

BRAC+: Improved Behavior Regularized Actor Critic for Offline Reinforcement Learning

Chi Zhang

Sanmukh Rao Kuppannagari

Viktor K Prasanna

ZHAN527@USC.EDU

KUPPANNA@USC.EDU

PRASANNA@USC.EDU

Hughes Aircraft Electrical Engineering Center, 3740 McClintock Ave, Los Angeles, CA 90089

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

Online interactions with the environment to collect data samples for training a Reinforcement Learning (RL) agent is not always feasible due to economic and safety concerns. The goal of Offline Reinforcement Learning is to address this problem by learning effective policies using previously collected datasets. Standard off-policy RL algorithms are prone to overestimations of the values of out-of-distribution (less explored) actions and are hence unsuitable for Offline RL. Behavior regularization, which constraints the learned policy within the support set of the dataset, has been proposed to tackle the limitations of standard off-policy algorithms. In this paper, we improve the behavior regularized offline reinforcement learning and propose BRAC+. First, we propose quantification of the out-of-distribution actions and conduct comparisons between using Kullback–Leibler divergence versus using Maximum Mean Discrepancy as the regularization protocol. We propose an analytical upper bound on the KL divergence as the behavior regularizer to reduce variance associated with sample based estimations. Second, we mathematically show that the learned Q values can diverge even using behavior regularized policy update under mild assumptions. This leads to large overestimations of the Q values and performance deterioration of the learned policy. To mitigate this issue, we add a gradient penalty term to the policy evaluation objective. By doing so, the Q values are guaranteed to converge. On challenging offline RL benchmarks, BRAC+ outperforms the baseline behavior regularized approaches by 40% ~ 87% and the state-of-the-art approach by 6%.

1. Introduction

Reinforcement Learning (RL) has shown great success in a wide range of applications including board games (Silver et al., 2016), energy systems (Zhang et al., 2019), robotics (Lin, 1992), recommendation systems (Choi et al., 2018), etc. The success of RL relies heavily on extensive online interactions with the environment for exploration. However, this is not always feasible in the real world as it can be expensive or dangerous (Levine et al., 2020).

Offline RL, also known as batch RL, avoids online interactions with the environment by learning from a static dataset that is collected in an offline manner (Levine et al., 2020). While standard off-policy RL algorithms (Mnih et al., 2013; Lillicrap et al., 2016; Haarnoja et al., 2018) can, in theory, be employed to learn from an offline data. In practice, they perform poorly due to distributional shift between the behavior policy (probability distribution of actions conditioned on states as observed in the dataset) of the collected

dataset and the learned policy (Levine et al., 2020). The distributional shift manifests itself in form of overestimation of the out-of-distribution (OOD) actions, leading to erroneous Bellman backups.

Prior works tackle this problem via behavior regularization (Fujimoto et al., 2018b; Kumar et al., 2019; Wu et al., 2019; Siegel et al., 2020). This ensures that the learned policy stays “close” to the behavior policy. This is achieved by adding a regularization term that calculates the “divergence” between the learned policy and the behavior policy. Kernel Maximum Mean Discrepancy (MMD) (Gretton et al., 2007), Wasserstein distance and KL divergence are widely used (Wu et al., 2019). The regularization term is either fixed (Wu et al., 2019), or tuned via dual gradient descent (Kumar et al., 2019), or applied using a trust region objective (Siegel et al., 2020).

In this paper, we propose improvements to the Behavior Regularized Actor Critic (BRAC) algorithm presented in (Wu et al., 2019) and propose BRAC+. The key idea of behavior regularization is that the learned policy only takes actions that have high probability in the dataset. Under this criteria, we compare KL divergence and Maximum Mean Discrepancy (MMD) for behavior regularization. We show that MMD may erroneously assign low penalty to out-of-distribution actions when the behavior policy is multi-modal, leading to catastrophic failure. Moreover, sample-based estimation of the KL divergence used by current techniques (Wu et al., 2019) is both computationally expensive and prone to high variance. Therefore, to reduce variance, we derive an analytical upper bound on the KL divergence measure as the regularization term in the objective function. Next, we mathematically show that the learned Q values can diverge even using behavior regularized policy update under mild assumptions. This is due to the erroneous generalization of the neural network, which leads to large overestimations of the Q values and performance deterioration of the learned policy. To mitigate this problem, we add a gradient penalty term to the policy evaluation objective. By doing so, the Q values are guaranteed to converge.

Our experiments suggest that BRAC+ outperforms the baseline behavior regularized approaches by 40% ~ 87% and the state-of-the-art approach by 6% on challenging D4RL benchmarks (Fu et al., 2020).

2. Background

Markov Decision Process RL algorithms aim to solve Markov Decision Process (MDP) with unknown dynamics. A Markov decision process (Sutton and Barto, 2018) is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, R, P, \mu \rangle$, where \mathcal{S} is the set of states, \mathcal{A} is the set of actions, $R(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ defines the intermediate reward when the agent transitions from state s to s' by taking action a , $P(s'|s, a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ defines the probability when the agent transitions from state s to s' by taking action a , $\mu : \mathcal{S} \rightarrow [0, 1]$ defines the starting state distribution. The objective of reinforcement learning is to select policy $\pi : \mu \rightarrow P(A)$ to maximize the following objective:

$$J(\pi) = \mathbb{E}_{\substack{s_0 \sim \mu, a_t \sim \pi(\cdot|s_t) \\ s_{t+1} \sim P(\cdot|s_t, a_t)}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \right] \quad (1)$$

The Q function under policy π is defined as $Q^\pi(s_t, a_t) = \mathbb{E}_\pi \sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'})$.

Off-policy Reinforcement Learning Modern deep off-policy reinforcement learning algorithms such as DQN (Mnih et al., 2013), SAC (Haarnoja et al., 2018) and TD3 (Fujimoto et al., 2018a) optimize Equation 1 by learning the Q function using data stored in a replay buffer \mathcal{D} :

$$\min_{\psi} \mathbb{E}_{(s,a,s',r) \sim \mathcal{D}} [(Q_{\psi}^{\pi}(s,a) - (r(s,a) + \gamma \mathbb{E}_{a' \sim \pi_{\theta}} Q_{\psi'}^{\pi}(s',a')))]^2 \quad (\text{Policy Evaluation}) \quad (2)$$

where the Q function and the policy π are approximated by neural networks parameterized by ψ and θ , respectively. ψ' denotes the parameters of the target Q network (Mnih et al., 2013). Then, the policy is trained to maximize the learned Q function:

$$\max_{\theta} \mathbb{E}_{s \sim \mathcal{D}} [Q_{\psi}^{\pi}(s, \pi_{\theta})] \quad (\text{Policy Update}) \quad (3)$$

Although data collected from any policies can be used to perform off-policy learning in Equation 2 and 3, it is crucial that the updated policy keeps interacting with the environment to collect on-policy data so that erroneous generalization of the Q function on unseen states can be corrected (Levine et al., 2020).

