DEMODICE: OFFLINE IMITATION LEARNING WITH SUPPLEMENTARY IMPERFECT DEMONSTRATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider offline imitation learning (IL), which aims to mimic the expert's behavior from its demonstration without further interaction with the environment. One of the main challenges in offline IL is to deal with the narrow support of the data distribution exhibited by the expert demonstrations that cover only a small fraction of the state and the action spaces. As a result, offline IL algorithms that rely only on expert demonstrations are very unstable since the situation easily deviates from those in the expert demonstrations. In this paper, we assume additional demonstration data of unknown degrees of optimality, which we call imperfect demonstrations. Under this setting, we propose DemoDICE, which effectively utilizes imperfect demonstrations by matching the stationary distribution of a policy with experts' distribution while penalizing its deviation from the overall demonstrations. Compared with the recent IL algorithms that adopt adversarial minimax training objectives, we substantially stabilize overall learning process by reducing minimax optimization to a direct convex optimization in a principled manner. Using extensive tasks, we show that DemoDICE achieves promising results in the offline IL from expert and imperfect demonstrations.

1 INTRODUCTION

Reinforcement learning (RL) (Sutton et al., 1998) aims to learn a intelligent behavioral strategy based on reward feedback. Although RL has achieved remarkable success in many challenging domains, its practicality and applicability are still limited in two respects. First, we need to specify the reward function, which may be non-trivial to do so in many real-world problems that require complex decision making. Second, the standard RL setting assumes online interaction with the environment during the intermediate stages of learning, which is infeasible for mission-critical tasks.

Imitation learning (IL) (Pomerleau, 1991; Ng & Russell, 2000) addresses the first limitation of RL, where the agent is trained to mimic the expert from demonstration instead of specifying the reward function. It is well known that adopting supervised learning for training the imitating agent, commonly referred to as behavioral cloning (BC), is vulnerable to the distribution drift (Ross et al., 2011). Thus, most of the successful IL algorithms rely on online experiences collected from the environment by executing intermediate policies during training. Recent progress made by adversarial imitation learning (AIL) (Ho & Ermon, 2016; Ke et al., 2019; Kostrikov et al., 2020) achieving state-of-the-art results on challenging imitation tasks still relies on such an online training paradigm.

Unfortunately, in many realistic tasks such as robotic manipulation and autonomous driving, online interactions are either costly or dangerous. Offline RL (Fujimoto et al., 2019; Kumar et al., 2019; 2020; Levine et al., 2020; Wang et al., 2020; Lee et al., 2021; Kostrikov et al., 2021) aims to address these concerns by training the agent from the pre-collected set of experiences without online interactions. To prevent an issue caused by the distributional shift, offline RL algorithms mitigate the phenomenon by constraining the shift or making a conservative evaluation of the policy being learned. A representative example is OptiDICE (Lee et al., 2021), which performs a simple convex optimization. While OptiDICE achieves promising performance in offline RL benchmarks, this promise is conditioned on the careful choice of an f-divergence as a regularizer; using the KL here may incur the numerical instability of OptiDICE.

In this paper, we are concerned with offline IL problems. Finding an effective algorithm for these problems is tricky. For instance, naively extending the offline RL algorithms (which assume a reward

function) to the offline IL setting does not work. In practice, expert demonstrations are scarce due to the high cost of obtaining them. Thus, they typically cover only a small fraction of the state and action spaces, which in turn makes the distribution drift issue even more stand out compared with the standard offline RL setting with a reward function. We mitigate this issue by assuming a large number of supplementary imperfect demonstrations, without requiring any level of optimality for these imperfect demonstrations; they may contain expert or near-expert trajectories (Wu et al., 2019) as well as random ones all together. This generality covers the situations from real-world applications, but at the same time, it poses a significant challenge for the design of a successful offline IL algorithm.

