Offline Reinforcement Learning with Fisher Divergence Critic Regularization

Ilya Kostrikov [1 2] **Jonathan Tompson** [2] **Rob Fergus** [1 3] **Ofir Nachum** [2]

Abstract

Many modern approaches to offline Reinforcement Learning (RL) utilize *behavior regularization*, typically augmenting a model-free actor critic algorithm with a penalty measuring divergence of the policy from the offline data. In this work, we propose an alternative approach to encouraging the learned policy to stay close to the data, namely parameterizing the critic as the log-behavior-policy, which generated the offline data, plus a state-action value offset term, which can be learned using a neural network. Behavior regularization then corresponds to an appropriate regularizer on the offset term. We propose using a gradient penalty regularizer for the offset term and demonstrate its equivalence to Fisher divergence regularization, suggesting connections to the score matching and generative energy-based model literature. We thus term our resulting algorithm Fisher-BRC (Behavior Regularized Critic). On standard offline RL benchmarks, Fisher-BRC achieves both improved performance and faster convergence over existing state-of-the-art methods. [1]

1. Introduction

Reinforcement learning (RL) describes the field of machine learning concerned with learning a policy to solve a task through stochastic trial-and-error experience in an environment. The default setting typically assumed in RL is of *online* access to the environment; i.e., the learned policy may collect new trial-and-error experience directly from the environment. However, in many practical scenarios, where deploying a new policy to interact with the live environment is expensive or associated with risks or safety concerns (Thomas, 2015), it is more common to have only *offline* access to the environment. That is, the trial-and-error experience available for learning a task-solving policy is a static, offline dataset of experience collected by some other *behavior* policy. This setting is known as offline RL and has attracted a significant amount of interest in recent years (Levine et al., 2020; Lange et al., 2012).

Many of the recent approaches to offline RL utilize some form of *behavior regularization*, in which the learned policy is compelled to stay close to the data-generating behavior policy (Wu et al., 2019). For example, in model-free actor critic algorithms, behavior regularization is typically done by augmenting the actor loss with a penalty measuring the divergence of the learned policy from the behavior policy (Jaques et al., 2019; Kumar et al., 2019; Wu et al., 2019), reminiscent of KL-control methods in related literature (Kappen et al., 2012; Jaques et al., 2017). While straightforward, a disadvantage of this approach is that it does little to regularize the critic itself, and thus, it is common for the critic to take on wildly extrapolated values on actions unseen in the training data, which may dominate any behavior regularization applied to the actor, as we demonstrate in this work.

In this work, we propose an alternative approach to encouraging the learned policy to stay close to the offline data. Focusing on the critic, we propose parameterizing the critic values as the logits of the behavior policy plus an additional offset term. When the actor (the learned policy) is then trained to choose actions which maximize the critic value, it will be compelled to stay close to the behavior policy as long as the offset term is suitably 'small'. Thus, behavior regularization in this setting corresponds to augmenting the standard Bellman error critic loss with an appropriate regularization on the offset term.

What should this regularization term be? After noting that, in continuous control, the offset term's effect on the learned policy is via the gradients of the offset with respect to actions, we propose regularizing the offset term with a gradient penalty. While this may appear heuristic at first, we present mathematical derivations establishing a connection between this gradient penalty and the *Fisher divergence*, which appears in the score matching and energy-based generative

[1]New York University, USA [2]Google Research, USA [3]Google DeepMind, USA. Correspondence to: Ilya Kostrikov <ikostrkov@gmail.com>.

[1]Code to reproduce our results is available at https://github.com/google-research/google-research/tree/master/fisher_brc.

model literature (Lyu, 2012; Bao et al., 2020), interpreting the critic values as the energy function of a Boltzmann distribution.

We thus term our newly proposed actor critic algorithm *Fisher-BRC* (behavior regularized critic). To aid conceptual understanding of Fisher-BRC, we analyze its training dynamics in a simple toy setting, highlighting the advantage of its implicit Fisher divergence regularization as opposed to the more explicit divergence penalties imposed by alternative offline RL methods. We then present an extensive evaluation of Fisher-BRC on standard offline RL benchmarks. We find that Fisher-BRC yields state-of-the-art performance compared to a variety of existing model-free and model-based RL methods. We further show that while Fisher-BRC learns better policies, it is also much more computationally efficient than more sophisticated offline RL algorithms. Overall, Fisher-BRC presents a new approach to behavior regularization in offline RL with compelling practical benefits.

2. Related Work

Our work adds to the rich literature on behavior regularization methods in offline RL (Wu et al., 2019), which propose a number of regularizations in RL training that compel the learned policy to stay close to the offline data. These regularizers have appeared as divergence penalties (Kumar et al., 2019; Jaques et al., 2019; Wu et al., 2019), implicitly through appropriate network initializations (Matsushima et al., 2020), or more explicitly through careful parameterization of the policy (Fujimoto et al., 2019). Another way to apply behavior regularizers is via modification of the critic learning objective as in (Nachum et al., 2019; Kumar et al., 2020). Our work is unique in applying behavior regularization through a parameterization of the critic, and empirically, we find this to yield much better performance.

In this work, we demonstrate that the specific parameterization we employ has connections to Fisher divergence regularization, establishing connections with energy-based models. Prior methods have proposed using energy-based models for policies to express more multi-modal distributions (Haarnoja et al., 2017; Heess et al., 2013). Meanwhile, while our work focuses on reinforcement learning – i.e., learning a return-maximizing policy – other works have established connections between energy-based models and inverse RL or imitation learning (Finn et al., 2016).

In practice, our regularization reduces to a gradient penalty applied to the offset term in the critic. Gradient penalties have appeared in previous work, arguably first popularized in machine learning by Wasserstein generative-adversarial networks (Arjovsky et al., 2017), but also used in imitation learning (Kostrikov et al., 2018), cross-domain disentanglement (Gonzalez-Garcia et al., 2018), and uncertainty estimation (van Amersfoort et al., 2020).

3. Background

We introduce the notation and assumptions used in this paper, as well as provide an in-depth review of the methods most closely related to ours.

3.1. Reinforcement Learning

In this work, we consider environments that can be represented as a Markov Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, p_0, p, r, \gamma)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $p_0(s)$ is a distribution of initial states, $p(s'|s,a)$ is a stochastic dynamics model, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function and $\gamma \in [0, 1)$ is a discount. We restrict our work to continuous action spaces; i.e., $\mathcal{A} \subset \mathbb{R}^d$ for some d. The goal of reinforcement learning is to find a policy $\pi(a|s)$ that maximizes the cumulative discounted returns $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 \sim p_0(\cdot), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)]$. In online reinforcement learning, it is usually assumed that an agent learns based on experience generated by the agent interacting with the learning environment.