Behavior Regularized Actor Critic Behavior Regularized Actor Critic (BRAC) (Wu et al., 2019) solves the offline reinforcement learning problem by augmenting the policy update step in Equation 3 as:

$$\max_{\theta} \mathbb{E}_{s \sim \mathcal{D}} [Q_{\psi}^{\pi}(s, \pi_{\theta})] \quad \text{s. t.} \quad \mathbb{E}_{s \sim \mathcal{D}} [D(\pi_{\theta}(\cdot|s) || \pi_b(\cdot|s))] < \epsilon \quad (\text{BRAC Policy Update}) \quad (4)$$

where D is a distance measurement between the learned policy π_{θ} and the behavior policy π_b . The behavior policy is defined as the solution that maximizes the probability of actions conditioned on states in the dataset:

$$\pi_b(a|s) = \arg \max_{\pi_b} \mathbb{E}_{(s,a) \sim \mathcal{D}} [p(a|s)] \quad (5)$$

Common distance measurements used in previous works (Wu et al., 2019; Kumar et al., 2019) include Maximum Mean Discrepancy (MMD) and KL divergence. BRAC-p (Wu et al., 2019) uses the standard policy evaluation in Equation 2 while BRAC-v (Wu et al., 2019) also augments the policy evaluation. As they show similar empirical results, our method BRAC+ proposes improvements over the simpler version BRAC-p.

3. Problem Statement

Given a fixed dataset $\mathcal{D} = \{(s_i, a_i, s'_i, r_i)\}_{i=1}^N$, the objective of offline reinforcement learning is to learn policy π_{θ} such that Equation 1 is maximized. In principle, any off-policy algorithms can be directly applied. The major challenge is the absence of on-policy data to correct the erroneous generalization of the Q function on unseen states (Levine et al., 2020).

4. Related Work

We briefly summarize prior works in deep offline RL and discuss their relationship with our approach. Please refer to (Levine et al., 2020) for traditional batch RL approaches. As discussed in Section 1, the fundamental challenge in learning from a static data is to avoid out-of-distribution actions (Levine et al., 2020). This requires solving two problems: 1) estimation of the behavior policy, and 2) quantification of the out-of-distribution actions. We follow BCQ (Fujimoto et al., 2018b), BEAR (Kumar et al., 2019) and BRAC (Wu et al., 2019) by learning the behavior policy using a conditional Variational Auto-encoder (VAE) (Kingma and Welling, 2014). To avoid out-of-distribution actions, BCQ generates actions in the target values by perturbing the behavior policy. However, this is over-pessimistic in most of the cases. BRAC (Wu et al., 2019) constrains the policy using various sample-based divergence measurements including Maximum Mean Discrepancy (MMD), Wasserstein distance and KL divergence with penalized policy update (BRAC-p) or policy evaluation (BRAC-v). BEAR (Kumar et al., 2019) is an instance of BRAC with penalized policy improvement using MMD (Gretton et al., 2007). Sample-based estimation is computationally expensive and suffers from high variance. In contrast, our method uses an analytical upper-bound of the KL divergence to constrain the distance between the learned policy and the behavior policy. It is both computationally efficient and has low variance. (Siegel et al., 2020) solves trust-region objective instead of using penalty. For KL regularized policy improvement with fixed temperature, the optimal policy has a closed form solution (Wang et al., 2020). However, tuning the fixed temperature is difficult as it doesn't imply the actual divergence. On the contrary, automatical temperature tuning via dual gradient descent can bound the actual divergence value within a certain threshold. (Fox, 2019) presents a closed-form expression for the regularization coefficient that completely eliminates the bias in entropy-regularized value updates. However, the `softmax` operator introduced by the approach makes it hard to use in continuous action space. CQL (Kumar et al., 2020) avoids estimating the behavior policy by learning a conservative Q function that lower-bounds its true value. Hyperparameter search is another challenging problem in offline RL. (Lee et al., 2020) uses a gradient-based optimization to search the hyperparameter using held-out data. MOPO (Yu et al., 2020) follows MBPO Janner et al. (2019) with additional reward penalty on unreliable model-generated transitions. MBOP (Argenson and Dulac-Arnold, 2020) learns the dynamics model, the behavior policy and a truncated value function to perform online planning. (Kidambi et al., 2020) learns a surrogate MDP using the dataset, such that taking out-of-distribution actions transit to the terminal state. The out-of-distribution actions are detected using the agreement of the predictions from ensembles of the learned dynamics models.

5. Improving Behavior Regularized Offline Reinforcement Learning

In this section, we first quantify the criteria of the out-of-distribution actions and compare various existing approaches to satisfy the criteria. We argue that Kullback–Leibler divergence (KLD) is superior to Maximum Mean Discrepancy (MMD) in meeting the criteria. However, accurately estimation of the KL divergence requires large amounts of samples to reduce the variance of the estimator. To avoid the expensive sampling, we derive an upper bound on the KL divergence between the learned policy and the behavior policy that can

be computed analytically. Lastly, we mathematically show that the difference between the Q values of the learned policy and the behavior policy can be arbitrarily large even if the KL divergence is small. This leads to the learned Q function diverging. To prevent it, we apply the gradient penalty when learning the Q function.

5.1. Quantification of Out-of-distribution Actions

It is critical to define a criteria that accurately distinguishes between policies that generate in-distribution and out-of-distribution actions. The key insight that can be leveraged to define this criteria is that the learned policy should only take actions that have high probability in the dataset \mathcal{D} . We say that an action is in-distribution if $\pi_b(a|s) > \epsilon$. Ideally, the learned policy π_θ should have positive probability for in-distribution actions and zero probability elsewhere. However, this is impossible for continuous policies represented as neural networks. In practice, we say a policy distribution π_θ only contains in-distribution actions if:

$$\mathbb{E}_{a \sim \pi_\theta(\cdot|s)}[\pi_b(a|s)] > \epsilon \quad (6)$$

where ϵ is the pre-defined threshold. For the rest of the paper, we use the terms “a policy distribution contains in/out-of-distribution actions” and “in/out-of-distribution actions” interchangeably. Based on the criteria, we compare two behavior regularization protocols that are used in previous works (Kumar et al., 2019; Wu et al., 2019).

