OptiDICE: Offline Policy Optimization via Stationary Distribution Correction Estimation

Jongmin Lee [1 *] **Wonseok Jeon** [2 3 *] **Byung-Jun Lee** [4] **Joelle Pineau** [2 3 5] **Kee-Eung Kim** [1 6]

Abstract

We consider the offline reinforcement learning (RL) setting where the agent aims to optimize the policy solely from the data without further environment interactions. In offline RL, the distributional shift becomes the primary source of difficulty, which arises from the deviation of the target policy being optimized from the behavior policy used for data collection. This typically causes overestimation of action values, which poses severe problems for model-free algorithms that use bootstrapping. To mitigate the problem, prior offline RL algorithms often used sophisticated techniques that encourage underestimation of action values, which introduces an additional set of hyperparameters that need to be tuned properly. In this paper, we present an offline RL algorithm that prevents overestimation in a more principled way. Our algorithm, OptiDICE, directly estimates the stationary distribution corrections of the optimal policy and does not rely on policy-gradients, unlike previous offline RL algorithms. Using an extensive set of benchmark datasets for offline RL, we show that OptiDICE performs competitively with the state-of-the-art methods.

1. Introduction

The availability of large-scale datasets has been one of the important factors contributing to the recent success in machine learning for real-world tasks such as computer vision (Deng et al., 2009; Krizhevsky et al., 2012) and natural language processing (Devlin et al., 2019). The standard workflow in developing systems for typical machine learning tasks is to train and validate the model on the dataset, and then to deploy the model with its parameter fixed when we are satisfied with training. This offline training allows us to address various operational requirements of the system without actual deployment, such as acceptable level of prediction accuracy rate once the system goes online.

However, this workflow is not straightforwardly applicable to the standard setting of reinforcement learning (RL) (Sutton & Barto, 1998) because of the online learning assumption: the RL agent needs to continuously explore the environment and learn from its trial-and-error experiences to be properly trained. This aspect has been one of the fundamental bottlenecks for the practical adoption of RL in many real-world domains, where the exploratory behaviors are costly or even dangerous, e.g. autonomous driving (Yu et al., 2020b) and clinical treatment (Yu et al., 2020a).

Offline RL (also referred to as batch RL) (Ernst et al., 2005; Lange et al., 2012; Fujimoto et al., 2019; Levine et al., 2020) casts the RL problem in the offline training setting. One of the most relevant areas of research in this regard is the off-policy RL (Lillicrap et al., 2016; Haarnoja et al., 2018; Fujimoto et al., 2018), since we need to deal with the distributional shift resulting from the trained policy being deviated from the policy used to collect the data. However, without the data continuously collected online, this distributional shift cannot be reliably corrected and poses a significant challenge to RL algorithms that employ bootstrapping together with function approximation: it causes compounding overestimation of the action values for model-free algorithms (Fujimoto et al., 2019; Kumar et al., 2019), which arises from computing the bootstrapped target using the predicted values of *out-of-distribution* actions. To mitigate the problem, most of the current offline RL algorithms have proposed sophisticated techniques to encourage underestimation of action values, introducing an additional set of hyperparameters that needs to be tuned properly (Fujimoto et al., 2019; Kumar et al., 2019; Jaques et al., 2019; Lee et al., 2020; Kumar et al., 2020).

In this paper, we present an offline RL algorithm that essentially eliminates the need to evaluate out-of-distribution actions, thus avoiding the problematic overestimation of values. Our algorithm, *Offline Policy **Opti**mization via Stationary **DI**stribution **C**orrection **E**stimation* (OptiDICE),

*Equal contribution [1]School of Computing, KAIST [2]Mila, Quebec AI Institute [3]School of Computer Science, McGill University [4]Gauss Labs Inc. [5]Facebook AI Research [6]Graduate School of AI, KAIST. Correspondence to: Jongmin Lee <jmlee@ai.kaist.ac.kr>, Wonseok Jeon <jeonwons@mila.quebec>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

estimates stationary distribution ratios that correct the discrepancy between the data distribution and the *optimal* policy's stationary distribution. We first show that such optimal stationary distribution corrections can be estimated via minimax optimization that does not involve sampling from the target policy. Then, we derive and exploit the closed-form solution to the sub-problem of the aforementioned minimax optimization, which reduces the overall problem into an unconstrained convex optimization, and thus greatly stabilizing our method. To the best of our knowledge, OptiDICE is the first deep offline RL algorithm that optimizes policy purely in the space of *stationary distributions*, rather than in the space of either Q-functions or policies (Nachum et al., 2019b). In the experiments, we demonstrate that OptiDICE performs competitively with the state-of-the-art methods using the D4RL offline RL benchmarks (Fu et al., 2021).

2. Background

We consider the reinforcement learning problem with the environment modeled as a Markov Decision Process (MDP) $M = \langle S, A, T, R, p_0, \gamma \rangle$ (Sutton & Barto, 1998), where S is the set of states s, A is the set of actions a, $R : S \times A \to \mathbb{R}$ is the reward function, $T : S \times A \to \Delta(S)$ is a transition probability, $p_0 \in \Delta(S)$ is an initial state distribution, and $\gamma \in [0, 1]$ is a discount factor. The policy $\pi : S \to \Delta(A)$ is a mapping from state to distribution over actions. While $T(s, a)$ and $\pi(s)$ indicate distributions by definition, we let $T(s'|s, a)$ and $\pi(a|s)$ denote their evaluations for brevity. For the given policy π, the stationary distribution d^π is defined as

$$d^\pi(s,a) = \begin{cases} (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s, a_t = a) & \text{if } \gamma < 1, \\ \lim_{T\to\infty} \frac{1}{T+1} \sum_{t=0}^{T} \Pr(s_t = s, a_t = a) & \text{if } \gamma = 1, \end{cases}$$

where $s_0 \sim p_0$ and $a_t \sim \pi(s_t), s_{t+1} \sim T(s_t, a_t)$ for all time step t. The goal of RL is to learn an optimal policy that maximizes rewards through interactions with the environment: $\max_\pi \mathbb{E}_{(s,a)\sim d^\pi}[R(s, a)]$. The value functions of policy π is defined as $Q^\pi(s, a) := \mathbb{E}_{\pi,M} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|s_0 = s, a_0 = a\right]$ and $V^\pi(s) := \mathbb{E}_{a\sim\pi(s)}[Q^\pi(s, a)]$, where the action-value function Q^π is a unique solution of the Bellman equation:

$$Q^\pi(s,a) = R(s,a) + \gamma \mathbb{E}_{\substack{s'\sim T(s,a) \\ a'\sim\pi(s')}}[Q^\pi(s',a')].$$

In offline RL, the agent optimizes the policy from static dataset $D = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^N$ collected before the training phase. We denote the empirical distribution of the dataset by d^D and will abuse the notation d^D to represent $s \sim d^D$, $(s, a) \sim d^D$, and $(s, a, s') \sim d^D$.

Prior offline model-free RL algorithms, exemplified by (Fujimoto et al., 2019; Kumar et al., 2019; Wu et al., 2019; Lee et al., 2020; Kumar et al., 2020; Nachum et al., 2019b), rely on estimating Q-values for optimizing the target policy. This procedure often yields unreasonably high Q-values due to the compounding error from bootstrapped estimation with out-of-distribution actions sampled from the target policy (Kumar et al., 2019).