3.2. Offline Reinforcement Learning

In this work, we focus on the offline setting, in which the agent cannot generate new experience data and learns based on a provided dataset $\mathcal{D}$ of (s, a, r, s') tuples generated by some other policy interacting with the environment. We call the policy that generated this dataset the *behavior* policy and denote it as μ.

Offline datasets usually do not provide complete state-action coverage. That is, the set $\{(s, a) \mid (s, a, r, s') \in \mathcal{D}\}$ is typically a small subset of the full space $\mathcal{S} \times \mathcal{A}$. Standard reinforcement learning methods such as SAC (Haarnoja et al., 2019) or DDPG (Lillicrap et al., 2019) suffer when applied to these datasets due to extrapolation errors (Fujimoto et al., 2019; Kumar et al., 2019).

Behavior regularization is a prominent approach to offline reinforcement learning that aims to address this problem by using appropriate regularizers to compel the learned policy to stay close to the data. There are two common ways to incorporate behavior regularization into the actor-critic framework – via policy regularization or via a critic penalty. Since our own approach is related to both techniques, in the following sections we describe the two approaches as well as problems associated with them.

3.3. Policy Regularization

Policy regularization can be imposed either during critic or policy learning. First, we describe a family of approaches based on applying behavior constraints to training policies. These constraints can be applied in a hard fashion, by restricting the policy action space to the actions seen in the offline dataset as in BCQ (Fujimoto et al., 2019):

$$\pi(s) := \underset{a_i+\xi_\phi(s,a_i,\Phi)}{\arg\max} Q_\theta(s, a_i + \xi_\phi(s, a_i, \Phi)), \quad \{a_i \sim \mu(\cdot|s)\}_{i=1}^n$$

where $\pi(s)$ is a deterministic policy, $\xi_\phi(s, a, \Phi)$ is a perturbation model on actions constrained to the interval $[-\Phi, \Phi]$, $\mu(a|s)$ is a behavioral policy contracted by fitting a density model to the offline dataset. The hyperparameter n controls the number of sampled actions. In other words, one samples n perturbed actions from the behavior policy and chooses from the action with the largest critic-approximated value. One of the limitations of this approach is that a large number of sampled actions might be required for competitive performance (Ghasemipour et al., 2021).

Using a divergence penalty is an alternative approach to policy regularization. Instead of having hard constraints on the training policy, one can regularize the policy with an appropriately chosen probability divergence such as the KL-divergence (Wu et al., 2019; Jaques et al., 2019):

$$\max_\pi \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)} [Q_\theta(s, a)] - \alpha D_{KL}(\pi(\cdot|s) \| \mu(\cdot|s)) \right]. \quad (1)$$

Although these approaches demonstrate impressive performance on some tasks, they share a common problem. The Q-function, Q_θ, learned via standard TD-error minimization on $\mathcal{D}$ receives no learning signal for actions not observed in the replay buffer, while this same Q-function is nevertheless queried on out-of-distribution actions during the policy update – i.e., when $Q_\theta(s, a)$ is evaluated on $a \sim \pi(\cdot|s)$ in eq. (1) – and for bootstrapping critic targets in the squared TD-loss:

$$J(Q_\theta) := \mathop{\mathbb{E}}_{\substack{(s,a,s')\sim\mathcal{D} \\ a'\sim\pi(\cdot|s')}} [(r(s, a) + \gamma Q_{\hat{\theta}}(s', a') - Q_\theta(s, a))^2]. \quad (2)$$

Thus, issues with critic extrapolation can still dominate divergence regularizers applied to the policy.

3.4. Critic Penalty

Other works, such as AlgaeDICE (Nachum et al., 2019) and CQL (Kumar et al., 2020) attempt to incorporate some divergence regularization into the critic. In particular, AlgaeDICE introduces a term that pushes Q-values down for actions sampled from the training policy while minimizing TD-error via residual learning:

$$\min_\theta \alpha(1-\gamma) \mathop{\mathbb{E}}_{\substack{s_0\sim\pi_0(\cdot) \\ a_0\sim\pi(s_0)}} [Q_\theta(s_0, a_0)] + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}} [(r(s, a) + \gamma \mathop{\mathbb{E}}_{\substack{s'\sim p(\cdot|s,a) \\ a'\sim\pi(\cdot|s')}} [Q_\theta(s', a')] - Q_\theta(s, a))^2].$$

This formulation can be generalized by replacing the squared function by some convex function f. The choice of f may be shown to correspond to an implicit f-divergence regularization on the learned policy with respect to state-action distributions. For example, a choice of $f(x) = \exp(x - 1)$ or $f(x) = \log \mathbb{E}_\mathcal{D} \exp(x)$ corresponds to an implicit KL-divergence.

Based on a similar idea, CQL (Kumar et al., 2020) extends the standard critic loss $J(Q_\theta)$ in eq. (2) with additional terms that minimize Q-values sampled from a policy and maximize values of the dataset actions:

$$\min_\theta J(Q_\theta) + \lambda \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}} [\log \Sigma_a \exp(Q_\theta(s, a)) - Q_\theta(s, a)]. \quad (3)$$

Although both of these methods provide learning signal to the critic Q-values on the entire action space, AlgaeDICE is based on residual learning with slower convergence than fitted TD-learning (Baird, 1995). On the other hand, although CQL can achieve better empirical performance, the log-sum-exp term that appears in its formulation is not tractable for continuous actions and must be computed via numerical integration. The authors of CQL propose doing this via Monte-Carlo sampling with importance weights, where the current training policy is used to draw samples for the procedure. This process can add a significant computational burden to actor critic training. In contrast, our proposed Fisher-BRC will be shown to achieve favorable empirical performance while maintaining computational efficiency closer to standard actor critic.

4. Fisher-BRC

We now continue to describe our own approach to behavior regularization in offline RL settings, which aims to circumvent the issues associated with competing methods described in the previous section, namely (1) lack of well-defined critic values on out-of-distribution actions and (2) computational inefficiency of critic penalty approaches. We begin with a conceptual derivation of our method, before presenting a more formal connection to Fisher divergence regularization. See Algorithm 1 for a sketch of the full algorithm.

Algorithm 1 Fisher-BRC [Sketch].

Input: Dataset $\mathcal{D}$, offset network O_θ, policy network π_ϕ.

1. Learn approximate μ using behavioral cloning.
2. Update θ using objective in eq. (7).
3. Update ϕ using entropy-regularized objective $\mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\phi(\cdot|s)}[O_\theta(s,a)+\log\mu(a|s)+\alpha\mathcal{H}(\pi_\phi(\cdot|s))]$.
4. Repeat from 2.

Return: π_ϕ.