Kernel MMD Kernel Maximum Mean Discrepancy (MMD) (Gretton et al., 2007) was first introduced in (Kumar et al., 2019) to penalize the policy from diverging from the behavior policy:

$$\text{MMD}_k^2(\pi(\cdot|s), \pi_b(\cdot|s)) = \mathbb{E}_{x, x' \sim \pi(\cdot|s)}[K(x, x')] - 2 \mathbb{E}_{\substack{x \sim \pi(\cdot|s), \\ y \sim \pi_b(\cdot|s)}}[K(x, y)] + \mathbb{E}_{y, y' \sim \pi_b(\cdot|s)}[K(y, y')] \quad (7)$$

where K is a kernel function. Symmetric kernel functions such as Laplacian and Gaussian kernels are typically used (Gretton et al., 2007). Kernel MMD can effectively prevent out-of-distribution actions for single-modal datasets (e.g. The dataset is collected by a Gaussian policy) as shown in (Kumar et al., 2019). However, it fails on multi-modal datasets (e.g. The dataset is collected by several distinct Gaussian policies.). This is because actions with low MMD distance may have low probability density. We show an example in Figure 1. In the middle figure, the behavior policy is a mixture of two Gaussian distributions. To minimize the MMD distance between the learned Gaussian policy and the behavior policy, the mean of the Gaussian policy is around zero, which has low density. This causes out-of-distribution actions are preferred to in-distribution actions and the failure of the behavior regularized offline reinforcement learning.

KL divergence For two probability distributions P and Q on some probability space χ , the KL divergence from Q to P is given as

$$\mathcal{D}_{\text{KL}}(P, Q) = \int_{x \sim \chi} P(x) \log \frac{P(x)}{Q(x)} dx \quad (8)$$

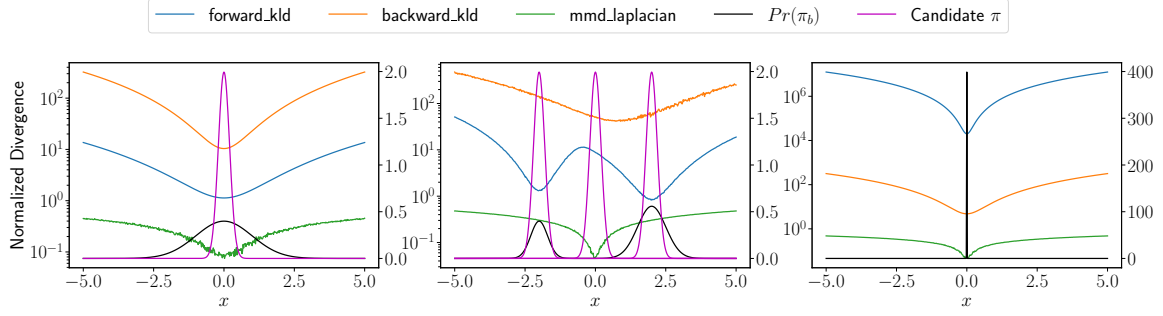


Figure 1: The black curve in the two figures represents the behavior distribution π_b . The blue, orange and green curves in the graph show forward KL, backward KL and MMD with Laplacian kernel between π_b and $\pi_\theta = \mathcal{N}(x, \sigma)$, respectively, where x is a variable between $[-10, 10]$ and σ is fixed. In the left figure, $\pi_b = \mathcal{N}(0, 1)$. In the middle figure, π_b is a mixture of two Gaussian distributions: $\mathcal{N}(-2.0, 0.3)$ and $\mathcal{N}(2.0, 0.5)$. The weight of each component is 0.3 and 0.7, respectively. σ is set to 0.2. In the right figure, $\pi_b = \mathcal{N}(0, 0.001)$. We show the candidate policies in purple for detailed comparison.

By constraining the KL divergence $\mathcal{D}_{\text{KL}}(\pi_\theta(\cdot|s), \pi_b(\cdot|s)) < \epsilon_{\text{KL}}$, criteria 6 is automatically satisfied if we fix the entropy of the learned policy: $\mathcal{H}(\pi_\theta) = \mathcal{H}_0$. Estimating the KL divergence requires i) learning an explicit likelihood model from the dataset; and ii) generating a large number of samples to reduce the variance. We can achieve step i) by applying state-of-the-art deep generative models such as Conditional Normalizing Flow (Kobyzev et al., 2020) or Variational Auto-encoder (Kingma and Welling, 2014). However, evaluating the likelihood of a large number of samples is computationally expensive.

Analytical KL divergence upper bound To reduce the variance and avoid computationally expensive sampling, we derive an upper bound on the KL divergence between the learned policy π_θ and the behavior policy π_b that can be computed analytically. Assume we learn π_b using a Conditional VAE (Kingma and Welling, 2014) with latent variable Z . According to the evidence lower bound (ELBO), we obtain $\log \pi_b(a|s) \geq \mathbb{E}_{z \sim q(z|s,a)}[\log p(a|s, z)] - \mathcal{D}_{\text{KL}}(q(z|s, a)||p(z))$, where $q(z|s, a)$ is the approximated posterior distribution and $p(z)$ is the prior. Then, the KL divergence is bounded by:

$$\begin{aligned}
 \mathcal{D}_{\text{KL}}(\pi_\theta(\cdot|s)||\pi_b(\cdot|s)) &= \mathbb{E}_{a \sim \pi_\theta}[\log \pi_\theta(a|s) - \log \pi_b(a|s)] \\
 &\leq \mathbb{E}_{a \sim \pi_\theta}[\log \pi_\theta(a|s)] - \mathbb{E}_{a \sim \pi_\theta}[\mathbb{E}_{z \sim q(z|s,a)}[\log p(a|s, z)] - \mathcal{D}_{\text{KL}}(q(z|s, a)||p(z))] \\
 &= \mathbb{E}_{a \sim \pi_\theta, z \sim q(z|s,a)}[\mathcal{D}_{\text{KL}}(\pi_\theta(\cdot|s)||p(\cdot|s, z)) + \mathcal{D}_{\text{KL}}(q(z|s, a)||p(z))] \\
 &= \mathcal{D}_{\text{KL}}^{\text{upper}}(\pi_\theta(\cdot|s)||\pi_b(\cdot|s))
 \end{aligned} \tag{9}$$

In our Conditional VAE implementation (Kingma and Welling, 2014), $p(\cdot|s, z)$ is the output distribution of the decoder. $q(z|s, a)$ is the output distribution of the encoder. Both of them are Gaussian distributions with parameterized mean and variance. $p(z) = \mathcal{N}(0, 1)$ is the prior distribution. We parameterize the learned policy as a Gaussian distribution:

$\pi_\theta(a|s) = \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$, where μ_θ and σ_θ represents the output of a neural network. Since there is a closed-form solution of the KL divergence between two Gaussian distributions, Equation 9 can be computed analytically.