3. OptiDICE

In this section, we present *Offline Policy **Opti**mization via Stationary **DI**stribution **C**orrection **E**stimation* (OptiDICE). Instead of the *optimism in the face of uncertainty* principle (Szita & Lörincz, 2008) in online RL, we discourage the uncertainty as in most offline RL algorithms (Kidambi et al., 2020; Yu et al., 2020c); otherwise, the resulting policy may fail to improve on the data-collection policy, or even suffer from severe performance degradation (Petrik et al., 2016; Laroche et al., 2019). Specifically, we consider the regularized policy optimization framework (Nachum et al., 2019b)

$$\pi^* := \arg\max_\pi \mathbb{E}_{(s,a)\sim d^\pi}[R(s,a)] - \alpha D_f(d^\pi || d^D), \quad (1)$$

where $D_f(d^\pi||d^D) := \mathbb{E}_{(s,a)\sim d^D}\left[f\left(\frac{d^\pi(s,a)}{d^D(s,a)}\right)\right]$ is the f-divergence between the stationary distribution d^π and the dataset distribution d^D, and $\alpha > 0$ is a hyperparameter that balances between pursuing the reward-maximization and penalizing the deviation from the distribution of the offline dataset (i.e. penalizing distributional shift). We assume $d^D > 0$ and f being strictly convex and continuously differentiable. Note that we impose regularization in the space of stationary distributions rather than in the space of policies (Wu et al., 2019). However, optimizing for π in (1) involves the evaluation of d^π, which is not directly accessible in the offline RL setting.

To make the optimization tractable, we reformulate (1) in terms of optimizing a stationary distribution $d : S \times A \to \mathbb{R}$. For brevity, we consider discounted MDPs ($\gamma < 1$) and then generalize the result to undiscounted MDPs ($\gamma = 1$). Using d, we rewrite (1) as

$$\max_d \mathbb{E}_{(s,a)\sim d}[R(s,a)] - \alpha D_f(d||d^D) \quad (2)$$

$$\text{s.t. } (\mathcal{B}_* d)(s) = (1-\gamma)p_0(s) + \gamma(\mathcal{T}_* d)(s) \;\; \forall s, \quad (3)$$

$$d(s,a) \geq 0 \;\; \forall s, a, \quad (4)$$

where $(\mathcal{B}_* d)(s) := \sum_{\bar{a}} d(s, \bar{a})$ is a marginalization operator, and $(\mathcal{T}_* d)(s) := \sum_{\bar{s},\bar{a}} T(s|\bar{s}, \bar{a}) d(\bar{s}, \bar{a})$ is a transposed Bellman operator[1]. Note that when $\alpha = 0$, the optimiza-

[1]While AlgaeDICE (Nachum et al., 2019b) also proposes f-divergence-regularized policy optimization as (1), it imposes Bellman flow constraints on state-action pairs, whereas our formulation imposes constraints only on states, which is more natural for finding the optimal policy.

tion (2-4) is exactly the dual formulation of the linear program (LP) for finding an optimal policy of the MDP (Puterman, 1994), where the constraints (3-4) are often called the Bellman flow constraints. Once the optimal stationary distribution d^* is obtained, we can recover the optimal policy π^* in (1) from d^* by $\pi^*(a|s) = \frac{d^*(s,a)}{\sum_{\bar{a}} d^*(s,\bar{a})}$.

We then obtain the following Lagrangian for the constrained optimization problem in (2-4):

$$\max_{d \geq 0} \min_{\nu} \mathbb{E}_{(s,a)\sim d}[R(s,a)] - \alpha D_f(d||d^D) \qquad (5)$$
$$+ \sum_s \nu(s)\Big((1-\gamma)p_0(s) + \gamma(\mathcal{T}_* d)(s) - (\mathcal{B}_* d)(s)\Big),$$

where $\nu(s)$ are the Lagrange multipliers. Lastly, we eliminate the direct dependence on d and $\mathcal{T}_*$ by rearranging the terms in (5) and optimizing the distribution ratio w instead of d:

$$\begin{aligned}
&\mathbb{E}_{(s,a)\sim d}[R(s,a)] - \alpha D_f(d||d^D) \\
&\quad + \sum_s \nu(s)\Big((1-\gamma)p_0(s) + \gamma(\mathcal{T}_* d)(s) - (\mathcal{B}_* d)(s)\Big) \\
&= (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + \mathbb{E}_{(s,a)\sim d^D}\left[-\alpha f\left(\tfrac{d(s,a)}{d^D(s,a)}\right)\right] \qquad (6) \\
&\quad + \sum_{s,a} d(s,a)\underbrace{\Big(R(s,a) + \gamma(\mathcal{T}\nu)(s,a) - (\mathcal{B}\nu)(s,a)\Big)}_{=: e_\nu(s,a) \text{ (`advantage' using } \nu)} \\
&= (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + \mathbb{E}_{(s,a)\sim d^D}\left[-\alpha f\left(\tfrac{d(s,a)}{d^D(s,a)}\right)\right] \\
&\quad + \mathbb{E}_{(s,a)\sim d^D}\Big[\underbrace{\tfrac{d(s,a)}{d^D(s,a)}}_{=: w(s,a)}\big(e_\nu(s,a)\big)\Big] \\
&= (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + \mathbb{E}_{(s,a)\sim d^D}\big[-\alpha f\big(w(s,a)\big)\big] \\
&\quad + \mathbb{E}_{(s,a)\sim d^D}\big[w(s,a)\big(e_\nu(s,a)\big)\big] =: L(w,\nu). \qquad (7)
\end{aligned}$$

The first equality holds due to the property of the adjoint (transpose) operators $\mathcal{B}_*$ and $\mathcal{T}_*$, i.e. for any ν,

$$\sum_s \nu(s)(\mathcal{B}_* d)(s) = \sum_{s,a} d(s,a)(\mathcal{B}\nu)(s,a),$$
$$\sum_s \nu(s)(\mathcal{T}_* d)(s) = \sum_{s,a} d(s,a)(\mathcal{T}\nu)(s,a),$$

where $(\mathcal{T}\nu)(s,a) = \sum_{s'} T(s'|s,a)\nu(s')$ and $(\mathcal{B}\nu)(s,a) = \nu(s)$. Note that $L(w,\nu)$ in (7) does not involve expectation over d, but only expectation over p_0 and d^D, which allows us to perform optimization only with the offline data.

Remark. *The terms in* (7) *will be estimated only by using the samples from the dataset distribution* d^D*:*

$$\hat{L}(w,\nu) := (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] \qquad (8)$$
$$+ \mathbb{E}_{(s,a,s')\sim d^D}\big[-\alpha f\big(w(s,a)\big) + w(s,a)\big(\hat{e}_\nu(s,a,s')\big)\big].$$

Here, $\hat{e}_\nu(s,a,s') := R(s,a) + \gamma\nu(s') - \nu(s)$ *is a single-sample estimation of advantage* $e_\nu(s,a)$*. On the other hand, prior offline RL algorithms often involve estimations using out-of-distribution actions sampled from the target policy, e.g. employing a critic to compute bootstrapped targets for the value function. Thus, our method is free from the compounding error in the bootstrapped estimation due to using out-of-distribution actions.*

In short, OptiDICE solves the problem

$$\max_{w \geq 0} \min_{\nu} L(w,\nu), \qquad (9)$$

where the optimal solution w^* of the optimization (9) represents the stationary distribution corrections between the *optimal* policy's stationary distribution and the dataset distribution: $w^*(s,a) = \frac{d^{\pi^*}(s,a)}{d^D(s,a)}$.