4.1. Conceptual Derivation

We start from the observation that we can represent the entropy smoothed Q-values of the behavior policy $\mu(\cdot|s)$ as

$$Q(s,a) = V(s) + \log\mu(a|s).$$

This decomposition of Q-values into state-value and log-policy is popular in the entropy-regularized online RL literature (Peters et al., 2010; Nachum et al., 2017), where the μ in our notation is treated as the policy π to be learned. Our own setting is markedly different from these previous approaches in that μ is the behavior policy and is fixed. Nevertheless, what would happen if we were to parameterize Q-values in this way, i.e, as

$$Q_\theta(s,a) := V_\theta(s) + \log\mu(a|s), \tag{4}$$

and then learn via standard TD error minimization (2)? Well, there is an advantage, but also a disadvantage.

First, the main advantage of this formulation is that, in contrast to a Q-function parameterized by its own neural network function approximator, the density $\mu(a|s)$ is well-defined for all actions, and thus Q is more likely to have a well-behaved landscape even though its training may only cover a small subset of the full action space. Accordingly, the learned policy π, when trained in the standard way to choose actions which maximize the Q-values, is thus encouraged to stay close to μ, without the need for an explicit divergence penalty as in (1). In practice, knowledge of μ is not explicitly provided, and one usually resorts to behavioral cloning on the offline dataset to approximate μ (Wu et al., 2019). Nevertheless, the advantage of the formulation in (4) still holds, since even if μ is trained on a sparse subset of all possible actions, the fact that it is a normalized probability distribution means that $\mu(a|s)$ assigns low probabilities to actions outside of $\mathcal{D}$. Still, in practice some density models might fail to generalize to out-of-distribution data (Kirichenko et al., 2020). For this reason, we parameterize the approximate density μ as a mixture density model (Bishop, 1994).

As for the disadvantage, it is clear that the formulation of Q in (4) is too restrictive. If this representation is used for training a new policy π, the new policy will be limited to copying the behavior policy μ, regardless of V_θ, and this will lead to suboptimal performance. In order to address this issue and enable the learned π to generalize beyond just mimicking μ, we propose replacing the value function $V_\theta(s)$ with a state-action value *offset* function $O_\theta(s,a)$:

$$Q_\theta(s,a) := O_\theta(s,a) + \log\mu(a|s). \tag{5}$$

With this representation one can learn a richer representation of Q-values. However, this parameterization can potentially put us back in the fully-parameterized Q_θ regime of vanilla actor critic. It is clear that the offset term must be suitably constrained or regularized to find an appropriate middle-ground between the overly-restrictive $V_\theta(s)$ and the fully-parameterized alternative that is liable to extrapolation errors and policy divergence.

To motivate what an appropriate regularization on O_θ should be, we begin by dissecting exactly how the offset term impacts the learned policy. Let's take a closer look at the policy updates in standard continuous-control actor critic (Lillicrap et al., 2019; Haarnoja et al., 2019). These updates are based on the chain-rule gradient computation below:

$$\nabla_\phi Q_\theta(s,\pi_\phi(s)) = \left[\nabla_a O_\theta(s,a) + \nabla_a \log\mu(a|s)\right]_{a=\pi_\phi(s)} \nabla_\phi\pi_\phi(s). \tag{6}$$

From this expression, it is clear that potential extrapolation issues with the actor arise via the gradient $\nabla_a O_\theta(s,a)$. That is, without appropriate constraints or regularizers on $O_\theta(s,a)$, its gradients might dominate over the gradients of the behavior policy $\nabla_a\log\mu(a|s)$. Therefore, as a way to control the trade-off between over-constraining π to be close to the behavior policy and learning more rich representations of Q-values, we propose using a gradient penalty regularizer of the form $\|\nabla_a O_\theta(s,a)\|^2$. Accordingly, the full critic optimization objective is as follows:

$$\min_\theta J(O_\theta + \log\mu) + \lambda \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{D}\\ a\sim\pi_\phi(\cdot|s)}} [\|\nabla_a O_\theta(s,a)\|^2], \tag{7}$$

where $J(\cdot)$ is defined as in eq. (2) and μ is not updated during critic learning. In this objective, λ is a hyperparameter that controls the contribution of the gradient penalty term. Unless otherwise noted, we set $\lambda = 0.1$ as the regularization coefficient.

A keen reader may note that the gradients in eq. (6) look similar to gradients of eq. (1), with the difference that we take gradients of the offset function instead of the critic function. However, in our setting the critic loss is substantially different, since the offset term plus the log behavior density learns to predict the *unmodified* Q-values of the training policy instead of Q-values augmented with a KL-divergence term. For example, if the rewards of the MDP already naturally

compel the learned policy to match the behavior policy, the offset term in Fisher-BRC can vanish, whereas the explicit divergence penalty in BRAC will bias the learned Q-values unnecessarily.

4.2. Fisher Divergence Derivation

We now show how the same objective in (7) may be derived from the perspective of Fisher divergence regularization. We begin by introducing the idea of Boltzmann policies – essentially policies expressed as a Boltzmann distribution with energy function given by a set of Q-values. We then present the Fisher divergence, and show how a Fisher divergence regularizer between a Boltzmann policy and the behavior policy reduces to the gradient penalty proposed in (7). Finally, we elaborate on connections to CQL, which we show can be interpreted as a KL divergence regularization between the Boltzmann policy and the behavior policy, and this insight may be of independent interest, since the original derivation of CQL is via a very different route.

Boltzmann Policies For a given Q-value function, the associated Boltzmann policy is given by the following expression:

$$\pi_{ebm}(a|s) := \frac{\exp(Q(s,a))}{\sum \exp(Q(s,a))}. \tag{8}$$

In an actor critic setting, the actor will recover this same policy if an entropy regularizer is added to the actor loss, as is commonly done (Haarnoja et al., 2019). While there are some works which use this representation of a policy more explicitly (Fox et al., 2015; Haarnoja et al., 2017; Nachum et al., 2017), a main disadvantage is that the normalization term may not be tractable for continuous actions, and so computing the policy distribution π_{ebm} explicitly requires performing computationally expensive numerical integration. Thus, it is more common to only recover the policy via entropy regularization on the actor loss.

Fisher Divergence In order to avoid issues with computing the normalization term, we consider the Fisher divergence, or Fisher information distance (Johnson, 2004):

$$F(p(\cdot), q(\cdot)) = \mathop{\mathbb{E}}_{x\sim p(\cdot)} \left[\|\nabla_x \log p(x) - \nabla_x \log q(x)\|^2 \right]. \tag{9}$$

As one can see from the formulation, in order to compute the Fisher divergence between two distributions, we need only have sampling access to $p(x)$ and the ability to compute $\nabla_x \log p(x), \nabla_x \log q(x)$. Crucially, the computation of either $\nabla_x \log p(x)$ or $\nabla_x \log q(x)$ does not require normalized distributions, since the normalization term (which is a constant with respect to x) disappears from $\log p(x), \log q(x)$ due to the differentiation. Thus, we can avoid computing the normalization term in (8).