Impact of the bias by using the analytical upper bound According to (Kingma and Welling, 2014), the bias term of the upper bound is $\mathcal{D}_{\text{KL}}(q(z|s, a)||p(z|s, a))$, where $p(z|s, a)$ denotes the true posterior distribution and $q(z|s, a)$ denotes the approximated posterior distribution. Using the linearity of the expectation, we obtain:

$$\mathbb{E}_{s \sim \mathcal{D}} [\mathcal{D}_{\text{KL}}(\pi_\theta(\cdot|s)||\pi_b(\cdot|s))] = \mathbb{E}_{s \sim \mathcal{D}} [\mathcal{D}_{\text{KL}}^{\text{upper}}(\pi_\theta(\cdot|s)||\pi_b(\cdot|s))] - \mathbb{E}_{s \sim \mathcal{D}} [\mathcal{D}_{\text{KL}}(q(z|s, a)||p(z|s, a))] \quad (10)$$

For a fixed dataset and the trained conditional VAE, $\mathbb{E}_{s \sim \mathcal{D}} [\mathcal{D}_{\text{KL}}(q(z|s, a)||p(z|s, a))]$ is a fixed value. Thus, we can directly set the threshold of the upper bound without the necessity to estimate the bias because the threshold of the true KL divergence is also a hyperparameter.

Based on the aforementioned arguments, the final policy update step of our improved behavior regularized actor critic is:

$$\max_{\theta} \mathbb{E}_{s \sim \mathcal{D}} [Q_{\psi}^{\pi}(s, \pi_{\theta})] \text{ s. t. } \mathbb{E}_{s \sim \mathcal{D}} [\mathcal{D}_{\text{KL}}^{\text{upper}}(\pi_{\theta}(\cdot|s)||\pi_b(\cdot|s))] < \epsilon_{\text{KL}}^{\text{upper}} \text{ and } \mathcal{H}(\pi_{\theta}) = \mathcal{H}_0 \quad (11)$$

We solve the constrained optimization problem in Equation 11 by using dual gradient descent (Robbins and Monro, 1951).

5.2. Gradient penalized policy evaluation

In this section, we argue that behavior regularization is not sufficient to learn stable policies from offline data due to arbitrarily large overestimation of the Q values. We start by presenting Theorem 1:

Theorem 1 *Let the policy before and after the update in Equation 11 be π_{old} and π_{new} . Assume the Q function generalizes such that π_{old} is never a local maximizer of $\mathbb{E}_{a \sim \pi} [Q^{\pi}(s, a)]$, and after each policy evaluation in Equation 2, we have $\mathbb{E}_{a \sim \pi_b} [Q^{\pi_{\text{new}}}(s, a) - Q^{\pi_{\text{old}}}(s, a)] > \delta$, where $\delta = \sup_a |Q^{\pi_{\text{new}}}(s, a) - Q^{\pi_{\text{old}}}(s, a)| \cdot \sqrt{\epsilon_{\text{KL}}/2}$ and $\epsilon_{\text{KL}} = \mathcal{D}_{\text{KL}}(\pi_{\text{new}}||\pi_b)$. Then, we have $\mathbb{E}_{a \sim \pi_{\text{new}}} [Q^{\pi_{\text{new}}}(s, a)] > \mathbb{E}_{a \sim \pi_{\text{old}}} [Q^{\pi_{\text{old}}}(s, a)]$.*

Proof Please see Appendix A. ■

Theorem 1 states that by iteratively updating the policy and the Q function using Equation 11 and Equation 2, under mild assumptions, the Q value of the current policy will monotonically increase after each iteration. Note that in practice, the assumption typically holds at states with low density. The gradients at those states are so large that even a small learning rate causes the increase of the Q values to be greater than δ . That’s why it’s crucial to penalize the large gradients of the Q function. Otherwise, it leads to infinitely large overestimation of the true Q value and thus a failure in learning a policy. This issue arises from the fact that we fit the Q function using $\mathbb{E}_{(s,a) \sim \mathcal{D}} [Q^{\pi}(s, a)]$. And the Q value of the policy $\mathbb{E}_{(s,a) \sim \pi} [Q^{\pi}(s, a)]$ relies on the generalization of the neural networks. Such generalization may be erroneous and may lead to divergence of the Q function. Note that in standard off-policy reinforcement learning, the erroneous generalization can be corrected due to the new **on-policy** data that is collected by environmental interactions (Levine et al., 2020).

Algorithm 1 BRAC+: Improved Behavior Regularized Actor Critic

- 1: Train the behavior policy π_b on the offline dataset $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^N$ via maximum likelihood estimation: $\pi_b = \arg \max_{\pi_\beta} \sum_{i=1}^N \log \pi_\beta(a_i | s_i)$
 - 2: Train initial policy: $\pi_\theta = \arg \min_{\pi_\theta \in \Pi} \mathcal{D}_{\text{KL}}(\pi, \pi_b)$
 - 3: **for** $e = 1 : E$ **do**
 - 4: **for** $t = 1 : T$ **do**
 - 5: Update Q network using Equation 12
 - 6: Update the policy using Equation 11
 - 7: Update the target network $\psi' = \tau\psi + (1 - \tau)\psi'$
 - 8: **end for**
 - 9: **end for**
-

