BRPO: Batch Residual Policy Optimization

Sungryull Sohn[*,1,2] , **Yinlam Chow**[*,1] , **Jayden Ooi**[*,1] ,
Ofir Nachum[1] , **Honglak Lee**[1,2] , **Ed Chi**[1] and **Craig Boutilier**[1]
[1]Google Research
[2]University of Michigan
srsohn@umich.edu, {yinlamchow, jayden, ofirnachum, honglak, edchi, cboutilier}@google.com

Abstract

In batch reinforcement learning (RL), one often constrains a learned policy to be close to the behavior (data-generating) policy, e.g., by constraining the learned action distribution to differ from the behavior policy by some maximum degree that is the *same at each state*. This can cause batch RL to be overly conservative, unable to exploit large policy changes at frequently-visited, high-confidence states without risking poor performance at sparsely-visited states. To remedy this, we propose *residual policies*, where the allowable deviation of the learned policy is *state-action-dependent*. We derive a new for RL method, *BRPO*, which learns both the policy and allowable deviation that jointly maximize a lower bound on policy performance. We show that BRPO achieves the state-of-the-art performance in a number of tasks.

1 Introduction

Deep reinforcement learning (RL) methods are increasingly successful in domains such as games [Mnih *et al.*, 2013], and robotic manipulation [Nachum *et al.*, 2019]. Much of this success relies on the ability to collect new data through *online* interactions with the environment during training, often relying on simulation. Unfortunately, this approach is impractical in many real-world applications where faithful simulators are rare, and in which active data collection through interactions with the environment is costly, time consuming, and risky.

Batch (or offline) RL [Lange *et al.*, 2012] is an emerging research direction that aims to circumvent the need for online data collection, instead learning a new policy using only offline trajectories generated by some behavior policy (e.g., the currently deployed policy in some application domain). In principle, any off-policy RL algorithm (e.g., DDPG [Lillicrap *et al.*, 2015], DDQN [Hasselt *et al.*, 2016]) may be used in this batch (or more accurately, "offline") fashion; but in practice, such methods have been shown to fail to learn when presented with arbitrary, static, off-policy data. This can arise for several reasons: lack of exploration [Lange *et al.*, 2012], generalization error on out-of-distribution samples in value estimation [Kumar *et al.*, 2019], or high-variance policy gradients induced by covariate shift [Mahmood *et al.*, 2014].

Various techniques have been proposed to address these issues, many of which can be interpreted as constraining or regularizing the learned policy to be *close* to the behavior policy [Fujimoto *et al.*, 2018; Kumar *et al.*, 2019] (see further discussion below). While these batch RL methods show promise, none provide improvement guarantees relative to the behavior policy. In domains for which batch RL is well-suited (e.g., due to the risks of active data collection), such guarantees can be critical to deployment of the resulting RL policies.

In this work, we use the well-established methodology of *conservative policy improvement (CPI)* [Kakade and Langford, 2002] to develop a theoretically principled use of behavior-regularized RL in the batch setting. Specifically, we parameterize the learned policy as a *residual policy*, in which a base (behavior) policy is combined linearly with a learned *candidate policy* using a mixing factor called the *confidence*. Such residual policies are motivated by several practical considerations. First, one often has access to offline data or logs generated by a deployed base policy which is known to perform reasonably well. The offline data can be used by an RL method to learn a candidate policy with better *predicted* performance, but if confidence in parts of that prediction is weak, relying on the base policy may be desirable. The base policy may also incorporate soft business constraints or some form of interpretability. Our residual policies blend the two in a learned, non-uniform fashion. When deploying a new policy, we use the CPI framework to derive updates that learn both the candidate policy and the confidence that jointly maximize a lower bound on performance improvement relative to the behavior policy. Crucially, while traditional applications of CPI, such as TRPO [Schulman *et al.*, 2015], use a constant or state-independent confidence, our performance bounds and learning rules are based on *state-action-dependent* confidences—this gives rise to bounds that are less conservative than their CPI counterparts.

In Sec. 2, we formalize residual policies and in Sec. 3 analyze a novel *difference-value function*. Sec. 4 holds our main result, a tighter lower bound on policy improvement for our residual approach (vs. CPI and TRPO). We derive the BRPO algorithm in Sec. 5 to jointly learn the candidate policy and confidence; experiments in Sec. 6 show its effectiveness.

[*]Equal contribution

2 Preliminaries

We consider a *Markov decision process (MDP)* $\mathcal{M} = \langle S, A, R, T, P_0 \rangle$, with state space S, action space A, reward function R, transition kernel T, and initial state distribution P_0. A policy π interacts with the environment, starting at $s_0 \sim P_0$. At step t, the policy samples an action a_t from a distribution $\pi(\cdot|s_t)$ over A and applies. The environment emits a reward $r_t = R(s_t, a_t) \in [0, R_{\max}]$ and next state $s_{t+1} \sim T(\cdot|s_t, a_t)$. In this work, we consider discounted infinite-horizon problems with discount factor $\gamma \in [0, 1)$.

Let $\Delta = \{\pi : S \times A \to [0,1],\ \sum_a \pi(a|s) = 1\}$ be the set of Markovian stationary policies. The expected (discounted) cumulative return of policy $\pi \in \Delta$, is $J_\pi := \mathbb{E}_{T,\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 \sim P_0]$. Our aim is to find an optimal policy $\pi^* \in \arg\max_{\pi \in \Delta} J_\pi$. In reinforcement learning (RL), we must do so without knowledge of R, T, using only trajectory data generated from the environment (see below) or access to a simulator (see above).

We consider pure offline or *batch RL*, where the learner has access to a fixed data set (or *batch*) of state-actions-reward-next-state samples $B = \{(s, a, r, s')\}$, generated by a (known) *behavior policy* $\beta(\cdot|s)$. No additional data collection is permitted. We denote by d_β the γ-discounted occupation measure of the MDP w.r.t. β.

In this work, we study the problem of *residual policy optimization (RPO)* in the batch setting. Given the behavior policy $\beta(a|s)$, we would like to learn a *candidate policy* $\rho(a|s)$ and a state-action *confidence* $\lambda(s, a)$, such that the final *residual policy* $\pi(a|s) = (1 - \lambda(s, a)) \cdot \beta(a|s) + \lambda(s, a) \cdot \rho(a|s)$ maximizes total return. As discussed above, this type of mixture allows one to exploit an existing, “well-performing” behavior policy. Intuitively, $\lambda(s, a)$ should capture how much we can trust ρ at each s, a pair, given the available data. To ensure that the residual policy is a probability distribution at every state $s \in S$, we constrain the confidence λ to lie in the set $\Lambda(s) = \{\lambda : S \times A \to [0, 1] : \sum_a \lambda(s, a) (\beta(a|s) - \rho(a|s)) = 0\}$.