Since the Fisher divergence is amenable to Boltzmann representations of policies, let's consider an optimization objective that consists of a squared TD-loss and a Fisher divergence term between the Boltzmann distribution and the behavior policy μ:

$$J(Q_\theta) + \lambda \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \left[F\left(\frac{\exp(Q(s,\cdot))}{\sum_a \exp(Q(s,a))}, \mu(\cdot|s) \right) \right] =$$
$$J(Q_\theta) + \lambda \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{D} \\ a\sim\pi_{emb}(\cdot|s)}} \left[\|\nabla_a \log \mu(a|s) - \nabla_a Q(s,a)\|^2 \right]. \tag{10}$$

The coefficient λ controls the strength of the Fisher divergence term. We can further simplify this objective by using the representation of Q proposed in (5):

$$J(O_\theta + \log \mu) + \lambda \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{D} \\ a\sim\pi_{ebm}(\cdot|s)}} \left[\|\nabla_a O_\theta(s,a)\|^2 \right]. \tag{11}$$

The only difference with eq. (7) is that actions are sampled from the training policy π_ϕ, while in this case the actions are sampled from the Bolzman policy π_{ebm}. In practice, sampling from π_{ebm} can be just as computationally expensive as computing the normalization term in eq. (3). However, as mentioned earlier, the Boltzmann policy π_{ebm} may be recovered by the actor via incorporation of an entropy regularizer in the actor loss. Thus, we propose to train our actor π_ϕ exactly in this manner (which is already popular in model-free actor critic algorithms (Haarnoja et al., 2019)), and then use it in eq. (11) as a plug-in approximation of π_{ebm}. In this way, we have arrived again to the same objective first defined in section 4.1.

The connection of our proposed objective to the Fisher divergence recalls similar quantities in the score matching and energy-based generative model literature. In fact, the Fisher divergence is a popular metric in these literatures exactly because it avoids an expensive computation of a log-normalizer, which is necessary for other common divergences (Lyu, 2012; Bao et al., 2020). Moreover, many works in the generative model literature employ gradient penalties, even if they do not explicitly make a connection to the Fisher divergence. This provides a further empirical advantage to our method, as these gradient penalties – due to their popularity – may be efficiently implemented using many modern machine learning libraries. We note that although we use a soft penalty, one can enforce hard constraints as in Spectral Normalized GANs (Miyato et al., 2018), but we leave this for future work.

Due the connection of our method to Fisher Divergence, we dub our method Fisher-BRC (Fisher Behavior Regularized Critic).

Connections to CQL The CQL objective, although originally derived in Kumar et al. (2020) from a very different perspective, can also be motivated as a regularizer on a

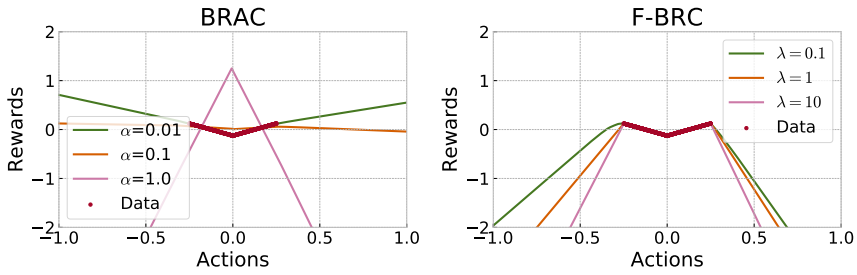

Figure 1. The objective landscapes (the regularized critic values) for the policy induced by the learned critic in BRAC or the parameterized offset critic in Fisher-BRC. The observed actions in the offline data are all within $[-0.25, 0.25]$, and suggest the optimal reward-maximizing actions as $\{-0.25, 0.25\}$. In BRAC (left), we see the landscape is heavily dependent on the choice of KL-divergence coefficient α, and it is easy to either over-regularize (with optimum around 0.0) or over-extrapolate (with optima far from the observed actions in $[-0.25, 0.25]$). On the other hand, due to the unique parameterization used in Fisher-BRC critic, its corresponding objective landscape correctly predicts the pessimistic reward values and peaks at the modes of the true reward function (right). We also see that Fisher-BRC is more robust to choice of regularizer coefficient λ.

Boltzmann policy. In the case of CQL, the regularizer is the common KL-divergence:

$$J(Q_\theta) + \lambda \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \left[D_{KL} \left(\mu(\cdot|s) \middle| \frac{\exp(Q(s,\cdot))}{\sum_a \exp(Q(s,a))} \right) \right]. \quad (12)$$

Expanding the KL-Divergence term yields,

$$D_{KL}(\mu(\cdot|s)|\pi_{emb}(\cdot|s)) = \mathop{\mathbb{E}}_{a\sim\mu(\cdot|s)} \left[\log \frac{\mu(a|s)}{\pi_{emb}(a|s)} \right] =$$

$$\mathop{\mathbb{E}}_{a\sim\mu(\cdot|s)} \left[\log \sum_a \exp(Q(s,a)) - Q(s,a) + \log \mu(a|s) \right],$$

and this is equivalent to the familiar form of CQL from (3), since $\log \mu(a|s)$ is a constant with respect to Q.

As mentioned earlier, for continuous distributions the normalization term in CQL is not tractable and necessitates expensive numerical integration. In CQL this integration is calculated by sampling from the training policy and importance weighting. Thus, our own method enjoys a significant computational advantage. Our use of a novel critic representation and the Fisher divergence allows us to circumvent this practical issue.

We note that in this derivation of CQL, the direction of divergence in eq. (12) is switched compared to eq. (10) in Fisher-BRC. One may derive a variant of Fisher-BRC using this same direction of the divergence, and this will result in the same expression in eq. (7), only that the expectation of the gradient penalty is switched to be over $(s, a) \sim \mathcal{D}$. Anecdotally, in initial experiments we did not observe large empirical differences when training with this alternative objective. However, due to the closer connection to the actor loss when using $a \sim \pi_\phi(\cdot|s)$ (see section 4.1), we stick to the formulation originally presented in eq. (7).

5. Experiments

We present empirical demonstrations of Fisher-BRC in a variety of settings. We start with a simple continuous bandit experiment that illustrates the difference between our method and more common behavior regularization techniques based on explicit divergence penalties. Then we evaluate our method against state-of-the-art offline RL model-based and model-free algorithms on the D4RL benchmark datasets. Finally, we analyze the effect of gradient penalty regularization and provide statistics on the computational advantage of Fisher-BRC compared to CQL.