Gradient penalized policy evaluation To fix this issue, we need to control how the learned Q function generalizes. CQL (Kumar et al., 2020) controls it by setting the maximum difference between the Q value of the learned policy and the behavior policy. In this work, we propose to penalize the gradient of the Q function, such that **the gradient of the Q function decreases as the distance between the learned policy and the behavior policy increases**. By doing so, there always exists a local maximizer of $\mathbb{E}_{a \sim \pi}[Q^\pi(s, a)]$. This guarantees the convergence of the Q function. Mathematically, we augment Equation 2 as:

$$\min_{\psi} \mathbb{E}_{(s, a, s', r) \sim \mathcal{D}} [(Q_{\psi}^{\pi}(s, a) - (r(s, a) + \gamma \mathbb{E}_{a' \sim \pi_{\theta}} Q_{\psi'}^{\pi}(s', a')))]^2 + \lambda \mathbb{E}_{a'' \sim \pi(\cdot|s)} (\|\nabla_{a''} Q_{\psi}^{\pi}(s, a'')\|_2 f(\mathcal{D}_{\text{KL}}(\pi || \pi_b))) \quad (12)$$

where π is the learned policy and f is a positive and monotonically increasing function. In practice, we find **softplus** function works very well. We tune the weight λ using dual gradient descent.

Tradeoff analysis Note that although gradient penalty prevents the Q function from divergence, it may be too conservative such that correct generalization is discarded. This may lead to worse performance compared with not using gradient penalty as shown in Section 6 for the hopper-medium-replay task.

6. Experiments

Our experiments¹ aim to answer the following questions:

1. How does the performance of our methods compare with baseline behavior regularized approaches and the state-of-the-art model-free and model-based offline RL methods? (Section 6.1)
2. How does the use of analytical variational upper bound on KL divergence for regularization term compare with Maximum Mean Discrepancy (MMD)? (Section 6.2)
3. Does the Q value diverge in real-world datasets if not using gradient penalty? Does the gradient penalized policy evaluation empirically guarantee the convergence of the Q function during the training? (Section 6.2)

1. Our implementation is available at <https://github.com/vermouth1992/bracp>.

To answer these questions, we evaluate our methods on a subset of the D4RL (Fu et al., 2020) benchmark. We consider three locomotion tasks (hopper, walker2d, and halfcheetah) and four types of datasets: 1) random (rand): collect the interactions of a run of random policy for 1M steps to create the dataset, 2) medium (med): collect the interactions of a run of medium quality policy for 1M steps as the dataset, 3) medium-expert (med-exp): run a medium quality policy and an expert quality policy for 1M steps, respectively, and combine their interactions to create the dataset, 4) mixed (medium-replay): train a policy using SAC (Haarnoja et al., 2018) until the performance of the learned policy exceeds a pre-determined threshold, and take the replay buffer as the dataset.

We compare against state-of-the-art model-free and model-based baselines, including behavior cloning, BEAR (Kumar et al., 2019) that constrains the learned policy within the support of the behavior policy using sampled MMD, offline SAC (Haarnoja et al., 2018), BRAC-p/v (Wu et al., 2019) that constrains the learned policy within the support of the behavior policy using various sample-based f -divergences to penalize either the policy improvement (p) or the policy evaluation (v), CQL(\mathcal{H}) (Kumar et al., 2020) that learns a Q function that lower-bounds its true value. We also compare against model-based approaches including MOPO (Yu et al., 2020) that follows MBPO (Janner et al., 2019) with additional reward penalties and MBOP (Argenson and Dulac-Arnold, 2020) that learns an offline model to perform online planning.

6.1. Comparative Results

Table 1: Results for OpenAI gym (Brockman et al., 2016) environments in the D4RL (Fu et al., 2020) datasets. For each task, we train for 1 million gradient steps and report the performance by running the policy obtained at the last epoch of the training for 100 episodes, averaged over 3 random seeds with standard deviation. Each number is the normalized score as proposed in (Fu et al., 2020). Please refer to (Fu et al., 2020) for results on more baselines.

Task Name	Model-Free						Model-Based	
	BC	SAC-offline	BEAR	BRAC-p/v	CQL(\mathcal{H})	BRAC+ (Ours)	MOPO	MBOP
halfcheetah-rand	2.1	30.5	25.1	24.1/31.2	35.4	29.7 \pm 1.4	35.4\pm2.5	6.3 \pm 4.0
walker2d-rand	1.6	4.1	7.3	-0.2/1.9	7.0	2.3 \pm 0.0	13.6\pm2.6	8.1 \pm 5.5
hopper-rand	9.8	11.3	11.4	11.0/ 12.2	10.8	12.2\pm0.1	11.7 \pm 0.4	10.8 \pm 0.3
halfcheetah-med	36.1	-4.3	41.7	43.8/46.3	44.4	48.2 \pm 0.3	42.3 \pm 1.6	44.6 \pm 0.8
walker2d-med	6.6	0.9	59.1	77.5/ 81.1	79.2	77.6 \pm 0.8	17.8 \pm 19.3	41.0 \pm 29.4
hopper-med	29.0	0.8	52.1	32.7/31.1	58.0	39.1 \pm 3.5	28.0 \pm 12.4	48.8 \pm 26.8
Average (single-modal)	14.2	7.2	32.8	31.5/34.0	39.1	34.9	24.8	26.6
halfcheetah-med-exp	35.8	1.8	53.4	44.2/41.9	62.4	79.2\pm6.2	63.3 \pm 38.0	105.9\pm 17.8
walker2d-med-exp	6.4	-0.1	40.1	76.9/81.6	98.7	102.4\pm3.3	44.6 \pm 12.9	70.2 \pm 36.2
hopper-med-exp	111.9	1.6	96.3	1.9/0.8	111.0	111.2 \pm 1.9	23.7 \pm 6.0	55.1 \pm 44.3
halfcheetah-mixed	38.4	-2.4	38.6	45.4/ 47.7	46.2	46.2 \pm 1.3	53.1\pm2.0	42.3 \pm 0.9
walker2d-mixed	11.3	1.9	19.2	-0.3/0.9	26.7	46.1 \pm 1.2	39.0\pm9.6	9.7 \pm 5.3
hopper-mixed	11.8	3.5	33.7	0.6/0.6	48.6	72.6\pm22.2	67.5\pm24.7	12.4 \pm 5.8
Average (multi-modal)	35.9	1.1	46.9	28.1/28.9	65.6	76.3	48.5	49.3
Average (overall)	25.1	4.1	39.8	29.8/31.4	52.4	55.6	36.7	37.9

Performance on multi-modal datasets We first compare the performance on multi-modal datasets i.e. med-exp and mixed datasets. Results shown in Table 1 suggest that

our method outperforms various model-free baselines on most of the multi-modal datasets, especially on hopper-mix and walker2d-mix by up to 1.5x. Compared with BEAR (Kumar et al., 2019), our method improves the performance by 63% due to the advantage of the KL divergence over the kernel MMD as discussed in Section 5.1. Compare with BRAC (Wu et al., 2019), our method improves the performance by 164% due to the advantage of the analytical KL divergence upper bound. Compare with the state-of-the-art approach (Kumar et al., 2020), our method improves the performance by 16.3%.