In this paper, we propose DemoDICE, a novel model-free algorithm for offline IL from expert and imperfect demonstrations. We formulate an offline IL objective which not only mitigates distribution shift from the demonstration-data distribution but also naturally utilizes imperfect demonstrations. Our new formulation allows us to leverage OptiDICE (Lee et al., 2021), which learns a policy in the space of stationary distributions but suffers from the instability issue in practice. We tackle the issue by proposing an alternative objective, which leads to a stable algorithm in practice while keeping the optimal stationary distribution of OptiDICE. Finally, we introduce a method to extract the optimal policy from a learned stationary distribution in a simple yet effective way. Our extensive evaluations show that DemoDICE achieves performance competitive to or better than a state-of-the-art off-policy IL algorithm in the offline-IL tasks with expert and imperfect demonstrations.

2 PRELIMINARIES

2.1 MARKOV DECISION PROCESS (MDP)

We assume an environment modeled as a Markov Decision Process (MDP), defined by tuple $M = \langle S, A, T, R, p_0, \gamma \rangle$, where S is the set of states, A is the set of actions, $T : S \times A \to \Delta(S)$ is the probability $p(s_{t+1}|s_t, a_t)$ of making transition from state s_t to state s_{t+1} by executing action a_t at timestep t, $R : S \times A \to \mathbb{R}$ is the reward function, $p_0 \in \Delta(S)$ is the distribution of the initial state s_0, and $\gamma \in [0, 1]$ is the discount factor. A policy $\pi : S \to \Delta(A)$ of MDP M is a mapping from states of M to distributions over actions. For simplicity, we use $T(s'|s, a)$ and $\pi(a|s)$ to indicate their evaluations. For the given policy π, the stationary distribution d^π is defined as follows:

$$d^\pi(s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t p\big(s_t = s, a_t = a \big| s_0 \sim p_0(\cdot), s_t \sim T(\cdot|s_{t-1}, a_{t-1}), a_t \sim \pi(\cdot|s_t)\big)$$

We assume a precollected dataset D^E of (s, a, s') tuples generated by the expert, and a precollected imperfect dataset D^I (generated by unknown degrees of optimality). We define $D^U = D^E \cup D^I$, the dataset of all demonstrations, and denote the distributions on state-action pairs or state-action-state tuples of the datasets D^E and D^U as d^E and d^U, respectively. Thus, for instance, we write $(s, a) \sim d^U$ to mean the uniform distribution over all state-action pairs (with multiplicities) appearing in some trace in D^U, and $(s, a, s') \sim d^U$ to denote the uniform distribution over all state-action-state tuples obtained similarly from D^U.

2.2 IMITATION LEARNING

Behavior cloning (BC) is a classical IL approach, which attempts to find a function that maps s to a via supervised learning. The standard BC finds a policy π by minimizing the cross-entropy loss:

$$\min_\pi J_{\text{BC}}(\pi) := \min_\pi -\frac{1}{|D|} \sum_{(s,a)\in D} \log \pi(a|s). \tag{1}$$

However, it is known to be brittle (Ross et al., 2011) when the interaction with the environment deviates from the scarce trajectories in D^E. In such cases, BC fails to recover optimal policies.

One of the notable approaches for IL is to formulate the problem as distribution matching (Ho & Ermon, 2016; Ke et al., 2019; Kostrikov et al., 2020). When instantiated with the KL divergence widely used in previous IL works (Ke et al., 2019; Kostrikov et al., 2020), the approach amounts to

finding a policy π by optimizing the following objective:

$$\max_{\pi} -D_{\mathrm{KL}}(d^{\pi}\|d^{E}) = \mathbb{E}_{(s,a)\sim d^{\pi}}\left[\log \frac{d^{E}(s,a)}{d^{\pi}(s,a)}\right]. \tag{2}$$

Since we cannot directly access the exact value of $d^E(s,a)$ and $d^\pi(s,a)$, we estimate their ratio by optimizing the objective of GAN (Goodfellow et al., 2014), given as follows:

$$\max_{c:S\times A\to(0,1)} J_{\mathrm{GAIL}}(c) := \mathbb{E}_{(s,a)\sim d^{E}}[\log c(s,a)] + \mathbb{E}_{(s,a)\sim d^{\pi}}[\log(1-c(s,a))]. \tag{3}$$

Based on this connection between generative adversarial networks (GANs) and IL, AIL algorithms focus on recovering the expert policy (Ho & Ermon, 2016; Kostrikov et al., 2019).