3.1. A closed-form solution

When the state and/or action spaces are large or continuous, it is a standard practice to use function approximators to represent terms such as w and ν, and perform gradient-based optimization of L. However, this could break nice properties for optimizing L, such as concavity in w and convexity in ν, which causes numerical instability and poor convergence for the maximin optimization (Goodfellow et al., 2014). We mitigate this issue by obtaining the closed-form solution of the inner optimization, which reduces the overall problem into a unconstrained convex optimization.

Since the optimization problem (2-4) is an instance of convex optimization, one can easily show that the strong duality holds by the Slater's condition (Boyd et al., 2004). Hence we can reorder the optimization from maximin to minimax:

$$\min_{\nu} \max_{w \geq 0} L(w,\nu). \qquad (10)$$

Then, for any ν, a closed-form solution to the inner maximization of (10) can be derived as follows:

Proposition 1. *The closed-form solution of the inner maximization of* (10)*,* $w_\nu^* := \arg\max_{w\geq 0} L(w,\nu)$*, is*

$$w_\nu^*(s,a) = \max\left(0, (f')^{-1}\left(\frac{e_\nu(s,a)}{\alpha}\right)\right) \quad \forall s,a, \qquad (11)$$

where $(f')^{-1}$ *is the inverse function of the derivative* f' *of* f *and is strictly increasing by strict convexity of* f*. (Proof in Appendix A.)*

A closer look at (11) reveals that that for a fixed ν, the optimal stationary distribution correction $w_\nu^*(s,a)$ is larger for a state-action pair with larger advantage $e_\nu(s,a)$. This solution property has a natural interpretation as follows. As $\alpha \to 0$, the term in (6) becomes the Lagrangian of the primal LP for solving the MDP, where $d(s,a)$ serve as Lagrange multipliers to impose constraints $R(s,a) + \gamma(\mathcal{T}\nu)(s,a) \leq$

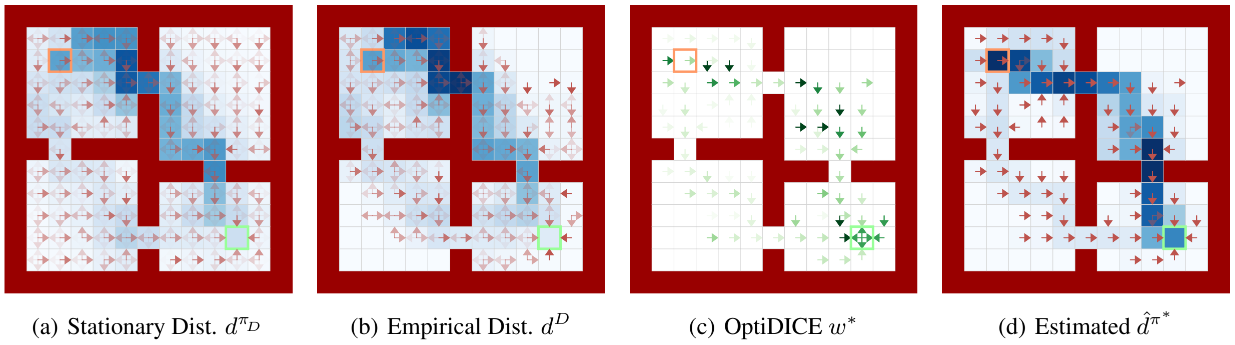

Figure 1. Illustrative example of how OptiDICE estimates the optimal policy's stationary distribution in the Four Rooms domain (Sutton et al., 1999; Nachum et al., 2019b). The initial state and the goal state are denoted by orange and green squares, respectively. Based on a sub-optimal data-collection policy π_D, which induces d^{π_D} shown in (a), a static dataset is sampled and its empirical distribution d^D ($\neq d^{\pi_D}$) shown in (b). The opacity of each square is determined by the state marginals of each stationary distribution, where the opacity of the arrow shows the policy induced by each stationary distribution. By multiplying the OptiDICE w^* obtained by solving (9) (shown in (c)), a near-optimal policy π^* is obtained from $\hat{d}^{\pi^*}(s,a) = d^D(s,a)w^*(s,a)$ shown in (d).

$\nu(s)\ \forall s,a$. Also, each $\nu(s)$ serves as the optimization variable representing the optimal state value function (Puterman, 1994). Thus, $e_{\nu^*}(s,a) = Q^*(s,a) - V^*(s)$, i.e. the advantage function of the *optimal policy*, should be zero for the optimal action while it should be lower for sub-optimal actions. For $\alpha > 0$, the convex regularizer $f\big(\frac{d(s,a)}{d^D(s,a)}\big)$ in (6) relaxes those constraints into soft ones, but it still prefers the actions with higher $e_{\nu^*}(s,a)$ over those with lower $e_{\nu^*}(s,a)$. From this perspective, α adjusts the softness of the constraints $e_\nu(s,a) \leq 0\ \forall s,a$, and f determines the relation between advantages and stationary distribution corrections.

Finally, we reduce the nested optimization in (10) to the following single minimization problem by plugging w_ν^* into $L(w,\nu)$:

$$\min_\nu L(w_\nu^*,\nu) = (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] \tag{12}$$
$$+ \mathbb{E}_{(s,a)\sim d^D}\Big[-\alpha f\Big(\max\Big(0,(f')^{-1}\big(\tfrac{1}{\alpha}e_\nu(s,a)\big)\Big)\Big)\Big]$$
$$+ \mathbb{E}_{(s,a)\sim d^D}\Big[\max\Big(0,(f')^{-1}\big(\tfrac{1}{\alpha}e_\nu(s,a)\big)\Big)\Big(e_\nu(s,a)\Big)\Big].$$

Proposition 2. *$L(w_\nu^*,\nu)$ is convex with respect to ν. (Proof in Appendix B.)*

The minimization of this convex objective can be performed much more reliably than the nested minimax optimization problem. For practical purposes, we use the following objective that can be easily optimized via sampling from D:

$$\tilde{L}(\nu) := (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] \tag{13}$$
$$+ \mathbb{E}_{(s,a,s')\sim d^D}\Big[-\alpha f\Big(\max\Big(0,(f')^{-1}\big(\tfrac{1}{\alpha}\hat{e}_\nu(s,a,s')\big)\Big)\Big)$$
$$+ \max\Big(0,(f')^{-1}\big(\tfrac{1}{\alpha}\hat{e}_\nu(s,a,s')\big)\Big)\Big(\hat{e}_\nu(s,a,s')\Big)\Big].$$

However, careful readers may notice that $\tilde{L}(\nu)$ can be a biased estimate of our target objective $L(w_\nu^*,\nu)$ in (12) due to non-linearity of $(f')^{-1}$ and double-sample problem (Baird, 1995) in $L(w_\nu^*,\nu)$. We justify $\tilde{L}(\nu)$ by formally showing that $\tilde{L}(\nu)$ is the upper bound of $L(w_\nu^*,\nu)$:

Corollary 3. *$\tilde{L}(\nu)$ in (13) is an upper bound of $L(w_\nu^*,\nu)$ in (12), i.e. $L(w_\nu^*,\nu) \leq \tilde{L}(\nu)$ always holds, where equality holds when the MDP is deterministic. (Proof in Appendix B.)*

Illustrative example *Figure 1* outlines how our approach works in the Four Rooms domain (Sutton et al., 1999) where the agent aims to navigate to a goal location in a maze composed of four rooms. We collected static dataset D consisting of 50 episodes with maximum time step 50 using the data-collection policy $\pi_D = 0.5\pi^*_{\text{true}} + 0.5\pi_{\text{rand}}$, where π^*_{true} is the optimal policy of the underlying true MDP and π_{rand} is the random policy sampled from the Dirichlet distribution, i.e. $\pi_{\text{rand}}(s) \sim \text{Dir}(1,1,1,1)\ \forall s$.