Performance on single-modal datasets The performance of our method on single-modal (rand and med) dataset outperforms the baseline methods (Kumar et al., 2019; Wu et al., 2019) as evident from Table 1 by 3% \sim 6%. Our method is worse than the state-of-the-art method (Kumar et al., 2020) particularly on datasets collected by random policies. This is because randomly gathered dataset typically contains several discontinuous modalities, where KL divergence based behavior regularization may stuck at a local optimum.

6.2. Ablation Study

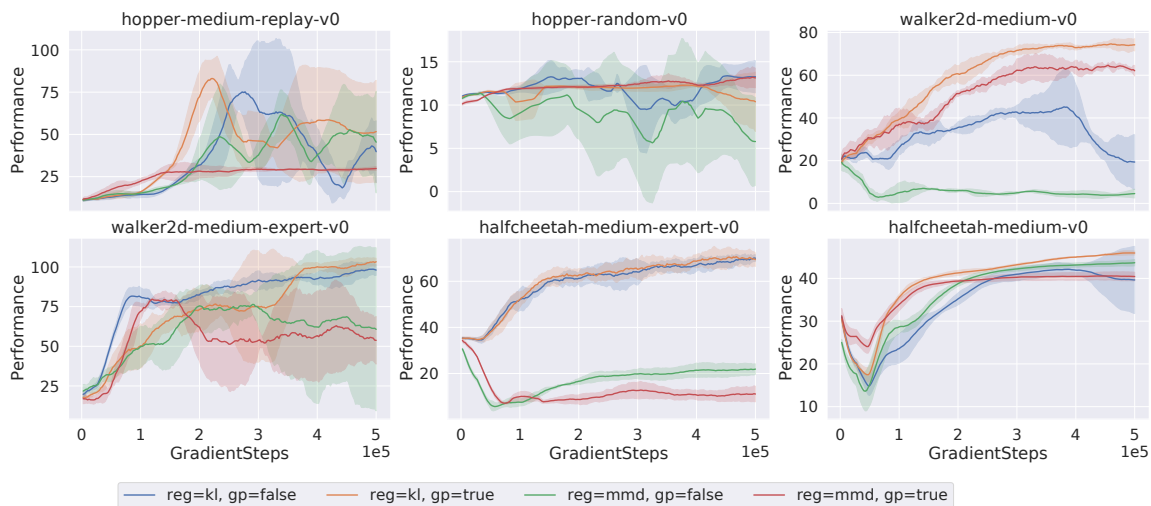


Figure 2: Learning curve of various tasks using KL divergence and MMD as behavior regularization protocol, and with and without using gradient penalized policy evaluation. Each setting is repeated for 3 random seeds. The curve is the mean and the shaded area is the standard deviation. The curves are smoothed by a factor of 20. The hyperparameters can be found in Appendix B.

To answer question (2) and (3), we conduct a thorough ablation study on BRAC+ on 6 tasks with different data collection policies (hopper-medium-replay (mixed), hopper-random, walker2d-medium, walker2d-medium-expert, halfcheetah-medium and halfcheetah-medium-expert).

Maximum Mean Discrepancy vs. KL divergence The results of using MMD versus KL divergence are shown in Figure 2. In general, using KL divergence as the behavior

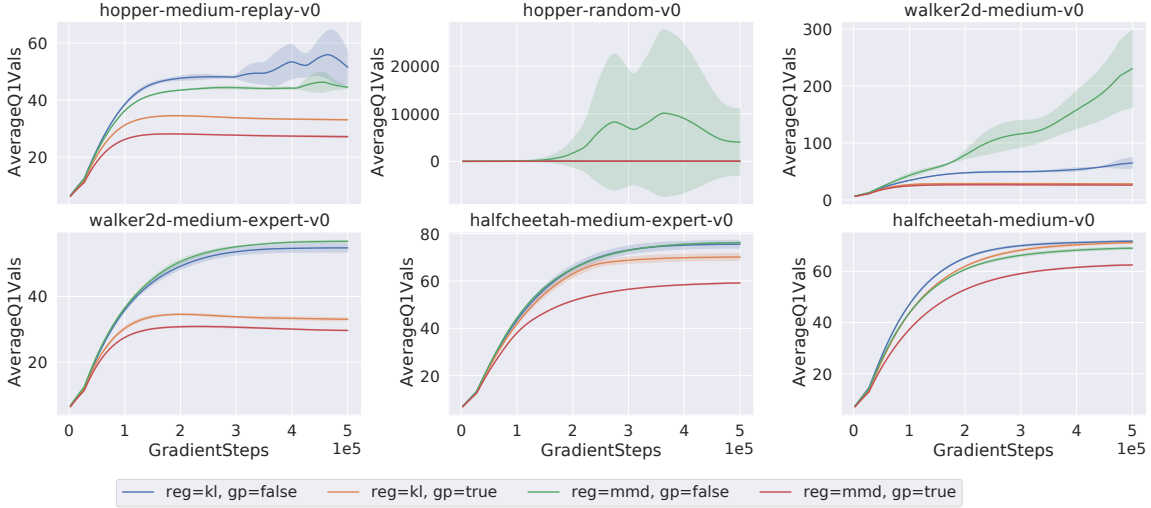


Figure 3: The average Q values of all the states in the dataset during training. The setting is the same as in Figure 2.

regularization protocol provides better performance than using MMD. The performance is similar for hopper-medium-replay and hopper-random. The performance by using KL divergence is slightly better for walker2d-medium and halfcheetah-medium. The performance discrepancy is huge for halfcheetah-medium-expert and walker2d-medium-expert task and using MMD fails to learn better policies than the behavior cloning. This is because the medium-expert tasks contain multi-modal behavior policies as depicted in Figure 1, where the MMD assigns low penalty to out-of-distribution actions.