5.1. Toy Continuous Bandit Problem

We begin with a simple conceptual demonstration comparing Fisher-BRC to the similar and common alternative of explicit divergence penalties applied to the learned policy. Specifically, we consider the loss in eq. (1), which corresponds to the policy update used in BRAC (Wu et al., 2019).

For the purpose of this experiment, we consider a continuous bandit with one-dimensional action space given by $[-1, 1]$. The rewards are given by

$$r(a) = \begin{cases} |a| - 0.125, & \text{if } a \in [-0.25, 0.25] \\ -\infty, & \text{otherwise.} \end{cases}$$

The offline training dataset is collected by sampling 1000 actions from a uniform distribution, $a \sim \mathcal{U}(-0.25, 0.25)$, and recording the corresponding rewards $r(a)$. Thus, the distribution of actions exhibits poor coverage of the full action space, and the rewards for $a \in [-1, -0.25] \cup [0.25, 1]$ are not observed in the dataset.

Both BRAC and Fisher-BRC require a fitted behavior policy. To do so, we fit a behavior policy $\mu(a)$ parameterized as

	BC	BRAC-p	BRAC-v	MBOP	CQL (GitHub)	CQL (Ours)	F-BRC (Ours)
halfcheetah-random	30.5	23.5	28.1	6.3 ± 4.0	27.1 ± 1.3	20.7 ± 0.6	33.3 ± 1.3
hopper-random	11.3	11.1	12.0	10.8 ± 0.3	10.6 ± 0.1	10.4 ± 0.1	11.3 ± 0.2
walker2d-random	4.1	0.8	0.5	8.1 ± 5.5	1.1 ± 2.2	10.0 ± 4.6	1.5 ± 0.7
halfcheetah-medium	36.1	44.0	45.5	44.6 ± 0.8	40.3 ± 0.3	38.9 ± 0.3	41.3 ± 0.3
walker2d-medium	6.6	72.7	81.3	41.0 ± 29.4	77.3 ± 3.8	69.2 ± 8.3	78.8 ± 1.0
hopper-medium	29.0	31.2	32.3	48.8 ± 26.8	42.2 ± 15.5	30.5 ± 0.7	99.4 ± 0.3
halfcheetah-expert	107.0	3.8	-1.1	-	54.4 ± 45.8	103.5 ± 1.3	108.4 ± 0.5
hopper-expert	109.0	6.6	3.7	-	67.7 ± 54.7	112.2 ± 0.2	112.3 ± 0.1
walker2d-expert	125.7	-0.2	-0.0	-	84.7 ± 42.7	107.2 ± 3.8	103.0 ± 5.0
halfcheetah-medium-expert	35.8	43.8	45.3	105.9 ± 17.8	21.7 ± 6.8	58.6 ± 8.7	93.3 ± 10.2
walker2d-medium-expert	11.3	-0.3	0.9	70.2 ± 36.2	104.0 ± 10.1	104.6 ± 10.4	105.2 ± 3.9
hopper-medium-expert	111.9	1.1	0.8	55.1 ± 44.3	111.3 ± 2.1	112.4 ± 0.2	112.4 ± 0.3
halfcheetah-mixed	38.4	45.6	45.9	42.3 ± 0.9	44.9 ± 1.1	42.0 ± 1.1	43.2 ± 1.5
hopper-mixed	11.8	0.7	0.8	12.4 ± 5.8	31.6 ± 3.6	29.0 ± 0.5	35.6 ± 1.0
walker2d-mixed	11.3	-0.3	0.9	9.7 ± 5.3	16.8 ± 3.1	16.5 ± 4.9	41.8 ± 7.9

Table 1. Comparison of our method (F-BRC) to prior work. The results for BC and BRAC are taken from Fu et al. (2020); the results for MBOP are taken from Argenson & Dulac-Arnold (2020); the results for CQL (GitHub) are taken from the author-provided open-source implementation of (Kumar et al., 2020); and the results for CQL (Ours) are from our own re-implementation of CQL. For all methods we run ourselves, we plot the normalized returns at the end of training (without early stopping) computed over 5 seeds. For every seed we run evaluation for 10 episodes.

a Laplace distribution.[2] Subsequently, we fit the critic for BRAC and F-BRC. In BRAC, we fit a critic using mean squared error to match the rewards: $\mathbb{E}_{(a,r)\sim\mathcal{D}}[(R_\theta(a)-r)^2]$. For Fisher-BRC, we use the representation and regularization from eq. (7), assuming an initialization of π to $\mathcal{U}$:

$$\mathop{\mathbb{E}}_{(a,r)\sim\mathcal{D}}[(O_\theta(a) + \log\mu(a) - r)^2] + \lambda \mathop{\mathbb{E}}_{a_{reg}\sim\mathcal{U}(-1,1)} \|\nabla_a O_\theta(a_{reg})\|^2.$$

These critics then determine the objective landscape for the learned policy. We plots these landscapes in fig. 1. Specifically, we plot $R_\theta(a) + \alpha\log\mu(a)$ for BRAC and $O_\theta(a) + \log\mu(a)$ for Fisher-BRC for a variety of choices of α and λ. Recall that the policy will be learned to choose actions which maximize these values. Thus, ideally these objective landscapes should possess optima around the globally optimal actions $\{-0.25, 0.25\}$.

We observe that it is hard to pick the KL-coefficient α for the policy loss landscape induced by BRAC to avoid either over generalizing (with optima outside of $[-0.25, 0.25]$) or over regularizing (with optima far from the optimal actions $\{-0.25, 0.25\}$); see fig. 1, left.

On the other hand, Fisher-BRC correctly constrains the learned value function – in fact, the value function is unmodified for in-distribution samples – and is robust to the choice of the regularization hyperparameter (fig. 1, right).

[2]Our choice of a Laplace parameterization is to match the absolute value appearing in the definition of rewards $r(a)$. If a Gaussian parameterization is used, then the rewards may be modified to a quadratic function to achieve the same result.