In this example, we explicitly constructed Maximum Likelihood Estimate (MLE) MDP $\hat{M}$ based on the static dataset D. We then obtained ν^* by minimizing (12) where $\hat{M}$ was used to exactly compute $e_\nu(s,a)$. Then, $w^*_{\nu^*}$ was estimated directly via Eq. (11) (*Figure 1(c)*). Finally, w^* was multiplied by d^D to correct d^D towards an optimal policy, resulting in $\hat{d}^{\pi^*}$, which is the stationary distribution of the estimated optimal policy (*Figure 1(d)*). For tabular MDPs, the global optima (ν^*, w^*) can always be obtained. We describe these experiments on OptiDICE for finite MDPs in Appendix C.

3.2. Stationary distribution correction estimation with function approximation

Based on the results from the previous section, we assume that ν and w are parameterized by θ and ϕ, respectively, and that both models are sufficiently expressive, e.g. using deep neural networks. Using these models, we optimize θ by

$$\min_\theta J_\nu(\theta) := \min_\theta \tilde{L}(\nu_\theta). \tag{14}$$

After obtaining the optimizing solution θ^*, we need a way to evaluate $w^*(s,a) = \frac{d^{\pi^*}(s,a)}{d^D(s,a)}$ for any (s,a) to finally obtain the optimal policy π^*. However, the closed-form solution $w^*_{\nu_\theta}(s,a)$ in **Proposition 1** can be evaluated only on (s,a) in D since it requires both $R(s,a)$ and $\mathbb{E}_{s' \sim T(s,a)}[\nu_\theta(s')]$ to evaluate the advantage e_ν. Therefore, we use a parametric model e_ϕ that approximates the advantage inside the analytic formula presented in (11), so that

$$w_\phi(s,a) := \max\left(0, (f')^{-1}\left(\frac{e_\phi(s,a)}{\alpha}\right)\right). \tag{15}$$

We consider two options to optimize ϕ once we obtain ν_θ from Eq. (14). First, ϕ can be optimized via

$$\min_\phi J_w(\phi;\theta) := \min_\phi -\hat{L}(w_\phi, \nu_\theta) \tag{16}$$

which corresponds to solving the original minimax problem (10). We also consider

$$\begin{aligned}&\min_\phi J_w^{\text{MSE}}(\phi;\theta) \\ &:= \min_\phi \mathbb{E}_{(s,a,s') \sim d^D}\left[\left(e_\phi(s,a) - \hat{e}_{\nu_\theta}(s,a,s')\right)^2\right], \end{aligned} \tag{17}$$

which minimizes the mean squared error (MSE) between the advantage $e_\phi(s,a)$ and the target induced by ν_θ. We observe that using either J_w or J_w^{MSE} works effectively, which will be detailed in our experiments. In our implementation, we perform joint training of θ and ϕ, rather than optimizing ϕ after convergence of θ.

3.3. Policy extraction

As the last step, we need to extract the optimal policy π^* from the optimal stationary distribution corrections $w_\phi(s,a) = \frac{d^{\pi^*}(s,a)}{d^D(s,a)}$. While the optimal policy can be easily obtained by $\pi^*(a|s) = \frac{d^D(s,a) w_\phi(s,a)}{\sum_{\bar{a}} d^D(s,\bar{a}) w_\phi(s,\bar{a})}$ for tabular domains, this procedure is not straightforwardly applicable to continuous domains.

One of the ways to address continuous domains is to use importance-weighted behavioral cloning: we optimize the parameterized policy π_ψ by maximizing the log-likelihood on (s,a) that would be sampled from the optimal policy π^*:

$$\begin{aligned}&\max_\psi \mathbb{E}_{(s,a) \sim d^{\pi^*}}[\log \pi_\psi(a|s)] \\ &\qquad = \max_\psi \mathbb{E}_{(s,a) \sim d^D}\left[w_\phi(s,a) \log \pi_\psi(a|s)\right].\end{aligned}$$

Despite its simplicity, this approach does not work well in practice, since π_ψ will be trained only on samples from the intersection of the supports of d^{π^*} and d^D, which becomes very scarce when π^* deviates significantly from the data collection policy π_D.

We thus use the information projection (I-projection) for training the policy:

$$\min_\psi \mathbb{KL}\left(d^D(s)\pi_\psi(a|s) || d^D(s)\pi^*(a|s)\right), \tag{18}$$

where we replace $d^{\pi^*}(s)$ by $d^D(s)$ for $d^{\pi^*}(s,a)$. This results in minimizing the discrepancy between $\pi_\psi(a|s)$ and $\pi^*(a|s)$ on the stationary distribution over states from π_D. This approach is motivated by the desideratum that the policy π_ψ should be trained at least on the states observed in D to be robust upon deployment. Now, rearranging the terms in (18), we obtain

$$\begin{aligned}&\mathbb{KL}\left(d^D(s)\pi_\psi(a|s) || d^D(s)\pi^*(a|s)\right) \\ &= -\mathbb{E}_{\substack{s \sim d^D \\ a \sim \pi_\psi(s)}}\Bigg[\log \underbrace{\frac{d^*(s,a)}{d^D(s,a)}}_{= w_\phi(s,a)} - \log \frac{\pi_\psi(a|s)}{\pi_D(a|s)} - \underbrace{\log \frac{d^*(s)}{d^D(s)}}_{\text{constant for } \pi}\Bigg] \\ &= -\mathbb{E}_{\substack{s \sim d^D \\ a \sim \pi_\psi(s)}}\left[\log w_\phi(s,a) - \mathbb{KL}(\pi_\psi(\bar{a}|s) || \pi_D(\bar{a}|s))\right] + C \\ &=: J_\pi(\psi; \phi, \pi_D)\end{aligned} \tag{19}$$

We can interpret this I-projection objective as a KL-regularized actor-critic architecture (Fox et al., 2016; Schulman et al., 2017), where $\log w_\phi(s,a)$ taking the role of the critic and π_ψ being the actor[2]. Note that I-projection requires us to evaluate π_D for the KL regularization term. For this, we employ another parameterized policy π_β to approximate π_D, trained via simple behavioral cloning (BC).

3.4. Generalization to $\gamma = 1$

For $\gamma = 1$, our original problem (2-4) for the stationary distribution d is an ill-posed problem: for any d that satisfies the Bellman flow constraints (3-4) and a constant $c \geq 0$, cd also satisfies the Bellman flow constraints (3-4) (Zhang et al., 2020a). We address this issue by adding additional normalization constraint $\sum_{s,a} d(s,a) = 1$ to (2-4). By using analogous derivation from (2) to (10) with the normalization constraint—introducing a Lagrange multiplier $\lambda \in \mathbb{R}$ and changing the variable d to w—we obtain the following minimax objective for w, ν and λ:

$$\begin{aligned}&\min_{\nu,\lambda} \max_{w \geq 0} L(w,\nu,\lambda) \\ &\quad := L(w,\nu) + \lambda(1 - \mathbb{E}_{(s,a) \sim d^D}[w(s,a)]) \\ &\quad = (1-\gamma)\mathbb{E}_{s \sim p_0}[\nu(s)] + \mathbb{E}_{(s,a) \sim d^D}[-\alpha f(w(s,a))] \\ &\qquad - \mathbb{E}_{(s,a) \sim d^D}[w(s,a)(e_\nu(s,a) - \lambda))] + \lambda.\end{aligned} \tag{20}$$

[2]When $f(x) = x \log x$ (i.e. KL-divergence), $(f')^{-1} = \exp(x-1)$, and we have $\log w_{\nu^*}(s,a) = \frac{1}{\alpha} e_{\nu^*}(s,a) - 1$ by Eq. (11). Given that $e_{\nu^*}(s,a)$ represents an approximately optimal advantage $A^*(s,a) \approx Q^*(s,a) - V^*(s)$ (Section 3.1), the policy extraction via I-projection (19) corresponds to a KL-regularized policy optimization: $\max_\pi \mathbb{E}_{a \sim \pi}[\frac{1}{\alpha} A^*(s,a) - KL(\pi(\bar{a}|s) || \pi_D(\bar{a}|s))]$.