Related work. Similar to the above policy formulation, CPI [Kakade and Langford, 2002] also develops a policy mixing methodology that guarantees performance improvement when the confidence λ is a constant. However, CPI is an online algorithm, and it learns the candidate policy independently of (not jointly with) the mixing factor; thus, extension of CPI to offline, batch setting is unclear. Other existing work also deals with online residual policy learning without jointly learning mixing factors [Johannink *et al.*, 2019; Silver *et al.*, 2018]. Common applications of CPI may treat λ as a hyper-parameter, which specifies the maximum total-variation distance between the learned and behavior policy distributions (see standard proxies in [Schulman *et al.*, 2015; Pirotta *et al.*, 2013] for details). *Batch-constrained Q-learning (BCQ)* [Fujimoto *et al.*, 2018; Fujimoto *et al.*, 2019] incorporates the behavior policy when defining the admissible action set in Q-learning for selecting the highest-valued actions that are similar to data samples in the batch. *BEAR* [Kumar *et al.*, 2019] is motivated as a means to control the accumulation of out-of-distribution value errors; but its main algorithmic contribution is realized by adding a regularizer to the loss that measures the kernel maximum mean discrepancy (MMD) [Gretton *et al.*, 2007] between the learned and behavior policies similar to KL-control [Jaques *et al.*, 2019]. Algorithms such as SPI [Ghavamzadeh *et al.*, 2016] and SPIBB [Laroche and Trichelair, 2017] bootstraps the learned policy with the behavior policy when the uncertainty in the update for current state-action pair is high, where the uncertainty is measured by the visitation frequency of state-action pairs in the batch data. While these methods work well in some applications it is unclear if they have any performance guarantees.

3 The Difference-value Function

We begin by defining and characterizing the *difference-value function*, a concept we exploit in the derivation of our batch RPO method in Secs. 4 and 5. For any $s \in S$, let $V_\pi(s)$ and $V_\beta(s)$ be the value functions induced by policies π and β, respectively. Using the structure of the residual policy, we establish two characterizations of the *difference-value function* $\Delta V_{\pi,\beta}(s) := V_\pi(s) - V_\beta(s)$.

Lemma 1. *Let $A_\pi(s, a) := Q_\pi(s, a) - V_\pi(s)$ be the advantage function w.r.t. residual policy π, where Q_π is the state-action value. The difference-value is $\Delta V_{\pi,\beta}(s) = \mathbb{E}_{T,\beta}[\sum_{t=0}^{\infty} \gamma^t \Delta \hat{A}_{\pi,\beta,\rho,\lambda}(s_t) \mid s_0 = s]$, where $\Delta \hat{A}_{\beta,\rho,\lambda}(s) = \sum_{a \in A} \beta(a|s) \cdot \lambda(s, a) \cdot \frac{\rho(a|s) - \beta(a|s)}{\beta(a|s)} \cdot A_\pi(s, a)$ is the residual reward that depends on λ and difference of candidate policy ρ and behavior policy β.*

This result establishes that the difference value is essentially a value function w.r.t. the residual reward. Moreover, it is proportional to the advantage of the target policy, the confidence, and the difference of policies. While the difference value can be estimated from behavior data batch B, this formulation requires knowledge of the advantage function A_π w.r.t. the target policy, which must be re-learned at every π-update in an off-policy fashion. Fortunately, we can show that the difference value can also be expressed as a function of the advantage w.r.t. the *behavior policy* β:

Theorem 2. *Let $A_\beta(s, a) := Q_\beta(s, a) - V_\beta(s)$ be the advantage function induced by β, in which Q_β is the state-action value. The difference-value is given by $\Delta V_{\pi,\beta}(s) = \mathbb{E}_{T,\pi}[\sum_{t=0}^{\infty} \gamma^t \Delta A_{\beta,\rho,\lambda}(s_t) | s_0 = s]$, where*

$$\Delta A_{\beta,\rho,\lambda}(s) = \sum_{a \in A} \beta(a|s) \cdot \lambda(a|s) \cdot \frac{\rho(a|s) - \beta(a|s)}{\beta(a|s)} \cdot A_\beta(s, a)$$

is the residual reward that depends on λ and difference of candidate policy ρ and behavior policy β.

In our RPO approach, we exploit the nature of the difference-value function to solve the maximization w.r.t. the confidence and candidate policy: $(\lambda^*(s, \cdot), \rho^*(\cdot|s)) \in \arg\max_{\lambda \in \Lambda(s), \rho \in \Delta} \Delta V_\pi(s)$, $\forall s \in S$. Since $\lambda(s, \cdot) = 0$ implies $\Delta V_{\pi,\beta}(s) = 0$, the optimal difference-value function $\Delta V^*(s) := \max_{\lambda \in \Lambda(s), \rho \in \Delta} \Delta V_\pi(s)$ is always lower-bounded by 0. We motivate computing (λ, ρ) with the above difference-value formulation rather than as a standard RL problem as follows. In the tabular case, optimizing (λ, ρ) with either formulation gives an identical result. However, both the difference-value function in Theorem 2 and the standard RL objective require sampling data generated by the updated policy π. In the batch setting, when fresh samples are unavailable, learning (λ, ρ) with off-policy data may incur instability due to high generalization error [Kumar *et al.*, 2019]. While this can be alleviated by adopting the CPI methodology, applying CPI directly to RL can be overly conservative [Schulman *et al.*, 2015]. By contrast, we leverage the special structure of the difference-value function (e.g., non-negativity) below, using this new formulation together with CPI to derive a less conservative RPO algorithm.

4 Batch Residual Policy Optimization

We now develop an RPO algorithm that has stable learning performance in the batch setting and *performance improvement guarantees*. For the sake of brevity, in the following we only present the main results on performance guarantees of RPO. Proofs of these results can be found in the appendix of the extended paper. We begin with the following baseline result, directly applying Corollary 1 of the TRPO result to RPO to ensure the residual policy π performs no worse than β.

