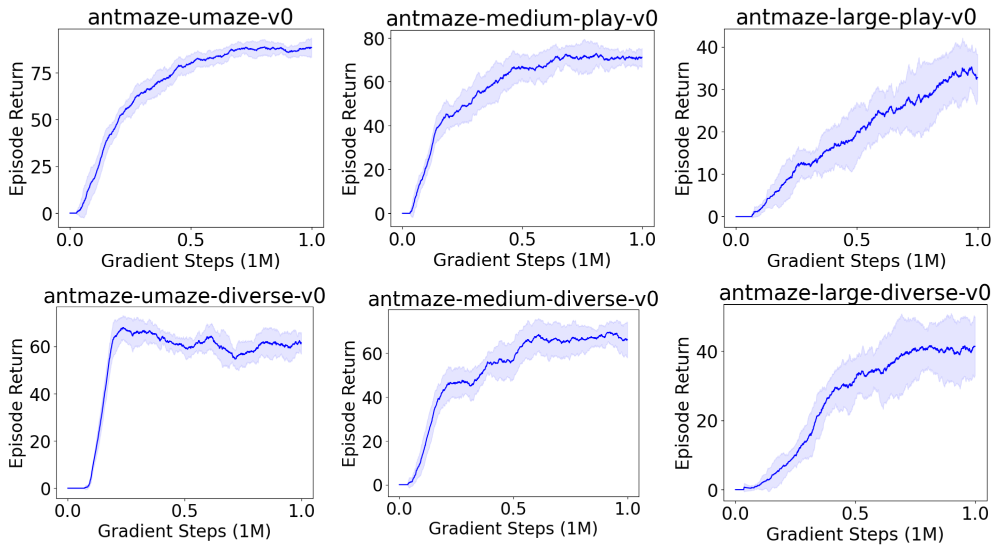


Figure 8: IQL offline training results on AntMaze. Shaded area denotes one standard deviation.

Table 7: 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

[5]`https://github.com/rail-berkeley/rlkit/tree/master/examples/iql`
[6]`https://github.com/ikostrikov/implicit_q_learning`
[7]Link to the IQL paper. IQL's Table 5 & 6 are presented in the supplementary material.

G Computing infrastructure

Our experiments are conducted on various type of 8GPUs machines. Different machines may have different GPU types, which can be 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.

References

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

[2] 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.

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

[4] 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.

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

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

[7] 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.

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

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

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

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

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

[13] 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*, pages 3682–3691. PMLR, 2021.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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, 2020.

[19] 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.

[20] 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.

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

[22] 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.

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

[24] 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.

[25] 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.

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

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

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

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

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

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

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

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

[34] 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.

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

[36] 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.

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

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

[39] 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.

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

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

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

[43] 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.

[44] 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.

[45] 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.

[46] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *CoRR*, abs/1802.09477, 2018.

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

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

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

[50] 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.

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

[52] 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.

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

[54] 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, 2018.

[55] 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.