Similar to (8), we define $\hat{L}(w, \nu, \lambda)$, an unbiased estimator for $L(w, \nu, \lambda)$ such that

$$\hat{L}(w, \nu, \lambda) := (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + \lambda \quad (21)$$
$$+ \mathbb{E}_{(s,a,s')\sim d^D}\left[-\alpha f\big(w(s,a)\big) + w(s,a)\big(\hat{e}_{\nu,\lambda}(s,a,s')\big)\right],$$

where $\hat{e}_{\nu,\lambda}(s,a,s') := \hat{e}_\nu(s,a,s') - \lambda$. We then derive a closed-form solution for the inner maximization in (20):

Proposition 4. *The maximizer $w^*_{\nu,\lambda} : S \times A \to \mathbb{R}$ of the inner optimization of (20), which is defined by $w^*_{\nu,\lambda} := \arg\max_{w\geq 0} L(w,\nu,\lambda)$, is*

$$w^*_{\nu,\lambda}(s,a) = \max\left(0, (f')^{-1}\left(\frac{e_\nu(s,a) - \lambda}{\alpha}\right)\right).$$

(Proof in Appendix D.)

Similar to (13), we minimize the biased estimate $\tilde{L}(\nu, \lambda)$, which is an upper bound of $L(w^*_{\nu,\lambda}, \nu, \lambda)$, by applying the closed-form solution from **Proposition 4**:

$$\tilde{L}(\nu, \lambda) := (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] \quad (22)$$
$$+ \mathbb{E}_{(s,a,s')\sim d^D}\Bigg[-\alpha f\Big(\max\big(0, (f')^{-1}\big(\tfrac{1}{\alpha}\hat{e}_{\nu,\lambda}(s,a,s')\big)\big)\Big)$$
$$+ \max\Big(0, (f')^{-1}\big(\tfrac{1}{\alpha}\hat{e}_{\nu,\lambda}(s,a,s')\big)\Big)\big(\hat{e}_{\nu,\lambda}(s,a,s')\big)\Bigg] + \lambda.$$

By using the above estimators, we correspondingly update our previous objectives for θ and ϕ as follows. First, the objective for θ is modified to

$$\min_\theta J_\nu(\theta, \lambda) := \min_\theta \tilde{L}(\nu_\theta, \lambda). \quad (23)$$

For ϕ, we modify our approximator in (15) by including the Lagrangian $\lambda' \in \mathbb{R}$:

$$w_{\phi,\lambda'}(s,a) := \max\left(0, (f')^{-1}\left(\frac{e_\phi(s,a) - \lambda'}{\alpha}\right)\right).$$

Note that $\lambda' \neq \lambda$ is used to stabilize the learning process. For optimizing over ϕ, the minimax objective (16) is modified as

$$\min_\phi J_w(\phi, \lambda'; \theta) := \min_\phi -\hat{L}(w_{\phi,\lambda'}, \nu_\theta, \lambda'), \quad (24)$$

while the same objective $J_w^{\text{MSE}}(\phi; \theta)$ in (17) is used for the MSE objective. We additionally introduce learning objectives for λ and λ', which is required for the normalization constraint discussed in this subsection:

$$\min_\lambda J_\nu(\theta, \lambda) \quad \text{and} \quad \min_{\lambda'} J_w(\phi, \lambda'; \theta). \quad (25)$$

Finally, by using the above objectives in addition to BC objective and policy extraction objective in (19), we describe our algorithm, OptiDICE, in **Algorithm 1**, where we train

Algorithm 1 OptiDICE

```
Input: A dataset $D := \{(s_i, a_i, r_i, s'_i)\}_{i=1}^N$, a set of initial
    states $D_0 := \{s_{0,i}\}_{i=1}^{N_0}$, neural networks $\nu_\theta$ and $e_\phi$
    with parameters $\theta$ and $\phi$, learnable parameters $\lambda$ and $\lambda'$,
    policy networks $\pi_\beta$ and $\pi_\psi$ with parameter $\beta$ and $\psi$, a
    learning rate $\eta$
1: for each iteration do
2:     Sample mini-batches from $D$ and $D_0$, respectively.
3:     Compute $\theta$-gradient to optimize (23):
           $g_\theta \approx \nabla_\theta J_\nu(\theta, \lambda)$
4:     Compute $\phi$-gradient for either one of objectives:
           $g_\phi \approx \nabla_\phi J_w(\phi, \lambda'; \theta)$      (minimax obj. (24))
           $g_\phi \approx \nabla_\phi J_w^{\text{MSE}}(\phi; \theta)$      (MSE obj. (17))
5:     Compute $\lambda$ and $\lambda'$ gradients to optimize (25):
           $g_\lambda \approx \nabla_\lambda J_\nu(\theta, \lambda), g_{\lambda'} \approx \nabla_{\lambda'} J_w(\phi, \lambda'; \theta)$
6:     Compute $\beta$-gradient $g_\beta$ for BC.
7:     Compute $\psi$-gradient via (19) (policy extraction):
           $g_\psi \approx \nabla_\psi J_\pi(\psi; \phi, \pi_\beta)$
8:     Perform SGD updates:
           $\theta \leftarrow \theta - \eta g_\theta, \quad \lambda \leftarrow \lambda - \eta g_\lambda, \quad \beta \leftarrow \beta - \eta g_\beta,$
           $\phi \leftarrow \phi - \eta g_\phi, \quad \lambda' \leftarrow \lambda' - \eta g_{\lambda'}, \quad \psi \leftarrow \psi - \eta g_\psi.$
9: end for
Output: $\nu_\theta \approx \nu^*$, $w_{\phi,\lambda'} \approx w^*$, $\pi_\psi \approx \pi^*$,
```

neural network parameters via stochastic gradient descent. In our algorithm, we use a warm-up iteration—optimizing all networks except π_ψ—to prevent π_ψ from its converging to sub-optimal policies during its initial training. In addition, we empirically observed that using the normalization constraint stabilizes OptiDICE's learning process even for $\gamma < 1$, thus we used the normalization constraint in all experiments (Zhang et al., 2020a).

4. Experiments

In this section, we evaluate OptiDICE for both tabular and continuous MDPs. For the f-divergence, we chose $f(x) = \frac{1}{2}(x-1)^2$, i.e. χ^2-divergence for the tabular-MDP experiment, while we use its softened version for continuous MDPs (See Appendix E for details).