Lemma 3. *For any value function $U : S \to \mathbb{R}$, the difference-return satisfies $J_\pi - J_\beta \geq \frac{1}{1-\gamma}\widetilde{L}_{U,\beta,\rho,\lambda} - \frac{2\gamma}{(1-\gamma)^2} \cdot \epsilon_{U,\beta,\rho,\lambda} \cdot \mathbb{E}_{s\sim d_\beta}\left[\sqrt{\frac{1}{2}D_{\mathrm{KL}}(\beta(s)\|\rho(s))}\right]$, where the surrogate objective and the penalty weight are*

$$\widetilde{L}_{U,\beta,\rho,\lambda} := \mathop{\mathbb{E}}_{(s,a,s')\sim d_\beta}\left[\lambda(s,a)\cdot\frac{\rho(a|s)-\beta(a|s)}{\beta(a|s)}\cdot\Delta U(s,a,s')\right],$$

$$\epsilon_{U,\beta,\rho,\lambda} := \max_s |\mathbb{E}_{\pi,T}[\Delta U(s,a,s')]|,$$

where $\Delta U(s,a,s') := R(s,a) + \gamma U(s') - U(s)$.

When $U = V_\pi$, one has $\mathbb{E}_{a\sim\pi(s)}[\Delta U(s,a,s')] = 0$, $\forall s \in S$, which implies that the inequality is tight—this lemma then coincides Lemma 1. While this CPI result forms the basis of many RL algorithms (e.g., TRPO, PPO), in many cases it is very loose since $\epsilon_{U,\beta,\rho,\lambda}$ is a maximum over all states. Thus, using this bound for policy optimization may be *overly conservative*, i.e., algorithms which rely on this bound must take very small policy improvement steps, especially when the penalty weight $\epsilon_{U,\beta,\rho,\lambda}$ is large, i.e., $|\epsilon_{U,\beta,\rho,\lambda}/(1-\gamma)| >> |\widetilde{L}_{U,\beta,\rho,\lambda}|$. While this approach may be reasonable in online settings—when collection of new data (with an updated behavior policy $\beta \leftarrow \pi$) is allowed—in the batch setting it is challenging to overcome such conservatism.

To address this issue, we develop a CPI method that is specifically tied to the difference-value formulation, and uses a state-action-dependent confidence $\lambda(s,a)$. We first derive the following theorem, which bounds the difference returns that are generated by β and π.

Theorem 4. *The difference return of (π, β) satisfies*

$$J_\pi - J_\beta \geq \frac{1}{1-\gamma}\left(L'_{\beta,\rho,\lambda} - \frac{\gamma}{1-\gamma}\cdot L''_{\beta,\rho,\lambda}\cdot\max_{s_0\in S} L'''_{\beta,\rho,\lambda}(s_0)\right),$$

where the surrogate objective function, regularization, and penalty weight are given by

$$L'_{\beta,\rho,\lambda} := \mathbb{E}_{(s,a)\sim d_\beta}\left[\lambda(s,a)\cdot\frac{\rho(a|s)-\beta(a|s)}{\beta(a|s)}\cdot A_\beta(s,a)\right]$$

$$L''_{\beta,\rho,\lambda} := \mathbb{E}_{(s,a)\sim d_\beta}\left[\lambda(s,a)\cdot\frac{|\rho(a|s)-\beta(a|s)|}{\beta(a|s)}\right]$$

$$L'''_{\beta,\rho,\lambda}(s_0) := \mathbb{E}_{(s,a)\sim d_\beta(s_0)}\left[\lambda(s,a)\cdot\frac{|\rho(a|s)-\beta(a|s)|}{\beta(a|s)}\cdot|A_\beta(s,a)|\right]$$

respectively, in which $d_\beta(s_0)$ is the discounted occupancy measure w.r.t. β given initial state s_0.

Unlike the difference-value formulations in Lemma 1 and Theorem 2, which require the knowledge of advantage function A_π or the trajectory samples generated by π, the lower bound in Theorem 4 is comprised only of terms that can be estimated directly using the data batch B (i.e., data generated by β). This makes it a natural objective function for batch RL. Notice also that the surrogate objective, the regularization, and the penalty weight in the lower bound are each proportional to the confidence and to the relative difference of the candidate and behavior policies. However, the $\max$ operator requires state enumeration to compute this lower bound, which is intractable when S is large or uncountable.

We address this by introducing a slack variable $\kappa \geq 0$ to replace the $\max$-operator with suitable constraints. This allows the bound on the difference return to be rewritten as: $J_\pi - J_\beta \geq \frac{1}{1-\gamma}L'_{\beta,\rho,\lambda} - \min_{\kappa\geq L'''_{\beta,\rho,\lambda}(s_0),\,\forall s_0}\frac{\gamma}{(1-\gamma)^2}L''_{\beta,\rho,\lambda}\kappa$.

Consider the Lagrangian of the lower bound:

$$\frac{L'_{\beta,\rho,\lambda}}{1-\gamma} - \min_{\kappa\geq 0}\max_{\eta(s)\geq 0,\forall s}\frac{\gamma\cdot L''_{\beta,\rho,\lambda}\cdot\kappa}{(1-\gamma)^2} - \sum_s \eta(s)(\kappa - L'''_{\beta,\rho,\lambda}(s)).$$

To simplify this saddle-point problem, we restrict the Lagrange multiplier to be $\eta(s) = \eta\cdot P_0(s) \geq 0$, where $\eta \geq 0$ is a scalar multiplier. Using this approximation and the strong duality of linear programming [Boyd and Vandenberghe, 2004] over primal-dual variables (κ, η), the saddle-point problem on $(\lambda, \rho, \eta, \kappa)$ can be re-written as

$$\begin{aligned}\mathcal{L}_{\beta,\rho,\lambda} &:= \max_{\eta\geq 0}\min_{\kappa\geq 0}\frac{L'_{\beta,\rho,\lambda} - L''_{\beta,\rho,\lambda}\cdot\kappa\cdot\frac{\gamma}{1-\gamma} - \eta\cdot\kappa + \eta L'''_{\beta,\rho,\lambda}}{1-\gamma}\\ &= \frac{1}{1-\gamma}\left(L'_{\beta,\rho,\lambda} - \frac{\gamma}{1-\gamma}L''_{\beta,\rho,\lambda}\cdot L'''_{\beta,\rho,\lambda}\right), \qquad (1)\end{aligned}$$

where $L'''_{\beta,\rho,\lambda} = \mathbb{E}_{s\sim P_0}[L'''_{\beta,\rho,\lambda}(s)]$. The equality is based on the KKT condition on (κ, η). Notice that the only difference between the CPI lower bound in Theorem 4 and the objective function $\mathcal{L}_{\beta,\rho,\lambda}$ is that the $\max$ operator is replaced by expectation w.r.t the initial distribution.

With certain assumptions on the approximation error of the Lagrange multiplier parametrization $\eta(s) \approx P_0(s)$, we can characterize the gap between the original CPI objective function in Theorem 4 and $\mathcal{L}_{\beta,\rho,\lambda}$. One approach is to look into the KKT condition of the original saddle-point problem and bound the sub-optimality gap introduced by this Lagrange parameterization. Similar derivations can be found in the analysis of approximate linear programming (ALP) algorithms [Abbasi-Yadkori *et al.*, 2019; Farias and Roy, 2003].