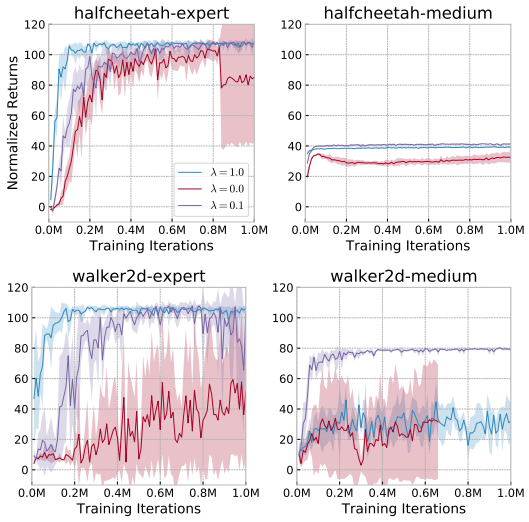

Figure 2. Performance of F-BRC for different values of the gradient penalty coefficient. A larger value, $\lambda = 1$, over-constraints the learned policy to stay close to the behavior policy. This leads to more stable performance on expert datasets, where the behavior policy is near-optimal, but worse performance on medium datasets. Without the regularization ($\lambda = 0.0$) Fisher-BRC collapses on most of these tasks; when the plot is cutoff, it means at least one of the seeds produced NaN values in training.

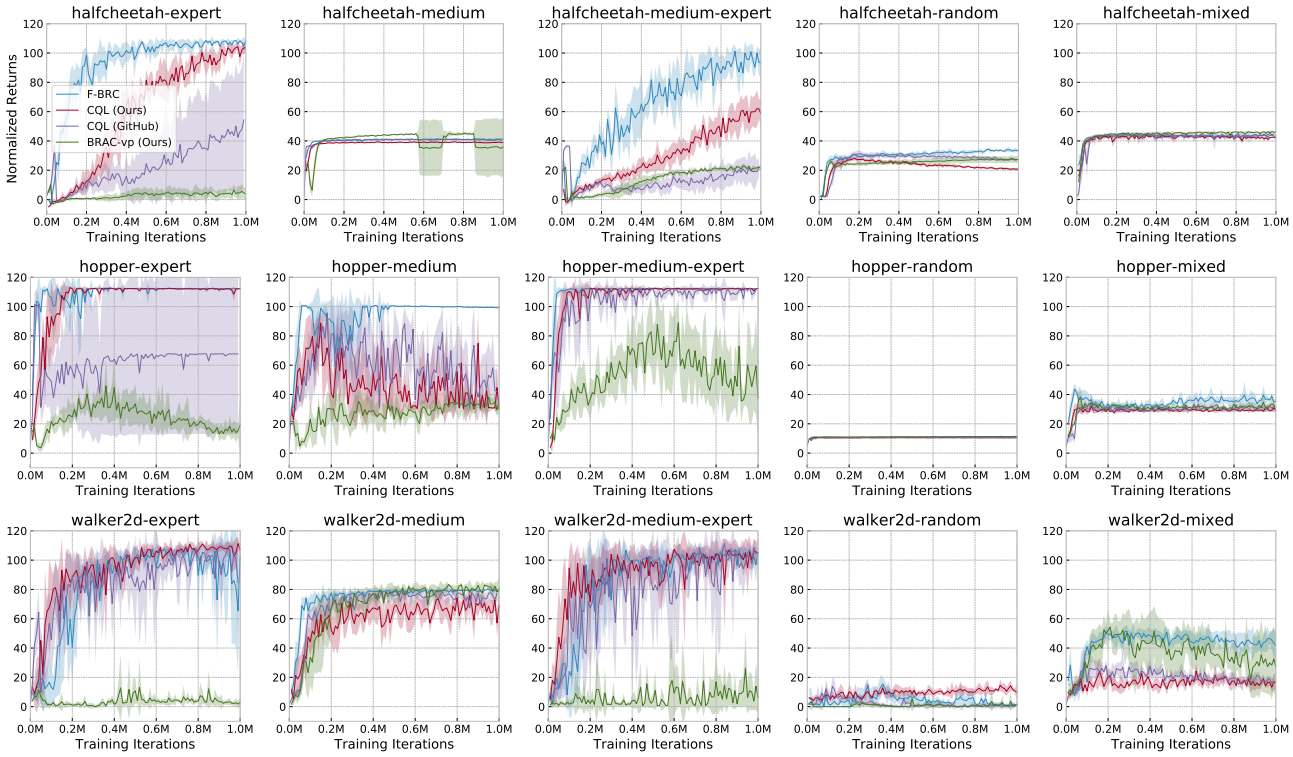

Figure 3. We compare F-BRC against prior methods in terms of convergence speed with respect to gradient updates steps. We see that Fisher-BRC enjoys better final performance and faster convergence in most tasks compared to BRAC and CQL.

5.2. Deep Offline RL Benchmarks

We continue to present Fisher-BRC on more complex environments. We compare our method to prior work on the OpenAI Gym MuJoCo tasks using D4RL datasets (Fu et al., 2020). We consider the following baselines: BRAC-vp and BRAC-pr (Wu et al., 2019), due to a similar policy learning objective, MBOP (Argenson & Dulac-Arnold, 2020), the state-of-state in offline model-based reinforcement learning, and CQL (Kumar et al., 2020), due to a similar connection with energy based learning. Our implementation for Fisher-BRC follows the standard SAC implementation, only that we use a 3-layer network as in CQL. Additional implementation details are in the appendix.

Overall performance The results of our method and all considered baselines are presented in Table 1. Our method performs comparably or surpasses prior work on most of the tasks. Notably, many of the baseline algorithms exhibit inconsistent performance across the tasks, achieving good performance on some while poor performance on others. In contrast, our Fisher-BRC exhibits consistent and good performance across almost all tasks.

Effect of gradient penalty As a way of investigating the effect of gradient penalty vs. the offset parameterization in Fisher-BRC, we evaluate different values of gradient penalty. We present results of $\lambda \in \{0.0, 0.1, 1.0\}$ in fig. 2. We note that the gradient penalty is an important component of Fisher-BRC, since when $\lambda = 0.0$ performance degrades dramatically. On the other hand, when λ is set too high ($\lambda = 1.0$), we see that the learned policy is over-constrained; i.e., we see that performance on the expert datasets is improved while performance is limited on the medium datasets, since the behavior policy in these datasets is highly sub-optimal. This is expected, since a high λ corresponds to a large gradient penalty on the regularization on the offset, compelling the offset to be near-constant with respect to actions.

Convergence speed We also evaluate the wall-clock training time of our method compared with CQL, which demonstrates comparable policy performance on the D4RL tasks but significantly different computational efficiency. The total training time for 1 million steps for Fisher-BRC is 1.4 hours of behavioral cloning pretraining followed by 6.2 hours of policy training. CQL total training time is 16.3 hours (which does not require pre-training of a behavior density policy). Our method converges faster not only in terms of gradient updates (see fig. 3), but it is also computationally faster due to omitting the expensive numerical integration to compute the log-sum-exp term of CQL. These experiments were carried out on a Google cloud instance

containing an AMD EPYC 7B12 CPU at 2.25GHz (using 8 of 64 available cores) and 32GB of RAM.