Gradient penalty vs. no gradient penalty From Figure 2, we observe that the performance by using gradient penalty is better than not using gradient penalty except the hopper-medium-replay task with MMD as divergence. This is because gradient penalty prevents correct out-of-distribution generalization in this task. For other tasks, it is clear that without gradient penalty, the performance starts to decrease after some gradient steps due to the erroneous overestimation of the out-of-distribution actions. As shown in Figure 3, the gradient penalized policy evaluation provides more conservative Q values compared with no gradient penalty. It is also worth noticing that the Q values of hopper-medium-replay, hopper-random and walker2d-medium diverges when not using gradient penalty. This verifies our arguments in Section 5.2.

7. Discussions and Limitations

We conjecture that the behavior-regularized approach with gradient penalty is not sufficient to tackle offline RL problems since it fully ignores the state distribution. To see this, we can create a dataset that only adds a few trajectories from an expert policy to a dataset collected by a low-quality policy. If the low-quality policy doesn't visit the "good" states in the expert policy (can't combine sub-optimal policies), behavior-regularized approach leads to a policy that imitates the expert policy. Such imitation is likely to fail due to compounding errors

(Ross et al., 2010). The right approach for this dataset is to completely ignore expert trajectories and combine sub-optimal policies in the low-quality regions. To achieve this, we need to consider the state distribution as the density of the “good” states is very low.

8. Conclusion

In this paper, we improved the behavior regularized offline reinforcement learning by proposing a low-variance upper bound of the KL divergence estimator to reduce variance and gradient penalized policy evaluation such that the learned Q functions are guaranteed to converge. Our experimental results on challenging benchmarks illustrate the benefits of our improvements.

Acknowledgments

This work is supported by U.S. National Science Foundation (NSF) under award number 2009057 and U.S. Army Research Office (ARO) under award number W911NF1910362. We would like to thank Chen-Yu Wei for his help in the mathematical derivation.

References

- Abien Fred Agarap. Deep learning using rectified linear units (relu). *ArXiv*, abs/1803.08375, 2018.
- Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. *ArXiv*, abs/2008.05556, 2020.
- Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.
- Sungwoon Choi, Heonseok Ha, Uiwon Hwang, Chanju Kim, Jung-Woo Ha, and S. Yoon. Reinforcement learning based recommender system using biclustering technique. *ArXiv*, abs/1801.05532, 2018.
- R. Fox. Toward provably unbiased temporal-difference value estimation. 2019.
- Justin Fu, Aviral Kumar, Ofir Nachum, G. Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *ArXiv*, abs/2004.07219, 2020.
- Scott Fujimoto, H. V. Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *ArXiv*, abs/1802.09477, 2018a.
- Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *CoRR*, abs/1812.02900, 2018b. URL <http://arxiv.org/abs/1812.02900>.
- Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Scholkopf, and Alexander J. Smola. A kernel approach to comparing distributions. In *AAAI*, pages 1637–1641, 2007. URL <http://www.aaai.org/Library/AAAI/2007/aaai07-262.php>.

- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018. URL <http://arxiv.org/abs/1801.01290>.
- Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *CoRR*, abs/1906.08253, 2019. URL <http://arxiv.org/abs/1906.08253>.
- Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel : Model-based offline reinforcement learning, 2020.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.
- Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.
- I. Kobyzev, S. Prince, and M. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 2020.
- Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *CoRR*, abs/1906.00949, 2019. URL <http://arxiv.org/abs/1906.00949>.
- Aviral Kumar, Aurick Zhou, G. Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *ArXiv*, abs/2006.04779, 2020.
- Byung-Jun Lee, Jongmin Lee, Peter Vrancx, DongHo Kim, and Kee-Eung Kim. Batch reinforcement learning with hyperparameter gradients. 2020.
- Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643, 2020.
- Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2016.
- Long-Ji Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, USA, 1992.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL <http://arxiv.org/abs/1312.5602>.
- Mark D. Reid and Robert C. Williamson. Generalised pinsker inequalities. *CoRR*, abs/0906.1244, 2009. URL <http://arxiv.org/abs/0906.1244>.
- H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

- Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686, 2010. URL <http://arxiv.org/abs/1011.0686>.
- Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, T. Lampe, Roland Hafner, and Martin A. Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *ArXiv*, abs/2002.08396, 2020.
- David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016. URL <http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html>.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.
- Ziyu Wang, A. Novikov, Konrad Zolna, Jost Tobias Springenberg, Scott Reed, B. Shahriari, N. Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, and N. D. Freitas. Critic regularized regression. *ArXiv*, abs/2006.15134, 2020.
- Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning, 2019.
- Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization, 2020.
- Chi Zhang, Sanmukh R. Kuppannagari, Rajgopal Kannan, and Viktor K. Prasanna. Building hvac scheduling using reinforcement learning via neural network based model approximation. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, BuildSys ’19, pages 287–296, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450370059. doi: 10.1145/3360322.3360861. URL <https://doi.org/10.1145/3360322.3360861>.

Appendix A. Proofs

A.1. Proof of Theorem 1

Proof Let the current policy be π_{old} and the Q function be $Q^{\pi_{\text{old}}}$. Since π_{old} is never a local maximizer of $\mathbb{E}_{a \sim \pi} [Q^{\pi}(s, a)]$, update each policy update step defined in Equation 4, we obtain:

$$\mathbb{E}_{a \sim \pi_{\text{new}}} [Q^{\pi_{\text{old}}}(s, a)] > \mathbb{E}_{a \sim \pi_{\text{old}}} [Q^{\pi_{\text{old}}}(s, a)] \quad (13)$$

Let $\Delta Q(s, a) = Q^{\pi_{\text{new}}}(s, a) - Q^{\pi_{\text{old}}}(s, a)$. According to Pinsker’s inequality (Reid and Williamson, 2009), we obtain:

$$|\mathbb{E}_{a \sim \pi_{\text{new}}}[\Delta Q(s, a)] - \mathbb{E}_{a \sim \pi_b}[\Delta Q(s, a)]| \leq \sup_a |\Delta Q(s, a)| \sqrt{\frac{\mathcal{D}_{\text{KL}}(\pi_{\text{new}} \parallel \pi_b)}{2}} \quad (14)$$