Compared with the vanilla CPI result from Lemma 3, there are two characteristics in problem (1) that make the optimization w.r.t. $\mathcal{L}_{\beta,\rho,\lambda}$ less conservative. First, the penalty weight

$L'''_{\beta,\rho,\lambda}$ here is smaller than $\epsilon_{U,\beta,\rho,\lambda}$ in Lemma 3, which means that the corresponding objective has less incentive to force ρ to be close to β. Second, compared with entropy regularization in vanilla CPI, here the regularization and penalty weight are both linear in $\lambda \in \Lambda \subseteq [0,1]^{|A|}$; thus, unlike vanilla CPI, whose objective is linear in λ, our objective is quadratic in λ—this modification ensures the optimal value is not a *degenerate extreme point* of Λ.[1]

5 The BRPO Algorithm

We now develop the BRPO algorithm, for which the general pseudo-code is given in Algorithm 1. Recall that if the candidate policy ρ and confidence λ are jointly optimized

$$(\rho^*, \lambda^*) \in \arg\max_{\lambda\in\Lambda,\ \rho\in\Delta} \mathcal{L}_{\beta,\rho,\lambda}, \tag{2}$$

then the residual policy $\pi^*(a|s) = (1-\lambda^*(s,a))\beta(a|s) + \lambda^*(s,a)\rho^*(a|s)$ performs no worse than behavior policy β. Generally, solutions for problem (2) use a form of minorization-maximization (MM) [Hunter and Lange, 2004], a class of methods that also includes expectation maximization. In the terminology of MM algorithms, $\mathcal{L}_{\beta,\rho,\lambda}$ is a surrogate function satisfying the following *MM properties*:

$$J_\pi - J_\beta \geq \mathcal{L}_{\beta,\rho,\lambda},\ \ J_\beta - J_\beta = \mathcal{L}_{\beta,\beta,\lambda} = \mathcal{L}_{\beta,\rho,0} = 0, \tag{3}$$

which guarantees that it minorizes the difference-return $J_\pi - J_\beta$ with equality at $\lambda = 0$ (with arbitrary ρ) or at $\rho = \beta$ (with arbitrary λ). This algorithm is also reminiscent of proximal gradient methods. We optimize λ and ρ in RPO with a simple two-step *coordinate-ascent*. Specifically, at iteration $k \in \{0,1,\ldots,K\}$, given confidence λ_{k-1}, we first compute an updated candidate policy ρ_k, and with ρ_k fixed, we update λ_k, i.e., $\mathcal{L}_{\beta,\rho_k,\lambda_k} \geq \mathcal{L}_{\beta,\rho_k,\lambda_{k-1}} \geq \mathcal{L}_{\beta,\rho_{k-1},\lambda_{k-1}}$. When λ and ρ are represented tabularly or with linear function approximators, under certain regularity assumptions (the Kurdyka-Lojasiewicz property [Xu and Yin, 2013]) coordinate ascent guarantees global convergence (to the limit point) for BRPO.

However, when more complex representations (e.g., neural networks) are used to parameterize these decision variables, this property no longer holds. While one may still compute (λ^*, ρ^*) with first-order methods (e.g., SGD), convergence to local optima is not guaranteed. To address this, we next further restrict the MM procedure to develop closed-form solutions for both the candidate policy and the confidence.

The Closed-form Candidate Policy ρ. To effectively update the candidate policy when given the confidence $\lambda \in \Lambda$, we develop a closed-form solution for ρ. Our approach is based on maximizing the following objective, itself a more conservative version of the CPI lower bound $\mathcal{L}_{\beta,\rho,\lambda}$:

$$\max_{\rho\in\Delta} \hat{\mathcal{L}}_{\beta,\rho,\lambda} := \mathbb{E}_{s\sim d_\beta}\Bigg[\mathbb{E}_{a\sim\beta}\left[\lambda(s,a)\frac{\rho(a|s)-\beta(a|s)}{\beta(a|s)}A_\beta\right] - \frac{\gamma\max\{\kappa_\lambda(s), \kappa_{|A_\beta|\lambda}(s)\}}{2(1-\gamma)}\cdot D_{\mathrm{KL}}(\rho\|\beta)(s)\Bigg]\cdot\frac{1}{1-\gamma}, \tag{4}$$

where $\kappa_g(s) = (1 + \log\mathbb{E}_\beta[\exp(g(a|s)^2)]) > 0$ for any arbitrary non-negative function g. To show that $\hat{\mathcal{L}}_{\beta,\rho,\lambda}$ in (4) is an eligible lower bound (so that the corresponding ρ-solution is an MM), we need to show that it satisfies the properties in (3). When $\rho = \beta$, by the definition of $\hat{\mathcal{L}}_{\beta,\rho,\lambda}$ the second property holds. To show the first property, we first consider the following problem:

$$\max_{\rho\in\Delta} \frac{1}{1-\gamma}\left(L'_{\beta,\rho,\lambda} - \frac{\gamma}{1-\gamma}\widetilde{L}''_{\beta,\rho,\lambda}\cdot\widetilde{L}'''_{\beta,\rho,\lambda}\right), \tag{5}$$

where $L'_\beta(\rho,\lambda)$ is given in Theorem 4, and

$$\begin{aligned}\widetilde{L}''_{\beta,\rho,\lambda} &= \mathbb{E}_{s\sim d_\beta}\left[\sqrt{\kappa_\lambda(s)\cdot D_{\mathrm{KL}}(\rho\|\beta)(s)/2}\right],\\ \widetilde{L}'''_{\beta,\rho,\lambda} &= \mathbb{E}_{s\sim d_\beta}\left[\sqrt{\kappa_{|A_\beta|\lambda}(s)\cdot D_{\mathrm{KL}}(\rho\|\beta)(s)/2}\right].\end{aligned} \tag{6}$$

The concavity of $\sqrt{(\cdot)}$ (i.e., $\mathbb{E}_{s\sim d_\beta}[\sqrt{(\cdot)}] \leq \sqrt{\mathbb{E}_{s\sim d_\beta}[(\cdot)]}$) and monotonicity of expectation imply that the objective in (4) is a lower bound of that in (7) below. Furthermore, by the weighted Pinsker's inequality [Bolley and Villani, 2005] $\sum_a |g(a|s)(\rho(a|s) - \beta(a|s))| \leq \sqrt{\kappa_g(s)D_{\mathrm{KL}}(\rho\|\beta)(s)/2}$, we have: $0 \leq \widetilde{L}''_{\beta,\rho,\lambda} \leq L''_{\beta,\rho,\lambda}$, $0 \leq \widetilde{L}'''_{\beta,\rho,\lambda} \leq L'''_{\beta,\rho,\lambda}$, which implies the objective in (5) is a lower-bound of that in (2) and validates the first MM property.