3 DEMODICE

In this section, we present a novel model-free offline IL algorithm named *offline imitation learning using additional imperfect Demonstrations via stationary DIstribution Correction Estimation* (DemoDICE). Starting from a regularized offline IL objective which accords with offline RL algorithms, we present a formulation that does not require on-policy samples. Such formulation allows us to use offline policy optimization for offline IL from expert and imperfect demonstrations (Section 3.1). Then we apply OptiDICE (Lee et al., 2021) and obtain a simple convex optimization objective. Since the objective is unstable in practice, we transform the objective to an alternative yet still convex objective (Section 3.2). Finally, we show how to extract the policy from the learned correction term (Section 3.3).

3.1 TRANSFORM CONSTRAINED OPTIMIZATION INTO NESTED OPTIMIZATION

In the context of offline RL, most works use some regularization to overcome the extrapolation error in offline settings (Fujimoto et al., 2019; Kumar et al., 2019). Especially, inspired by the following KL-divergence regularized policy optimization framework (Nachum et al., 2019b; Lee et al., 2021)

$$\pi^* := \arg\max_{\pi} -\mathbb{E}_{(s,a)\sim d^{\pi}}[R(s,a)] - \alpha D_{\mathrm{KL}}(d^{\pi}\|d^{U}), \tag{4}$$

we consider the following KL-divergence regularized distribution matching objective:

$$\pi^* := \arg\max_{\pi} -D_{\mathrm{KL}}(d^{\pi}\|d^{E}) - \alpha D_{\mathrm{KL}}(d^{\pi}\|d^{U}), \tag{5}$$

where $\alpha \geq 0$ is a hyperparameter that controls the balance between minimizing KL divergence with d^E and preventing deviation of d^π from d^U.

Many online AIL algorithms estimate divergence between expert and current policy using on-policy samples, which is not available in offline scenario. In contrast, to construct a tractable optimization problem in the offline setting, we consider a problem equivalent to Equation 5 in terms of stationary distribution d similar to OptiDICE (Lee et al., 2021):

$$\max_{d} \; -D_{\mathrm{KL}}(d\|d^{E}) - \alpha D_{\mathrm{KL}}(d\|d^{U}) \tag{6}$$

$$\text{s.t} \sum_{a} d(s,a) = (1-\gamma)p_0(s) + \gamma \sum_{\bar{s},\bar{a}} T(s|\bar{s},\bar{a})d(\bar{s},\bar{a}) \;\; \forall s, \tag{7}$$

$$d(s,a) \geq 0 \;\; \forall s,a. \tag{8}$$

The constraints (7-8) are called the Bellman flow constraints. The dual problem for the above constrained optimization problem is

$$\max_{d\geq 0}\min_{\nu} -D_{\mathrm{KL}}(d\|d^{E}) - \alpha D_{\mathrm{KL}}(d\|d^{U}) + \sum_{s}\nu(s)((1-\gamma)p_0(s) + \gamma(\mathcal{T}_* d)(s) - (\mathcal{B}_* d)(s)), \tag{9}$$

where $\nu(s)$ are the Lagrange multipliers, $(\mathcal{B}_* d)(s) := \sum_a d(s,a)$ is the marginalization operator, and $(\mathcal{T}_* d)(s) := \sum_{\bar{s},\bar{a}} T(s|\bar{s},\bar{a})d(\bar{s},\bar{a})$ is the transposed Bellman operator. We introduce following

derivations for the optimization problem (9) to use the theoretical analysis in OptiDICE (Lee et al., 2021):