6. Conclusions

We have introduced Fisher-BRC, a simple critic representation and regularization technique for offline reinforcement learning. Our derivations highlight connections between our training objective and Fisher divergence regularization from score matching and energy-based model literature. Our method is easy to implement and highly performant. Compared to existing offline RL algorithms, Fisher-BRC exhibits better and more consistent performance across a variety of domains.

Acknowledgements

We thank George Tucker for reviewing a draft of this paper and providing valuable feedback. We also thank Aviral Kumar for discussions and help with CQL results.

References

Argenson, A. and Dulac-Arnold, G. Model-based offline planning. *arXiv preprint arXiv:2008.05556*, 2020.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan, 2017.

Baird, L. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pp. 30–37. Elsevier, 1995.

Bao, F., Xu, K., Li, C., Hong, L., Zhu, J., and Zhang, B. Variational (gradient) estimate of the score function in energy-based latent variable models. *arXiv preprint arXiv:2010.08258*, 2020.

Bishop, C. M. Mixture density networks. 1994.

Finn, C., Christiano, P., Abbeel, P., and Levine, S. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models, 2016.

Fox, R., Pakman, A., and Tishby, N. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

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

Ghasemipour, S. K. S., Schuurmans, D., and Gu, S. S. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl, 2021.

Gonzalez-Garcia, A., van de Weijer, J., and Bengio, Y. Image-to-image translation for cross-domain disentanglement, 2018.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies, 2017.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft actor-critic algorithms and applications, 2019.

Heess, N., Silver, D., and Teh, Y. W. Actor-critic reinforcement learning with energy-based policies. In Deisenroth, M. P., Szepesvári, C., and Peters, J. (eds.), *Proceedings of the Tenth European Workshop on Reinforcement Learning*, volume 24 of *Proceedings of Machine Learning Research*, pp. 45–58, Edinburgh, Scotland, 30 Jun–01 Jul 2013. PMLR. URL http://proceedings.mlr.press/v24/heess12a.html.

Jaques, N., Gu, S., Bahdanau, D., Hernández-Lobato, J. M., Turner, R. E., and Eck, D. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *International Conference on Machine Learning*, pp. 1645–1654. PMLR, 2017.

Jaques, N., Ghandeharioun, A., Shen, J. H., Ferguson, C., Lapedriza, A., Jones, N., Gu, S., and Picard, R. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Johnson, O. *Information theory and the central limit theorem*. World Scientific, 2004.

Kappen, H. J., Gómez, V., and Opper, M. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

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

Kirichenko, P., Izmailov, P., and Wilson, A. G. Why normalizing flows fail to detect out-of-distribution data, 2020.

Kostrikov, I., Agrawal, K. K., Dwibedi, D., Levine, S., and Tompson, J. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning, 2018.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

Lange, S., Gabel, T., and Riedmiller, M. Batch reinforcement learning. In *Reinforcement learning*, pp. 45–73. Springer, 2012.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning, 2019.

Lyu, S. Interpretation and generalization of score matching. *arXiv preprint arXiv:1205.2629*, 2012.

Matsushima, T., Furuta, H., Matsuo, Y., Nachum, O., and Gu, S. Deployment-efficient reinforcement learning via model-based offline optimization. *arXiv preprint arXiv:2006.03647*, 2020.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1QRgziT-.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. *arXiv preprint arXiv:1702.08892*, 2017.

Nachum, O., Dai, B., Kostrikov, I., Chow, Y., Li, L., and Schuurmans, D. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

Peters, J., Mulling, K., and Altun, Y. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

Thomas, P. S. *Safe reinforcement learning*. PhD thesis, University of Massachusetts Libraries, 2015.

van Amersfoort, J., Smith, L., Teh, Y. W., and Gal, Y. Uncertainty estimation using a single deep deterministic neural network, 2020.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

A. Implementation Details

Behavior policy We fit the behavior model using a conditional Mixture of Gaussians (Bishop, 1994) with tanh squashing (Haarnoja et al., 2017). We use 5 mixture components. We train the density model with Adam optimizer (Kingma & Ba, 2014) for 10^6 steps and starting from learning rate 10^{-3} and decreasing it by 10 at $8 \cdot 10^5$ and $9 \cdot 10^5$ gradient update steps. Similar to BRAC we train the behavior actor with SAC-style entropy regularization with the same target entropy. We parameterize the model as a 3 layer MLP with relu activations and 256 hidden units.

Actor and critic learning Our implementation is based on Soft Actor Critic (Haarnoja et al., 2019). As in CQL we do not add entropy to the rewards and we modify the critic loss to accommodate the additional regularization term. We use default SAC hyper parameters without additional tuning, in contrast to CQL and BRAC which tune policy learning rate. Following CQL we increased network size for the actor and the critic to 3 layer MLP with 256 hidden units.

Survival bonus The linear term used in CQL can be seen as adding a survival bonus for the environments with early early termination. The derivation is included in the Appendix. Adding a positive constant to the rewards does not have an effect on the optimal policy in infinite horizon MDPs, but in practice Q-targets for terminal states are replaced with 0 that leads to having either a survival bonus or step penalty. For this reason, we add a reward bonus to our implementation as well for fair comparison. We choose the same value $\lambda_{cql} = 5$ as in CQL.

In particular, one can verify that

$$\begin{aligned}
&\nabla_\theta[-\lambda_{cql} Q_\theta(s,a) + (\gamma Q_{\hat{\theta}}(s',a') + r(s,a) - Q_\theta(s,a))^2] = \\
&-\lambda_{cql}\nabla_\theta Q_\theta(s,a) - (\gamma Q_{\hat{\theta}}(s',a') + r(s,a) - Q_\theta(s,a))\nabla_\theta Q_\theta(s,a) = \\
&-(\gamma Q_{\hat{\theta}}(s',a') + [r(s,a) + \lambda_{cql}] - Q_\theta(s,a))\nabla_\theta Q_\theta(s,a) = \\
&\nabla_\theta(\gamma Q_{\hat{\theta}}(s',a') + [r(s,a) + \lambda_{cql}] - Q_\theta(s,a))^2.
\end{aligned}$$

B. Gradient Penalty Ablation

We also present a full set of results for the ablations in fig. 2.