Now recall the optimization problem: $\max_{\rho\in\Delta}\hat{\mathcal{L}}_{\beta,\rho,\lambda}$. Since this optimization is over the state-action mapping ρ, the Interchangeability Lemma [Shapiro *et al.*, 2009] allows swapping the order of $\mathbb{E}_{s\sim d_\beta}$ and $\max_{\rho\in\Delta}$. This implies that at each $s \in S$ the candidate policy can be solved using:

$$\begin{aligned}\rho^*_\lambda &\in \arg\max_{\rho(\cdot|s)\in\Delta}\mathbb{E}_{a\sim\beta}\left[\left(\lambda\frac{\rho-\beta}{\beta}A_\beta\right) - \tau_\lambda(s)\log\frac{\rho(a|s)}{\beta(a|s)}\right]\\ &= \arg\max_{\rho(\cdot|s)\in\Delta}\mathbb{E}_{a\sim\rho}\left[\lambda(s,a)A_\beta - \tau_\lambda(s)\log\frac{\rho(a|s)}{\beta(a|s)}\right],\end{aligned} \tag{7}$$

where $\tau_\lambda(s) = \gamma\max\{\kappa_\lambda(s), \kappa_{|A_\beta|\lambda}(s)\}/(2-2\gamma)$ is the state-dependent penalty weight of the relative entropy regularization. By the KKT condition of (7), the optimal candidate policy ρ^*_λ has the form

$$\rho^*_\lambda(a|s) = \frac{\beta(a|s)\cdot\exp\left(\frac{\lambda(s,a)A_\beta}{\tau_\lambda(s)}\right)}{\mathbb{E}_{a'\sim\beta}[\exp(\lambda(s,a')A_\beta(s,a')/\tau_\lambda(s))]}. \tag{8}$$

Notice that the optimal candidate policy is a *relative softmax policy*, which is a common solution policy for many entropy-regularized RL algorithms [Haarnoja *et al.*, 2018]. Intuitively, when the mixing factor vanishes (i.e., $\lambda(s,a) = 0$), the candidate policy equals to the behavior policy, and with confidence we obtain the candidate policy by modifying the behavior policy β via *exponential twisting*.

The Closed-form Confidence λ. Given candidate policy ρ, we derive efficient scheme for computing the confidence that solves the MM problem: $\max_{\lambda\in\Lambda}\mathcal{L}_{\beta,\rho,\lambda}$. Recall that this optimization can be reformulated as a concave quadratic program (QP) with linear equality constraints, which has a unique optimal solution [Faybusovich and Moore, 1997]. However, since the decision variable (i.e., the confidence mapping) is infinite-dimensional, solving this QP is intractable without some assumptions about this mapping, To

[1]For example, when λ is state-dependent (which automatically satisfies the equality constraints in Λ), the linear objective in vanilla CPI makes the optimal value $\lambda^*(\cdot|s)$ a *0-1 vector*. Especially when $\frac{2\gamma}{1-\gamma}\mathbb{E}_{s\sim d_\beta}\left[\sqrt{\frac{1}{2}D_{\mathrm{KL}}(\beta(s)\|\rho(s))}\right]$ is large, then most entries of λ^* become zero, i.e., π will be very close to β.

resolve this issue, instead of using the surrogate objective $\mathcal{L}_{\beta,\rho,\lambda}$ in MM, we turn to its sample-based estimate. Specifically, given a batch of data $B = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^{|B|}$ generated by the behavior policy β, denote by

$$\overline{L}'_{\beta,\rho,\lambda} := \frac{1}{1-\gamma} \cdot \frac{1}{|B|} \sum_{i=1}^{|B|} \overline{\lambda}_i^\top \cdot ((\rho - \beta) \cdot A_\beta)_i$$

$$\overline{L}''_{\beta,\rho,\lambda} := \frac{1}{1-\gamma} \cdot \frac{1}{|B|} \sum_{i=1}^{|B|} \overline{\lambda}_i^\top \cdot |\rho - \beta|_i$$

$$\overline{L}'''_{\beta,\rho,\lambda} := \frac{\gamma}{1-\gamma} \cdot \frac{1}{|B|} \sum_{i=1}^{|B|} \overline{\lambda}_i^\top \cdot (|\rho - \beta| \cdot |A_\beta|)_i$$

the sample-average approximation (SAA) of functions $L'_{\beta,\rho,\lambda}$, $L''_{\beta,\rho,\lambda}$, and $L'''_{\beta,\rho,\lambda}$ respectively, where

$$(\rho - \beta)A_\beta = \{(\rho(\cdot|s_i) - \beta(\cdot|s_i))A_\beta(s_i, \cdot)\}_{s_i \in B}, \quad (9)$$

$$|\rho - \beta||A_\beta| = \{(|\rho(\cdot|s_i) - \beta(\cdot|s_i)||A_\beta(s_i, \cdot)|)\}_{s_i \in B}, \quad (10)$$

and $|\rho - \beta| = \{|\rho(\cdot|s_i) - \beta(\cdot|s_i)|\}_{s_i \in B}$ are $|A| \cdot |B|$-dimensional vectors, where each element is generated by a state sample from B, and $\overline{\lambda} = \{\lambda(\cdot|s_i)\}_{s_i \in B}$ is a $|A| \cdot |B|$-dimensional decision vector, where each $|A|$-dimensional element vector corresponds to the confidence w.r.t. state samples in B. Since the expectation in $L'_{\beta,\rho,\lambda}$, $L''_{\beta,\rho,\lambda}$, and $L'''_{\beta,\rho,\lambda}$ is over the stationary distribution induced by the behavior policy, all the SAA functions are *unbiased* Monte-Carlo estimates of their population-based counterparts. We now define $\overline{\mathcal{L}}_{\beta,\rho,\lambda} := \overline{L}'_{\beta,\rho,\lambda} - \overline{L}''_{\beta,\rho,\lambda}\overline{L}'''_{\beta,\rho,\lambda}$ as the SAA-MM objective and use this to solve for the confidence vector $\overline{\lambda}$ over the batch samples.