4.1. Random MDPs (tabular MDPs)

We validate tabular OptiDICE's efficiency and robustness using randomly generated MDPs by following the experimental protocol from Laroche et al. (2019) and Lee et al. (2020) (See Appendix F.1.). We consider a data-collection policy π_D characterized by the behavior optimality parameter ζ that relates to π_D's performance $\zeta V^*(s_0) + (1-\zeta)V^{\pi_{\text{unif}}}(s_0)$ where π_{unif} denotes the uniformly random policy. We eval-

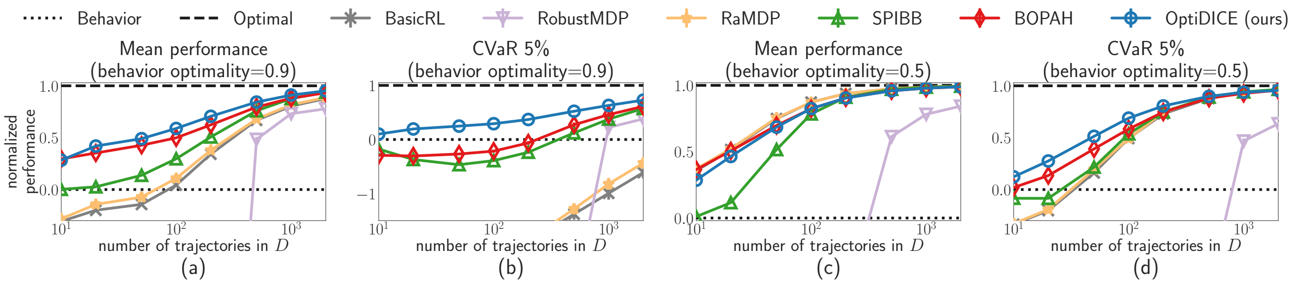

Figure 2. Performance of tabular OptiDICE and baseline algorithms in random MDPs. For baselines, we use *BasicRL* (a model-based RL computing an optimal policy via MLE MDP), *Robust MDP* (Nilim & El Ghaoui, 2005; Iyengar, 2005), *Reward-adjusted MDP (RaMDP)* (Petrik et al., 2016), *SPIBB* (Laroche et al., 2019), *BOPAH* (Lee et al., 2020). For varying numbers of trajectories and two types of data-collection policies ($\zeta = 0.9, 0.5$), the mean and the 5%-CVaR of normalized performances for 10,000 runs are reported with 95% confidence intervals. OptiDICE performs better than (for $\zeta = 0.9$) or on par with (for $\zeta = 0.5$) the baselines in the mean performance measure, while always outperforming the baselines in the CVaR performance measure.

uate each algorithm in terms of the normalized performance of the policy π, given by $(V^*(s_0) - V^{\pi_D}(s_0))/(V^{\pi}(s_0) - V^{\pi_D}(s_0))$, which intuitively measures the performance enhancement of π over π_D. Each algorithm is tested for 10,000 runs, and their mean and 5% conditional value at risk (5%-CVaR) are reported, where the mean of the worst 500 runs is considered for 5%-CVaR. Note that CVaR implicitly stands for the robustness of each algorithm.

We describe the performance of tabular OptiDICE and baselines in *Figure 2*. For $\zeta = 0.9$, where π_D is near-deterministic and thus d^D's support is relatively small, OptiDICE outperforms the baselines in both mean and CVaR (*Figure 2(a),(b)*). For $\zeta = 0.5$, where π_D is highly stochastic and thus d^D's support is relatively large, OptiDICE outperforms the baselines in CVaR, while performing competitively in mean. In summary, OptiDICE was more sample-efficient and stable than the baselines.

4.2. D4RL benchmark (continuous control tasks)

We evaluate OptiDICE in continuous MDPs using D4RL offline RL benchmarks (Fu et al., 2021). We use Maze2D (3 tasks) and Gym-MuJoCo (12 tasks) domains from the D4RL dataset (See Appendix F.2 for task description). We interpret terminal states as absorbing states and use the absorbing-state implementation proposed by Kostrikov et al. (2019a). For obtaining π_β discussed in Section 3.3, we use the tanh-squashed *mixture* of Gaussians policy π_β to embrace the multi-modality of data collected from heterogeneous policies. For the target policy π_ψ, we use a tanh-squashed Gaussian policy, following conservative Q Learning (CQL) (Kumar et al., 2020)—the state-of-the-art model-free offline RL algorithm. We provide detailed information of the experimental setup in Appendix F.2.

The normalized performance of OptiDICE and the best model-free algorithm for each domain is presented in *Table 1*, and learning curves for CQL and OptiDICE are shown in *Figure 3*, where $\gamma = 0.99$ used for all algorithms. Most notably, OptiDICE achieves state-of-the-art performance for all tasks in the Maze2D domain, by a large margin. In Gym-MuJoCo domain, OptiDICE achieves the best mean performance for 4 tasks (hopper-medium, hopper-medium-expert, walker2d-random, and halfcheetah-medium-expert). Another noteworthy observation is that OptiDICE overwhelmingly outperforms AlgaeDICE (Nachum et al., 2019b) in all domains (*Table 1* and *Table 3* in Appendix for detailed

Table 1. Normalized performance of OptiDICE compared with the best model-free baseline in the D4RL benchmark tasks (Fu et al., 2021). In the *Best baseline* column, the algorithm with the best performance among 8 algorithms (offline SAC (Haarnoja et al., 2018), BEAR (Kumar et al., 2019), BRAC (Wu et al., 2019), AWR (Peng et al., 2019), cREM (Agarwal et al., 2020), BCQ (Fujimoto et al., 2019), AlgaeDICE (Nachum et al., 2019b), CQL (Kumar et al., 2020)) is presented, taken from (Fu et al., 2021). OptiDICE achieved highest scores in 7 tasks.

D4RL Task	Best baseline	OptiDICE
maze2d-umaze	88.2$^{\text{Offline SAC}}$	**111.0**
maze2d-medium	33.8$^{\text{BRAC-v}}$	**145.2**
maze2d-large	40.6$^{\text{BRAC-v}}$	**155.7**
hopper-random	**12.2**$^{\text{BRAC-v}}$	11.2
hopper-medium	58.0$^{\text{CQL}}$	**94.1**
hopper-medium-replay	**48.6**$^{\text{CQL}}$	36.4
hopper-medium-expert	110.9$^{\text{BCQ}}$	**111.5**
walker2d-random	7.3$^{\text{BEAR}}$	**9.9**
walker2d-medium	**81.1**$^{\text{BRAC-v}}$	21.8
walker2d-medium-replay	**26.7**$^{\text{CQL}}$	21.6
walker2d-medium-expert	**111.0**$^{\text{CQL}}$	74.8
halfcheetah-random	**35.4**$^{\text{CQL}}$	11.6
halfcheetah-medium	**46.3**$^{\text{BRAC-v}}$	38.2
halfcheetah-medium-replay	**47.7**$^{\text{BRAC-v}}$	39.8
halfcheetah-medium-expert	64.7$^{\text{BCQ}}$	**91.1**