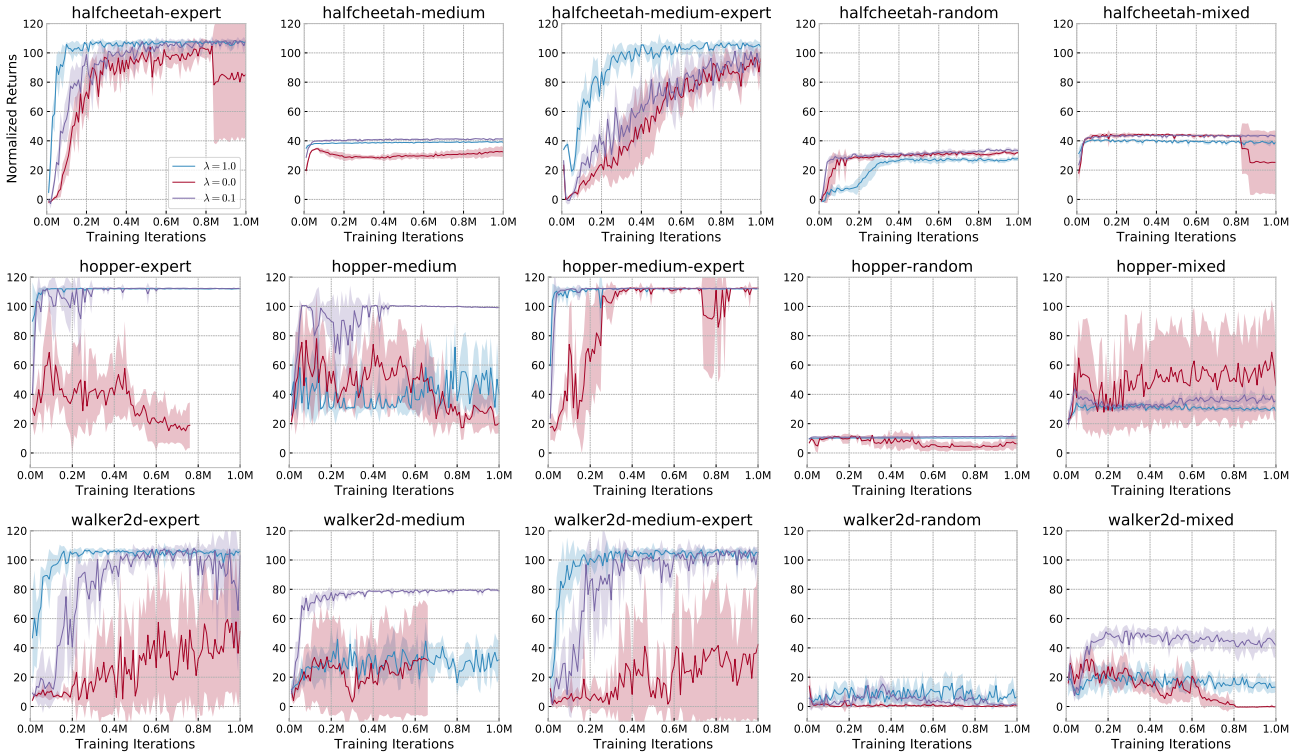

Figure 4. Performance of F-BRC for different values of the gradient penalty coefficient. A larger value, $\lambda = 1$, over-constraints the learned policy to stay close to the behavior policy. This leads to more stable performance on expert datasets, where the behavior policy is near-optimal, but worse performance on medium datasets. Without the regularization ($\lambda = 0.0$) Fisher-BRC collapses on most of these tasks; when the plot is cutoff, it means at least one of the seeds produced NaN values in training.

C. Critic Regularization Ablation

We also evaluate the effect of gradient penalty of standard Soft Actor Critic without the critic representation introduced in this paper.

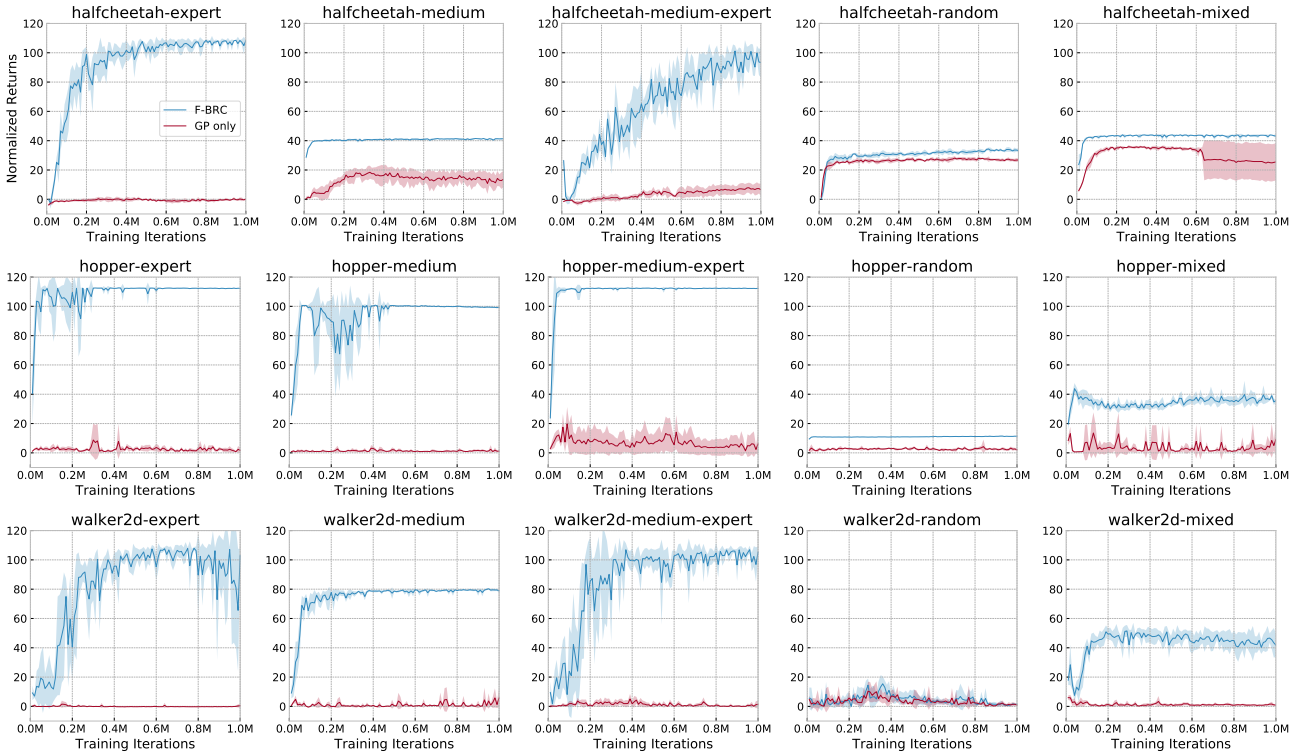

Figure 5. Performance of F-BRC without our critic representation. Without the critic representation, the gradient penalty term alone fails to improve performance of the underlying reinforcement learning algorithm on the offline datasets.