Now consider the following maximization problem:

$$\max_{\overline{\lambda} \in \overline{\Lambda}} \langle (\rho - \beta) \cdot A_\beta, \overline{\lambda} \rangle - \frac{\gamma}{|B|(1-\gamma)} \langle |\rho - \beta|, \overline{\lambda} \rangle \cdot \langle |\rho - \beta||A_\beta|, \overline{\lambda} \rangle, \quad (11)$$

where the feasible set $\overline{\Lambda} = \{\lambda \in [0, 1] : \sum_{a \in A} \lambda(s_i, a) \cdot (\rho(a|s_i) - \beta(a|s_i)) \cdot = 0, \forall i \in \{1, \ldots, |B|\}\}$ only imposes constraints on the states that appear in the batch B.

This finite-dimensional QP problem can be expressed in the following quadratic form:

$$\max_{\overline{\lambda} \in \overline{\Lambda}} \overline{\lambda}^\top \Big((\rho - \beta) \cdot A_\beta \Big) - \frac{1}{2} \cdot \overline{\lambda}^\top \Theta \overline{\lambda},$$

where the symmetric matrix is given by

$$\Theta_{\beta,\rho} := \frac{\gamma (D_{|A_\beta|} \cdot \Delta_{\beta,\rho} + \Delta_{\beta,\rho} \cdot D_{|A_\beta|}^\top)}{|B|(1-\gamma)},$$

$\Delta_{\beta,\rho} = |\rho - \beta| \cdot |\rho - \beta|^\top$, and $D_{|A_\beta|} = \mathrm{diag}(\{|A_\beta|\}_{a \in A, s \in B})$ is a $|B| \cdot |A| \times |B| \cdot |A|$-diagonal matrix whose elements are the absolute advantage function. By definition, Θ is positive-semi-definite, hence the QP above is concave. Using its KKT condition, the unique optimal confidence vector over batch B is given as

$$\overline{\lambda}^* = \min\{1, \max\{0, \Theta_{\beta,\rho}^{-1}((\rho - \beta) \cdot A_\beta + M_{\beta,\rho}^\top \nu_{\beta,\rho}\}\}, \quad (12)$$

where $M_{\beta,\rho} = \mathrm{blkdiag}(\{[\rho(a|x) - \beta(a|x)]_{a \in A}\}_{x \in B})$ is a $|B| \cdot |A| \times |B|$-matrix, and the Lagrange multiplier $\nu_{\beta,\rho} \in \mathbb{R}^{|B|}$ w.r.t. constraint $M_{\beta,\rho}\overline{\lambda} = 0$ is given by

$$\nu_{\beta,\rho} = -(M_{\beta,\rho}^\top \Theta_{\beta,\rho}^{-1} M_{\beta,\rho})^{-1} (M_{\beta,\rho}^\top \Theta_{\beta,\rho}^{-1} (\rho - \beta) \cdot A_\beta). \quad (13)$$

We first construct the confidence function $\lambda(s, a)$ from the confidence vector $\overline{\lambda}^*$ over B, in the following tabular fashion:

$$\lambda(s, a) = \begin{cases} \overline{\lambda}^*_{s,a} & \text{if } (s, a) \in B \\ 0 & \text{otherwise} \end{cases}.$$

While this construction preserves optimality w.r.t. the CPI objective (2), it may be overly conservative, because the policy equates to the behavior policy by setting $\lambda = 0$ at state-action pairs that are not in B (i.e., no policy improvement). To alleviate this conservatism, we propose to learn a confidence function that generalizes to out-of-distribution samples.

Learning the Confidence. Given a confidence vector $\overline{\lambda}^*$ corresponding to samples in batch B, we learn the confidence function $\lambda_\phi(s, a)$ in supervised fashion. To ensure that the confidence function satisfies the constraint: $\lambda_\phi \in \Lambda$, i.e., $\sum_a \lambda_\phi(s, a)(\rho(a|s) - \beta(a|s)) = 0$, $\lambda_\phi(s, a) \in [0, 1]$, $\forall s, a$[2], we parameterize it as

$$\lambda_{\phi^*}(s, a) := \frac{\pi_{\phi^*}(a|s) - \beta(a|s)}{\rho(a|s) - \beta(a|s)}, \ \forall (s, a) \in S \times A, \quad (14)$$

where $\pi_\phi \in \Delta$ is a learnable policy mapping, such that $\min\{\beta(a|s), \rho(a|s)\} \leq \pi_\phi(a|s) \leq \max\{\beta(a|s), \rho(a|s)\}$, $\forall s, a$. We then learn ϕ via the following KL distribution-fitting objective [Rusu *et al.*, 2015]:

$$\min_\phi \frac{1}{B} \sum_{(s,a) \in B} \pi_\phi(a|s) \log \left(\frac{\pi_\phi(a|s)}{(1 - \overline{\lambda}^*_{s,a})\beta(a|s) + \overline{\lambda}^*_{s,a} \cdot \rho(a|s)} \right).$$

While this approach learns λ_ϕ by generalizing the confidence vector to out-of-distribution samples, when π_ϕ is a NN, one challenge is to enforce the constraint: $\min\{\beta(a|s), \rho(a|s)\} \leq \pi_\phi(a|s) \leq \max\{\beta(a|s), \rho(a|s)\}$, $\forall s, a$. Instead, using an in-graph convex optimization NN [Amos and Kolter, 2017], we parameterize λ_ϕ with a NN with the following *constraint-projection layer* $\Phi : S \to A$ before the output:

$$\Phi(s) \in \arg \min_{\lambda \in \mathbb{R}^{|A|}} \frac{1}{2} \sum_{a \in A} \|\lambda_a - \widetilde{\lambda}^*_{s,a}\|^2,$$

$$\text{s.t.} \sum_a \lambda_a (\rho(a|s) - \beta(a|s)) = 0, \ 0 \leq \lambda \leq 1, \quad (15)$$

where, at any $s \in S$, the $|A|$-dimensional confidence vector label $\{\widetilde{\lambda}^*_{s,a}\}_{a \in A}$ is equal to $\{\overline{\lambda}^*_{\overline{s},a}\}_{a \in A}$ chosen from the batch confidence vector $\overline{\lambda}^*$ such that $\overline{s}$ in B is closest to s. Indeed, analogous to the closed-form solution in (12), this projection layer has a closed-form QP formulation with linear constraints: $\Phi(s) = \min\{1, \max\{0, \widetilde{\lambda}^*_{s,\cdot} + (\rho(\cdot|s) - \beta(\cdot|s)) \cdot \mu_{\beta,\rho}\}\}$, where Lagrange multiplier $\mu_{\beta,\rho}$ is given by $\mu_{\beta,\rho} = -(\rho(\cdot|s) - \beta(\cdot|s))^\top \widetilde{\lambda}^*_{s,\cdot} / \|\rho(\cdot|s) - \beta(\cdot|s)\|^2$.