According to the assumption, we have $\mathbb{E}_{a \sim \pi_b}[\Delta Q(s, a)] > \sup_a |\Delta Q(s, a)| \sqrt{\frac{\mathcal{D}_{\text{KL}}(\pi_{\text{new}} \parallel \pi_b)}{2}}$. Thus, we obtain:

$$\mathbb{E}_{a \sim \pi_{\text{new}}}[\Delta Q(s, a)] > 0 \quad (15)$$

Combining Equation 13 and Equation 15, we obtain:

$$\begin{aligned} & \mathbb{E}_{a \sim \pi_{\text{new}}}[Q^{\pi_{\text{new}}}(s, a)] - \mathbb{E}_{a \sim \pi_{\text{old}}}[Q^{\pi_{\text{old}}}(s, a)] \\ &= \mathbb{E}_{a \sim \pi_{\text{new}}}[Q^{\pi_{\text{new}}}(s, a)] - \mathbb{E}_{a \sim \pi_{\text{old}}}[Q^{\pi_{\text{old}}}(s, a)] + \mathbb{E}_{a \sim \pi_{\text{new}}}[Q^{\pi_{\text{old}}}(s, a)] - \mathbb{E}_{a \sim \pi_{\text{new}}}[Q^{\pi_{\text{old}}}(s, a)] \\ &= (\mathbb{E}_{a \sim \pi_{\text{new}}}[Q^{\pi_{\text{old}}}(s, a)] - \mathbb{E}_{a \sim \pi_{\text{old}}}[Q^{\pi_{\text{old}}}(s, a)]) + (\mathbb{E}_{a \sim \pi_{\text{new}}}[Q^{\pi_{\text{new}}}(s, a)] - \mathbb{E}_{a \sim \pi_{\text{new}}}[Q^{\pi_{\text{old}}}(s, a)]) \\ &> \mathbb{E}_{a \sim \pi_{\text{new}}}[Q^{\pi_{\text{new}}}(s, a) - Q^{\pi_{\text{old}}}(s, a)] = \mathbb{E}_{a \sim \pi_{\text{new}}}[\Delta Q(s, a)] > 0 \end{aligned} \quad (16)$$

Thus, $\mathbb{E}_{a \sim \pi_{\text{new}}}[Q^{\pi_{\text{new}}}(s, a)] > \mathbb{E}_{a \sim \pi_{\text{old}}}[Q^{\pi_{\text{old}}}(s, a)]$. ■

Appendix B. Implementation Details

Reward scaling Any affine transformation of the reward function does not change the optimal policy of the MDP. In our experiments, we rescale the reward to $[0, 1]$ as:

$$r' = (r - r_{\min}) / (r_{\max} - r_{\min}) \quad (17)$$

where r_{\max} and r_{\min} is the maximum and the minimum reward in the dataset.

Initialization If the dataset is collected using a narrow policy distribution in a high dimensional space (e.g. human demonstration), the constrained optimization problem using dual gradient descent finds it difficult to converge if random initialization is used for the policy network. To mitigate this issue, we start with a policy that has the minimum KL divergence with the behavior policy: $\pi_{\theta} = \arg \min_{\pi_{\theta} \in \Pi} \mathcal{D}_{\text{KL}}(\pi, \pi_{\beta})$, where Π represents a family of policy types. In this work, we consider Π as Gaussian policies. Correspondingly, we initialize the Q network to $Q^{\pi_{\theta}}$.

Policy network Our policy network is a 3-layer feed-forward neural network. The size of each hidden layer is 512. We apply RELU activation (Agarap, 2018) after each hidden layer. Following (Haarnoja et al., 2018), the output is a Gaussian distribution with diagonal covariance matrix. We apply \tanh to enforce the action bounds. The log-likelihood after applying the \tanh function has a simple closed form solution. We refer to (Haarnoja et al., 2018) Appendix C for more details.

Q network Following (Haarnoja et al., 2018; Fujimoto et al., 2018a; Wu et al., 2019), we train two independent Q network $\{Q_{\psi_1}, Q_{\psi_2}\}$ to penalize uncertainty over the future states. We maintain a target Q network $\{Q_{\psi'_1}, Q_{\psi'_2}\}$ with the same architecture and update the target weights using a weighted sum of the current Q network and the target Q network. When computing the target Q values, we simply take the minimum value of the two Q networks:

$$Q_{\psi'}(s', a') = \min_{j=1,2} Q_{\psi'_j}(s', a') \quad (18)$$

Each Q network is a 3-layer feed-forward neural network. The size of each hidden layer is 256. We apply RELU activation (Agarap, 2018) after each hidden layer.

Behavior policy network Following the previous work (Fujimoto et al., 2018b; Kumar et al., 2019), we learn a conditional variational auto-encoder (Kingma and Welling, 2014) as our behavior policy network. The encoder takes a pair of states and actions, and outputs a Gaussian latent variable Z . The decoder takes sampled latent code z and states, and outputs a mixture of Gaussian distributions. Both the architecture of the encoder and the decoder is a 3-layer feed-forward neural network. The size of each hidden layer is 512. The activation is relu (Agarap, 2018). To avoid epistemic uncertainty, we train B ensembles of behavior policy networks. At test time, we randomly select one model to perform the calculations. We found $B = 3$ is sufficient for all the experiments. We pre-train the behavior policy network for 400k gradient steps.

Hyperparameter selection The hyperparameters used in our method include the target entropy of the policy \mathcal{H}_0 and the divergence threshold ϵ . If the target policy entropy is larger than the behavior policy entropy, the OOD actions are naturally included. In our implementation, we set the $\mathcal{H}_0 = \mathcal{H}(\pi_b)/4$ for all the Gym tasks Fu et al. (2020). To set ϵ , we first find $\epsilon_{\min} = \min D(\pi_\theta, \pi_b)$. Then, $\epsilon = \epsilon_{\min} + \epsilon_{\text{generalization}}$, where $\epsilon_{\text{generalization}}$ indicates the OOD generalization for this dataset. For all the mixed and random datasets, we set $\epsilon_{\text{generalization}} = 3.0$. For the medium-expert datasets and hopper-medium, we set $\epsilon_{\text{generalization}} = 0.2$. We set $\epsilon_{\text{generalization}} = 1.0$ for walker2d-medium dataset and $\epsilon_{\text{generalization}} = 7.0$ for halfcheetah-medium dataset. The default hyperparameters are shown in Table 2.

Table 2: Default hyper-parameters

Hyper-parameter	Value
Optimizer	Adam (Kingma and Ba, 2015)
Policy learning rate	5e-6
Q network learning rate	3e-4
batch size	100
Target update rate τ	1e-3
Discount factor γ	0.99
Steps per epoch T	2000
Number of epochs	500