$$
\begin{aligned}
&- D_{\mathrm{KL}}(d\|d^E) - \alpha D_{\mathrm{KL}}(d\|d^U) + \sum_s \nu(s)((1-\gamma)p_0(s) + \gamma(\mathcal{T}_* d)(s) - (\mathcal{B}_* d)(s)) \\
&= (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + \mathbb{E}_{(s,a)\sim d}\left[\gamma(\mathcal{T}\nu)(s,a) - \nu(s) - \log\frac{d(s,a)}{d^E(s,a)} - \alpha\log\frac{d(s,a)}{d^U(s,a)}\right] \quad (10) \\
&= (1-\gamma)\mathbb{E}_{p_0}[\nu(s)] + \mathbb{E}_d\left[\gamma(\mathcal{T}\nu)(s,a) - \nu(s) \underbrace{+\log\frac{d^E(s,a)}{d^U(s,a)}}_{:=r(s,a)} - (1+\alpha)\log\underbrace{\frac{d(s,a)}{d^U(s,a)}}_{:=w(s,a)}\right], \quad (11)
\end{aligned}
$$

the equality in Equation 10 holds from the following properties of transpose operators:

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

where $(\mathcal{B}\nu)(s,a) = \nu(s)$, $(\mathcal{T}\nu)(s,a) = \sum_{s'} T(s'|s,a)\nu(s')$, with assumption $d^E(s,a) > 0$ when $d(s,a) > 0$ (Nachum et al., 2019a). We introduce another log ratio, denoted by $r(s,a)$ in Equation 11 to avoid using $\log\frac{d^E(s,a)}{d(s,a)}$, which requires on-policy samples to estimate. Unlike $\log\frac{d^E(s,a)}{d(s,a)}$, we can estimate $r(s,a)$ in the offline setting using d^E and d^U, as we will discuss in the next section in detail.

We change the distribution used in the expectation of Equation 11 from d to d^U by following the standard trick of importance sampling as follows:

$$
\begin{aligned}
&(1-\gamma)\mathbb{E}_{s_0\sim p_0} + \mathbb{E}_{(s,a)\sim d}[\underbrace{r(s,a) + \gamma(\mathcal{T}\nu)(s,a) - (\mathcal{B}\nu)(s,a)}_{:=A_\nu(s,a)\text{(advantage using }\nu)} - (1+\alpha)\log w(s,a)] \\
&= (1-\gamma)\mathbb{E}_{s_0\sim p_0} + \mathbb{E}_{(s,a)\sim d^U}\left[w(s,a)\big(A_\nu(s,a) - (1+\alpha)\log w(s,a)\big)\right] \\
&=: L(w,\nu;r). \quad (12)
\end{aligned}
$$

As an alternative, one can convert expectation of d to d^E instead of d^U in Equation 11 by the similar application of the trick of importance sampling.

$$
(1-\gamma)\mathbb{E}_{s_0\sim p_0} + \mathbb{E}_{(s,a)\sim d^E}\left[\exp(-r(s,a))w(s,a)\big(A_\nu(s,a) - (1+\alpha)\log w(s,a)\big)\right].
$$

This alternative is inferior to the original in Equation 12, since the latter uses the distribution d^U and lets the training algorithm use the whole dataset D^U but the former is based on d^E and the dataset D^E of expert demonstrations only.

In summary, DemoDICE solves the following maximin optimization:

$$
\max_{w\geq 0}\min_{\nu} L(w,\nu;r), \quad (13)
$$

where r is trained by using precollected datasets. Note that the solution w^* of Equation 13 is the ratio of two distributions, the stationary distribution d^{π^*} of the optimal policy π^* and the empirical distribution d^U from the entire dataset D^U of expert and imperfect demonstrations.

3.2 Pretrained stationary distribution ratio and a closed-form solution

To solve the problem (13), we can pretrain r, the negative log-ratio of $d^U(s,a)$ and $d^E(s,a)$. To this end, we train a discriminator $c : S\times A \to [0,1]$ using the following minimization objective (Goodfellow et al., 2014):

$$
\min_{c:S\times A\to[0,1]} J_c(d^E, d^U) := \mathbb{E}_{d^E}[\log c(s,a)] + \mathbb{E}_{d^U}[\log(1-c(s,a))], \quad (14)
$$

whose minimizer is $c^*(s,a) = \frac{d^E(s,a)}{d^U(s,a)+d^E(s,a)}$. By using c^*, r can be also obtained as

$$
r(s,a) = -\log\left(\frac{1}{c^*(s,a)} - 1\right). \quad (15)
$$

Since the optimization (9) is a convex optimization and the strong duality holds, the optimization (9) is equal to