Although the ρ-update is theoretically justified, in practice, when the magnitude of $\kappa_\lambda(s)$ becomes large (due to the conservatism of the weighted Pinsker inequality), the relative-softmax candidate policy (8) may be too close to the behavior policy β, impeding learning of the residual policy (i.e., $\pi \approx \beta$). To avoid this in practice, we can upper bound the temperature, i.e., $\kappa_\lambda(s) \leftarrow \min\{\kappa_{\max}, \kappa_\lambda(s)\}$, or introduce a weak temperature-decay schedule, i.e., $\kappa_\lambda(s) \leftarrow \kappa_\lambda(s) \cdot \epsilon^k$, with a tunable $\epsilon \in [0, 1)$.

[2]If one restricts λ_ϕ to be only *state*-dependent, this constraint immediately holds.

Environment-ε	DQN	BRPO-C	BRPO (ours)	BCQ	KL-Q	SPIBB	BC	Behavior Policy
Acrobot-0.05	**-91.2 ± 9.1**	-94.6 ± 3.8	**-91.9 ± 9.0**	-96.9 ± 3.7	-93.0 ± 2.6	-103.5 ± 24.1	-102.3 ± 5.0	-103.9
Acrobot-0.15	**-83.1 ± 5.2**	-91.7 ± 4.0	**-86.1 ± 10.1**	-97.1 ± 3.3	-92.1 ± 3.2	-91.1 ± 44.8	-113.1 ± 5.6	-114.3
Acrobot-0.25	**-83.4 ± 3.9**	-91.2 ± 4.1	**-85.3 ± 4.8**	-96.7 ± 3.1	-90.0 ± 2.9	-86.0 ± 5.8	-124.1 ± 7.0	-127.2
Acrobot-0.50	-84.3 ± 22.6	-90.9 ± 3.4	**-83.7 ± 16.6**	**-77.8 ± 13.5**	-84.5 ± 3.8	-106.8 ± 102.7	-173.7 ± 8.1	-172.4
Acrobot-1.00	-208.9 ± 174.8	**-156.8 ± 22.0**	**-121.7 ± 10.2**	-236.0 ± 85.6	-227.5 ± 148.1	-184.8 ± 150.2	-498.3 ± 1.7	-497.3
CartPole-0.05	82.7 ± 0.5	220.8 ± 117.0	**336.3 ± 122.6**	255.4 ± 11.1	**323.0 ± 13.5**	28.8 ± 1.2	205.6 ± 19.6	219.1
CartPole-0.15	299.3 ± 133.5	305.6 ± 95.2	**409.9 ± 64.4**	255.3 ± 11.4	**357.7 ± 84.1**	137.7 ± 11.7	151.6 ± 27.5	149.5
CartPole-0.25	368.5 ± 129.3	**405.1 ± 74.4**	316.8 ± 64.1	247.4 ± 128.7	**441.4 ± 79.8**	305.2 ± 119.7	103.0 ± 20.4	101.9
CartPole-0.50	271.5 ± 52.0	**358.3 ± 114.1**	**433.8 ± 93.5**	282.5 ± 111.8	314.1 ± 107.0	310.4 ± 128.0	39.7 ± 5.1	37.9
CartPole-1.00	118.3 ± 0.3	**458.6 ± 51.5**	**369.0 ± 42.3**	194.0 ± 25.1	209.7 ± 48.4	147.1 ± 0.1	22.6 ± 1.5	21.9
LunarLander-0.05	-236.4 ± 177.6	35.6 ± 61.7	**88.2 ± 32.0**	81.5 ± 14.9	**84.4 ± 26.3**	-200.4 ± 81.7	75.8 ± 17.7	73.7
LunarLander-0.15	-215.6 ± 140.4	79.6 ± 29.7	**103.9 ± 49.8**	80.3 ± 16.8	61.4 ± 39.0	**86.1 ± 73.3**	76.4 ± 16.6	84.9
LunarLander-0.25	2.5 ± 101.3	109.5 ± 40.7	**141.6 ± 11.0**	83.5 ± 14.6	78.7 ± 48.8	**166.0 ± 90.6**	57.9 ± 13.1	57.3
LunarLander-0.50	-104.6 ± 68.3	42.5 ± 71.4	**101.0 ± 39.6**	-13.2 ± 44.9	**66.2 ± 78.0**	-134.6 ± 17.1	-32.6 ± 6.5	-36.0
LunarLander-1.00	-65.6 ± 45.9	**53.5 ± 44.1**	**81.8 ± 42.1**	-69.1 ± 44.0	-139.2 ± 29.1	-107.1 ± 94.4	-177.4 ± 13.1	-182.6

Table 1: The mean and st. dev. of average return with the best hyperparameter configuration (with the top-2 results boldfaced). Full training curves are given in the appendix. For BRPO-C, the optimal confidence parameter is found by grid search.

Algorithm 1 BRPO algorithm

```
Require: $B$: batch data; Tunable parameter $\mu \in [0,1]$
1: for $t = 1, \dots, N$ do
2:     Sample mini-batch of transitions $(s, a, r, d, s') \sim B$
3:     Compute $\overline{\lambda}^*$ from Eq. (12)
4:     Update confidence $\phi^*$ by Eq. (15)
5:     Update candidate policy $\rho^*_{\lambda_{\phi^*}}$ by Eq. (8)
6:     Construct target critic network $V_{\theta'}(s') := (1 - \mu)\mathbb{E}_{a' \sim \beta}[Q_{\theta'}(s', a')] + \mu \max_{a'} Q_{\theta'}(s', a')$
7:     Update $\theta \leftarrow \arg\max_\theta \frac{1}{2}\left(Q_\theta(s,a) - r - \gamma V_{\theta'}(s')\right)^2$
8:     Update target network: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
```

6 Experimental Results

To illustrate the effectiveness of BRPO, we compare against six baselines: DQN [Mnih *et al.*, 2013], discrete BCQ [Fujimoto *et al.*, 2019], KL-regularized Q-learning (KL-Q) [Jaques *et al.*, 2019], SPIBB [Laroche and Trichelair, 2017], Behavior Cloning (BC) [Kober and Peters, 2010], and BRPO-C, which is a simplified version of BRPO that uses a constant (tunable) parameter as confidence weight[3]. We do not consider ensemble models, thus do not include methods like BEAR [Kumar *et al.*, 2019] among our baselines. CPI is also excluded since it is subsumed by BRPO-C with a grid search on the confidence. It is also generally inferior to BRPO-C because candidate policy learning does not optimize the performance of the final mixture policy. We evaluated on three discrete-action OpenAI Gym tasks [Brockman *et al.*, 2016]: Cartpole-v1, Lunarlander-v2, and Acrobot-v1.