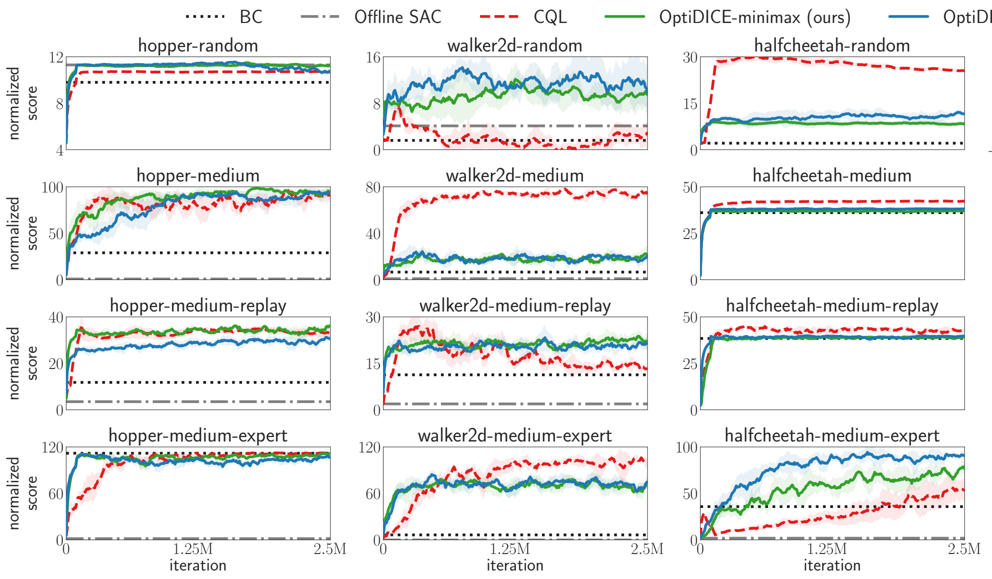

Figure 3. Performance of BC, offline SAC (Haarnoja et al., 2018), CQL (Kumar et al., 2020), OptiDICE-minimax (= OptiDICE with minimax objective in (16)) and OptiDICE-MSE (= OptiDICE with MSE objective in (17)) on D4RL benchmark (Fu et al., 2021) for $\gamma = 0.99$. For BC and offline SAC, we use the result reported in D4RL paper (Fu et al., 2021). For CQL and OptiDICE, we provide learning curves for each algorithm where the policy is optimized during 2,500,000 iterations. For CQL, we use the original code by authors with hyperparameters reported in the CQL paper (Kumar et al., 2020). OptiDICE strictly outperforms CQL on 6 tasks, while performing on par with CQL on 4 tasks. We report mean scores and their 95% confidence intervals obtained from 5 runs for each task.

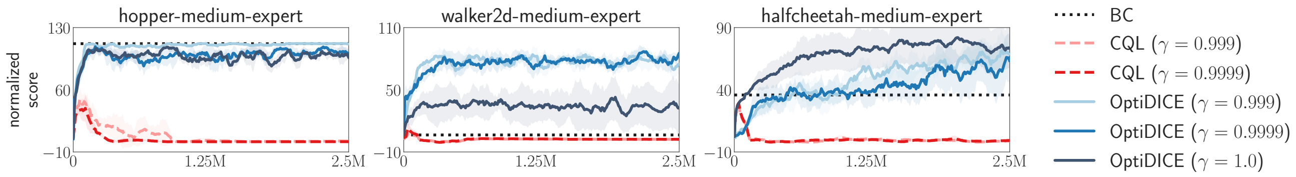

Figure 4. Performance of BC, CQL and OptiDICE (with MSE objective in (17)) in D4RL benchmark (Fu et al., 2021) for $\gamma = 0.999, 0.9999$ and 1.0, where the hyperparameters other than γ are the same as those in *Figure 3*.

performance of AlgaeDICE), although both AlgaeDICE and OptiDICE stem from the same objective in (1). This is because AlgaeDICE optimizes a nested max-min-max problem, which can suffer from severe overestimation by using out-of-distribution actions and numerical instability. In contrast, OptiDICE solves a simpler minimization problem and does not rely on out-of-distribution actions, exhibiting stable optimization.

As discussed in Section 3.4, OptiDICE can naturally be generalized to undiscounted problems ($\gamma = 1$). In *Figure 4*, we vary $\gamma \in \{0.999, 0.9999, 1.0\}$ to validate OptiDICE's robustness in γ by comparing with CQL in {hopper-medium-expert, walker2d-medium-expert, halfcheetah-medium-expert} (See Appendix G for the results for other tasks). The performance of OptiDICE stays stable, while CQL easily becomes unstable as γ increases, due to the divergence of Q-function. This is because OptiDICE uses *normalized* stationary distribution corrections, whereas CQL learns the action-value function whose values becomes unbounded as γ gets close to 1, resulting in numerical instability.

5. Discussion

Current DICE algorithms except for AlgaeDICE (Nachum et al., 2019b) only deal with either policy evaluation (Nachum et al., 2019a; Zhang et al., 2020a;b;b; Yang et al., 2020; Dai et al., 2020) or imitation learning (Kostrikov et al., 2019b), not policy optimization.

Although both AlgaeDICE (Nachum et al., 2019b) and OptiDICE aim to solve f-divergence regularized RL, each algorithm solves the problem in a different way. AlgaeDICE relies on off-policy evaluation (OPE) of the intermediate policy π via DICE (inner $\min_\nu \max_w$ of Eq. (26)), and then optimizes π via policy-gradient upon the OPE result (outer $\max_\pi$ of Eq. (26)), yielding an overall $\max_\pi \min_\nu \max_w$ problem of Eq. (26). Although the actual AlgaeDICE implementation employs an additional approximation for practical optimization, i.e. using Eq. (28) that removes the inner-

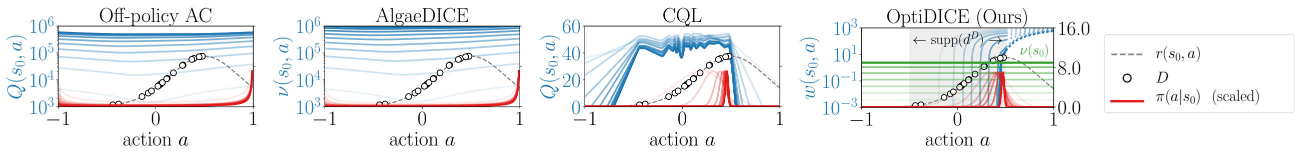

Figure 5. Illustration on overestimation. $r(s_0,a)$ is a ground-truth reward, D is a sampled offline dataset, and $\pi(a|s_0)$ is the policy density normalized by its maximum.

AlgaeDICE $\left(e_\nu^\pi(s,a) := r(s,a) + \gamma \mathbb{E}_{s'\sim T(s,a), a'\sim\pi(s')}[\nu(s',a')] - \nu(s,a),\ \ \hat{e}_\nu(s,a,s',a') := r(s,a) + \gamma\nu(s',a') - \nu(s,a)\right)$

$$\max_\pi \min_\nu \max_w \mathbb{E}_{(s,a)\sim d^D}\left[e_\nu^\pi(s,a)w(s,a) - \alpha f\big(w(s,a)\big)\right] + (1-\gamma)\mathbb{E}_{s_0\sim p_0, a_0\sim\pi(s_0)}\left[\nu(s_0,a_0)\right] \tag{26}$$

$$= \max_\pi \min_\nu \alpha \mathbb{E}_{(s,a)\sim d^D}\left[f_*\big(\tfrac{1}{\alpha}e_\nu^\pi(s,a)\big)\right] + (1-\gamma)\mathbb{E}_{s_0\sim p_0, a_0\sim\pi(s_0)}\left[\nu(s_0,a_0)\right] \tag{27}$$

$$\approx \max_\pi \min_\nu \alpha \mathbb{E}_{(s,a,s')\sim d^D, a'\sim\pi(s')}\left[f_*\big(\tfrac{1}{\alpha}\hat{e}_\nu(s,a,s',a')\big)\right] + (1-\gamma)\mathbb{E}_{s_0\sim p_0, a_0\sim\pi(s_0)}\left[\nu(s_0,a_0)\right] \tag{28}$$