$$\min_{\nu} \max_{w \geq 0} L(w, \nu; r). \tag{16}$$

In Lee et al. (2021), the solution of inner max optimization of (16) turns out to be $w^*_\nu(s,a) = \exp\left(\frac{A_\nu(s,a)}{1+\alpha} - 1\right)$ for all (s,a), where $A_\nu(s,a) := r(s,a) + \gamma(\mathcal{T}\nu)(s,a) - \nu(s)$. By using $w^*_\nu(s,a)$, optimization (16) can be reduced to a simple minimization problem as follows:

$$\min_{\nu} L(w^*_\nu, \nu; r) = (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + (1+\alpha)\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha} - 1\right)\right] \tag{17}$$

Although $L(w^*_\nu, \nu; r)$ in Equation 17 is convex on ν, we observe that the $L(w^*_\nu, \nu; r)$ in Equation 17 diverges in practice because its exponential term is prone to explosion. In order to derive a numerically-stable alternative objective, we describe two theoretical results:

Proposition 1. *Define the objective $\widetilde{L}(\nu; r)$ as*

$$\widetilde{L}(\nu; r) := (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + (1+\alpha)\log \mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha}\right)\right]. \tag{18}$$

Then, for sufficiently large families of ν, the following equality holds:

$$\min_{\nu} L(w^*_\nu, \nu; r) = \min_{\tilde{\nu}} \widetilde{L}(\tilde{\nu}; r).$$

In particular, there is a constant C such that the following equation holds for two optimal functions $\nu^* := \arg\min_\nu L(w^*_\nu, \nu; r)$ and $\tilde{\nu}^* := \arg\min_\nu \widetilde{L}(\nu; r)$: (The proof can be found in Appendix A.)

$$\nu^*(s,a) = \tilde{\nu}^*(s,a) + C \quad \forall s, a. \tag{19}$$

Furthermore, we observe following:

Proposition 2. *The objective $\widetilde{L}(\nu; r)$ is convex with respect to ν. (The proof is in Appendix B.)*

As observed by Lee et al. (2021), the objective $L(w^*_\nu, \nu; r)$ in Equation 17 has the instability issue since the gradient with respect to ν involves an unbounded easily-exploding function $\exp(\cdot)$. In contrast, the alternative objective $\widetilde{L}(\nu; r)$ in Equation 18 does not suffer from the same stability issue because in this case, the gradient forms a soft-max and is bounded by 1.

To summarize, we have presented a training objective $\widetilde{L}(\nu; r)$ which significantly stabilizes the objective $L(w^*_\nu, \nu; r)$ while having the same optimal stationary distribution. We converted the minimax optimization (16) to a single minimization, which is much more stable. Proposition 1 is, however, limited in that it leaves the optimal stationary distribution unnormalized. Thus, we need a method to extract an optimal policy from an unnormalized optimal stationary distribution. The next section presents such a method.

3.3 Policy extraction

Finally, we present a method for extracting a policy from the optimal $\tilde{\nu}^* = \arg\min_\nu \widetilde{L}(\nu; r)$. Our method is based on the following optimization, which amounts to the variant of weighted BC:

$$\min_{\pi} -\mathbb{E}_{(s,a)\sim d^{\pi^*}}[\log \pi(a|s)] = -\mathbb{E}_{(s,a)\sim d^U}[w^*(s,a)\log \pi(a|s)]. \tag{20}$$

Note that we cannot directly optimize the above objective because we cannot compute $w^*(s,a)$ using $\tilde{\nu}^*$. To overcome this challenge and derive our method, we observe that Proposition 1 implies the following relationship between $\tilde{\nu}^*$ and the exact distribution correction: (see Appendix C for more detailed derivation)

$$\widetilde{w}_{\tilde{\nu}^*}(s,a) := \exp\left(\frac{A_{\tilde{\nu}^*}(s,a)}{1+\alpha}\right) \propto \frac{d^*(s,a)}{d^U(s,a)}.$$