The behavior policy in each environment is trained using standard DQN until it reaches 75% of optimal performance, similar to the process adopted in related work (e.g., [Fujimoto *et al.*, 2018]). To assess how exploration and the quality of behavior policy affect learning, we generate five sets of data for each task by injecting different random exploration into the same behavior policy. Specifically, we add ε-greedy exploration for $\varepsilon = 1$ (fully random), 0.5, 0.25, 0.15, and 0.05, generating $100K$ transitions each for batch RL training.

All models use the same architecture for a given environment—details (architectures, hyper-parameters, etc.) are described in the appendix of the extended paper. While training is entirely offline, policy performance is evaluated online using the simulator, at every 1000 training iterations. Each measurement is the average return w.r.t. 40 evaluation episodes and 5 random seeds, and results are averaged over a sliding window of size 10.

Table 1 shows the average return of BRPO and the other baselines under the best hyper-parameter configurations in each task setting. Behavior policy performance decreases as ε increases, as expected, and BC matches that very closely. DQN performs poorly in the batch setting. Its performance improves as ε increases from 0.05 to 0.25, due to increased state-action coverage, but as ε goes higher (0.5, 1.0), the state space coverage decreases again since the (near-) random policy is less likely to reach a state far away from the initial state.

BCQ, KL-Q and SPIBB follow the behavior policy in some ways, and showing different performance characteristics over the data sets. The underperformance relative to BRPO is more prominent for very low or very high ε, suggesting deficiency due to overly conservative updates or following the behavior policy too closely, when BRPO is able to learn.

Since BRPO exploits the statistics of each (s, a) pair in the batch data, it achieves good performance in almost all scenarios, outperforming the baselines. The stable performance and robustness across various scenarios make BRPO an appealing algorithm for batch/offline RL in real-world, where it is usually difficult to estimate the amount of exploration required prior to training, given access only to batch data.

7 Concluding Remarks

We have presented *Batch Residual Policy Optimization* (BRPO) for learning residual policies in batch RL settings. Inspired by CPI, we derived learning rules for jointly optimizing both the candidate policy and *state-action dependent* confidence mixture of a residual policy to maximize a conservative lower bound on policy performance. BRPO is thus more exploitative in areas of state space that are well-covered by the batch data and more conservative in others. While we have shown successful application of BRPO to various benchmarks, future work includes deriving finite-sample analysis of BRPO, and applying BRPO to more practical batch domains (e.g., robotic manipulation, recommendation systems).

[3]For algorithms designed for online settings, we modify data collection to sample only from offline / batch data.

References

[Abbasi-Yadkori *et al.*, 2019] Y. Abbasi-Yadkori, P. Bartlett, X. Chen, and A. Malek. Large-scale markov decision problems via the linear programming dual. *arXiv:1901.01992*, 2019.

[Amos and Kolter, 2017] B. Amos and Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *ICML*, pages 136–145, 2017.

[Bolley and Villani, 2005] F. Bolley and C. Villani. Weighted csiszár-kullback-pinsker inequalities and applications to transportation inequalities. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, volume 14, pages 331–352, 2005.

[Boyd and Vandenberghe, 2004] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[Brockman *et al.*, 2016] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

[Farias and Roy, 2003] P. De Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations research*, 51(6):850–865, 2003.

[Faybusovich and Moore, 1997] L. Faybusovich and J. Moore. Infinite-dimensional quadratic optimization: interior-point methods and control applications. *Applied Mathematics and Optimization*, 36(1):43–66, 1997.

[Fujimoto *et al.*, 2018] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. *arXiv:1812.02900*, 2018.

[Fujimoto *et al.*, 2019] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. Benchmarking batch deep reinforcement learning algorithms. *arXiv:1910.01708*, 2019.

[Ghavamzadeh *et al.*, 2016] M. Ghavamzadeh, M. Petrik, and Y. Chow. Safe policy improvement by minimizing robust baseline regret. In *NIPS*, 2016.

[Gretton *et al.*, 2007] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. A kernel approach to comparing distributions. In *AAAI*, 2007.

[Haarnoja *et al.*, 2018] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ICML*, 2018.

[Hasselt *et al.*, 2016] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double Q-learning. In *AAAI*, 2016.

[Hunter and Lange, 2004] D. Hunter and K. Lange. A tutorial on MM algorithms. *The American Statistician*, 58(1):30–37, 2004.

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

[Johannink *et al.*, 2019] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. Ojea, E. Solowjow, and S.Levine. Residual reinforcement learning for robot control. In *ICRA*, pages 6023–6029. IEEE, 2019.

[Kakade and Langford, 2002] S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *ICML*, pages 267–274, 2002.

[Kober and Peters, 2010] J. Kober and J. Peters. Imitation and reinforcement learning. *IEEE Robotics & Automation Magazine*, 17(2):55–62, 2010.

[Kumar *et al.*, 2019] A. Kumar, J. Fu, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv:1906.00949*, 2019.

[Lange *et al.*, 2012] S. Lange, T. Gabel, and M. Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[Laroche and Trichelair, 2017] R. Laroche and P. Trichelair. Safe policy improvement with baseline bootstrapping. *arXiv:1712.06924*, 2017.

[Lillicrap *et al.*, 2015] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv:1509.02971*, 2015.

[Mahmood *et al.*, 2014] A. Mahmood, H. van Hasselt, and R. Sutton. Weighted importance sampling for off-policy learning with linear function approximation. In *NIPS*, pages 3014–3022, 2014.

[Mnih *et al.*, 2013] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv:1312.5602*, 2013.

[Nachum *et al.*, 2019] O. Nachum, M. Ahn, H. Ponte, S. Gu, and V. Kumar. Multi-agent manipulation via locomotion using hierarchical sim2real. *CoRL*, 2019.

[Pirotta *et al.*, 2013] Matteo Pirotta, Marcello Restelli, Alessio Pecorino, and Daniele Calandriello. Safe policy iteration. In *ICML*, pages 307–315, 2013.

[Rusu *et al.*, 2015] A. Rusu, S. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv:1511.06295*, 2015.

[Schulman *et al.*, 2015] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *ICML*, pages 1889–1897, 2015.

[Shapiro *et al.*, 2009] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[Silver *et al.*, 2018] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling. Residual policy learning. *arXiv:1812.06298*, 2018.

[Xu and Yin, 2013] Y. Xu and W. Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences*, 6(3):1758–1789, 2013.