OptiDICE $\left(e_\nu(s,a) := r(s,a) + \gamma\mathbb{E}_{s'\sim T(s,a)}[\nu(s')] - \nu(s),\ \ \hat{e}_\nu(s,a,s') := r(s,a) + \gamma\nu(s') - \nu(s), x_+ := \max(0,x)\right)$

$$\min_\nu \max_{w\geq 0} \mathbb{E}_{(s,a)\sim d^D}\left[e_\nu(s,a)w(s,a) - \alpha f\big(w(s,a)\big)\right] + (1-\gamma)\mathbb{E}_{s_0\sim p_0}[\nu(s_0)] \tag{29}$$

$$= \min_\nu \mathbb{E}_{(s,a)\sim d^D}\left[e_\nu(s,a)(f')^{-1}\big(\tfrac{1}{\alpha}e_\nu(s,a)\big)_+ - \alpha f\big((f')^{-1}\big(\tfrac{1}{\alpha}e_\nu(s,a)\big)_+\big)\right] + (1-\gamma)\mathbb{E}_{s_0\sim p_0}[\nu(s_0)] \tag{30}$$

$$\approx \min_\nu \mathbb{E}_{(s,a,s')\sim d^D}\left[\hat{e}_\nu(s,a,s')(f')^{-1}\big(\tfrac{1}{\alpha}\hat{e}_\nu(s,a,s')\big)_+ - \alpha f\big((f')^{-1}(\tfrac{1}{\alpha}\hat{e}_\nu(s,a,s'))_+\big)\right] + (1-\gamma)\mathbb{E}_{s_0\sim p_0}[\nu(s_0)] \tag{31}$$

most $\max_w$ via convex conjugate and uses a biased estimation of $f_*(\mathbb{E}_{s',a'}[\cdot])$ via $\mathbb{E}_{s',a'}[f_*(\cdot)]$, it still involves nested $\max_\pi \min_\nu$ optimization, susceptible to instability. In contrast, OptiDICE directly estimates the stationary distribution corrections of the optimal policy, resulting in $\min_\nu \max_w$ problem of Eq. (29). In addition, our implementation performs the single minimization of Eq. (31) (the biased estimate of $\min_\nu$ of (30)), which greatly improves the stability of overall optimization.

To see this, we conduct single-state MDP experiments, where $S = \{s_0\}$ is the state space, $A = [-1,1]$ is the action space, $T(s_0|s_0,a) = 1$ is the transition dynamics, $r(s_0,a)$ is a reward function, $\gamma = 0.9$, and D is the offline dataset. The blue lines in the figures present the estimates learned by each algorithm (i.e. Q, ν, w) (Darker colors mean later iterations). Similarly, the red lines visualize the action densities from intermediate policies. In this example, a vanilla off-policy actor-critic (AC) method suffers from the divergence of Q-values due to its TD target being outside the data distribution d^D. This makes the policy learn toward unreasonably high Q-values outside d^D. AlgaeDICE with Eq. (28) is no better for small α. CQL addresses this issue by lowering the Q-values outside d^D. Finally, OptiDICE computes optimal stationary distribution corrections $w(s,a) = \frac{d^{\pi^*}(s,a)}{d^D(s,a)}$ by Eq. (31) and Eq. (29) ($\max_w(\cdot)$ for ν^*). Then, the policy $\pi(a|s) \propto w(s,a)d^D(s,a)$ is extracted, automatically ensuring actions to be selected within the support of d^D (supp(d^D)).

Also, note that Eq. (31) of OptiDICE is *unbiased* (i.e., (31) = (30)) if T is deterministic (**Corollary 3**). In contrast, Eq. (28) of AlgaeDICE *is always biased* (i.e., (27) $\neq$ (28)) even for the deterministic T, due to its dependence on expectation w.r.t. π. Our biased objective of Eq. (31) removes the need for double sampling in Eq. (30).

6. Conclusion

We presented OptiDICE, an offline RL algorithm that aims to estimate stationary distribution corrections between the *optimal* policy's stationary distribution and the dataset distribution. We formulated the estimation problem as a minimax optimization that does not involve sampling from the target policy, which essentially circumvents the overestimation issue incurred by bootstrapped target with out-of-distribution actions, practiced by most model-free offline RL algorithms. Then, deriving the closed-form solution of the inner optimization, we simplified the nested minimax optimization for obtaining the optimal policy to a convex minimization problem. In the experiments, we demonstrated that OptiDICE performs competitively with the state-of-the-art offline RL baselines.

Acknowledgements

This work was supported by the National Research Foundation (NRF) of Korea (NRF-2019M3F2A1072238 and NRF-2019R1A2C1087634), and the Ministry of Science and Information communication Technology (MSIT) of Korea (IITP No. 2019-0-00075, IITP No. 2020-0-00940 and IITP No. 2017-0-01779 XAI). We also acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Canadian Institute of Advanced Research (CIFAR).

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Baird, L. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, 1995.

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

Dai, B., Nachum, O., Chow, Y., Li, L., Szepesvari, C., and Schuurmans, D. CoinDICE: Off-policy confidence interval estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.

Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 2005.

Fox, R., Pakman, A., and Tishby, N. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2016.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4RL: Datasets for deep data-driven reinforcement learning, 2021. URL https://openreview.net/forum?id=px0-N3_KjA.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2672–2680, 2014.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

Iyengar, G. N. Robust dynamic programming. *Mathematics of Operations Research*, 2005.

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, 2019.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. MOReL : Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 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. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019a.

Kostrikov, I., Nachum, O., and Tompson, J. Imitation learning via off-policy distribution matching. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019b.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

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 (NeurIPS)*, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Lange, S., Gabel, T., and Riedmiller, M. *Reinforcement learning: State-of-the-art*. Springer Berlin Heidelberg, 2012.

Laroche, R., Trichelair, P., and Des Combes, R. T. Safe policy improvement with baseline bootstrapping. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Lee, B.-J., Lee, J., Vrancx, P., Kim, D., and Kim, K.-E. Batch reinforcement learning with hyperparameter gradients. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 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. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, 2016.

Nachum, O., Chow, Y., Dai, B., and Li, L. DualDICE: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019a.

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

Nilim, A. and El Ghaoui, L. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 2005.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019.

Petrik, M., Ghavamzadeh, M., and Chow, Y. Safe policy improvement by minimizing robust baseline regret. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Puterman, M. L. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., 1st edition, 1994.

Schulman, J., Chen, X., and Abbeel, P. Equivalence between policy gradients and soft Q-learning, 2017.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT Press, 1998.

Sutton, R. S., Precup, D., and Singh, S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999.

Szita, I. and Lörincz, A. The many faces of optimism: A unifying approach. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning, 2019.

Yang, M., Nachum, O., Dai, B., Li, L., and Schuurmans, D. Off-policy evaluation via the regularized lagrangian. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yu, C., Liu, J., and Nemati, S. Reinforcement learning in healthcare: A survey, 2020a.

Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., and Darrell, T. BDD100K: A diverse driving dataset for heterogeneous multitask learning. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020b.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J., Levine, S., Finn, C., and Ma, T. MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020c.

Zhang, R., Dai, B., Li, L., and Schuurmans, D. GenDICE: Generalized offline estimation of stationary values. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020a.

Zhang, S., Liu, B., and Whiteson, S. GradientDICE: Rethinking generalized offline estimation of stationary values. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2020b.