Using this observation, we estimate Equation 20 using self-normalized importance sampling (Owen, 2013) as shown below:

$$\min_{\pi} J_{\pi}(\tilde{\nu}^*) = -\frac{\mathbb{E}_{(s,a)\sim d^U}\left[\widetilde{w}_{\tilde{\nu}^*}(s,a)\log\pi(a|s)\right]}{\mathbb{E}_{(s,a)\sim d^U}\left[\widetilde{w}_{\tilde{\nu}^*}(s,a)\right]}. \tag{21}$$

Based on the weighted BC, we simply extract the policy without training any additional network. Our policy extraction is highly related to variants of weighted BC (Wang et al., 2018; 2020; Siegel et al., 2020) in offline RL. The common idea of these methods is (1) training an action-value function $Q(s,a)$ and (2) estimating the corresponding advantage $A(s,a)$. Using an increasing, non-negative function $f(\cdot)$ (*e.g.,* $\exp(\cdot)$), they perform weighted BC on precollected dataset with defining weight $f(A(s,a),s,a)$ as:

$$\arg\min_{\pi} \mathbb{E}_{(s,a)\sim d^U}\left[-f(A(s,a),s,a)\log\pi(a|s)\right].$$

Finally, if we regard the advantage of optimal ν^* in Equation 17 as an advantage function, our policy extraction matches the offline RL based on weighted BC.

4 Related Works

Learning from imperfect demonstrations Among several recent works (Wu et al., 2019; Brown et al., 2019; 2020; Tangkaratt et al., 2020) that attempted to leverage imperfect demonstrations in IL, IC-GAIL (Wu et al., 2019) and Trajectory-ranked reward extrapolation (T-REX) (Brown et al., 2019) are closely related to our work. Assuming that some of trajectories in the imperfect demonstrations are provided with labels of ground-truth binary optimalities, IC-GAIL uses expert and imperfect demonstrations to address on-policy IL with the labels. T-REX assumes that all the suboptimal demonstrations are ranked according to their true accumulative rewards. Thanks to those ranked demonstrations, T-REX trains a reward function and exploits it to learn an optimal policy. However, these algorithms are not readily applicable to offline IL from expert and imperfect demonstrations since they require (1) interaction with environment and (2) extra information about imperfect demonstrations.

Stationary distribution corrections In RL and IL, some prior works have used distribution corrections for off-policy learning. In AlgaeDICE (Nachum et al., 2019b), regularization in the space of stationary distribution is augmented to a policy optimization objective to solve off-policy RL problems. In addition, by using dual formulation of f-divergence and change of variables (Nachum et al., 2019a), an off-policy learning objective is derived in AlgaeDICE. ValueDICE (Kostrikov et al., 2020) minimizes the KL divergence between the agent and the expert stationary distributions to solve IL problems. Similar to the AlgaeDICE, ValueDICE obtains an off-policy learning objective for distribution matching by using the dual formulation of KL divergence and change of variables. Both AlgaeDICE and ValueDICE objectives are optimized by nested optimization, which may suffer from numerical instability. In contrast, OptiDICE (Lee et al., 2021) reduces the same optimization used in AlgaeDICE to unconstrained convex optimization. Although it shows promising performance in offline RL tasks, directly applying it to offline IL from expert and imperfect demonstrations is not trivial as we discussed in Section 3.1 and Section 3.2.

5 Experiments

In this section, we present the empirical performance of DemoDICE and baseline methods on MuJoCo continuous control environments (Todorov et al., 2012) using the OpenAI Gym (Brockman et al., 2016) framework. We provide experimental results for 4 MuJoCo environments: Hopper, Walker2d, HalfCheetah, and Ant. We utilize D4RL datasets (Fu et al., 2020) to construct expert and imperfect demonstrations for our experiments. We construct datasets, baselines, and evaluation protocol according to the following procedures:

Datasets For each of MuJoCo environments, we utilize three types of D4RL datasets (Fu et al., 2020), whose name end with "-expert-v2", "-full_replay-v2", or "-random-v2". In the remainder of this section, we refer them by using corresponding suffixes. The set of expert demonstrations D^E consists of the first trajectory in `expert-v2`.

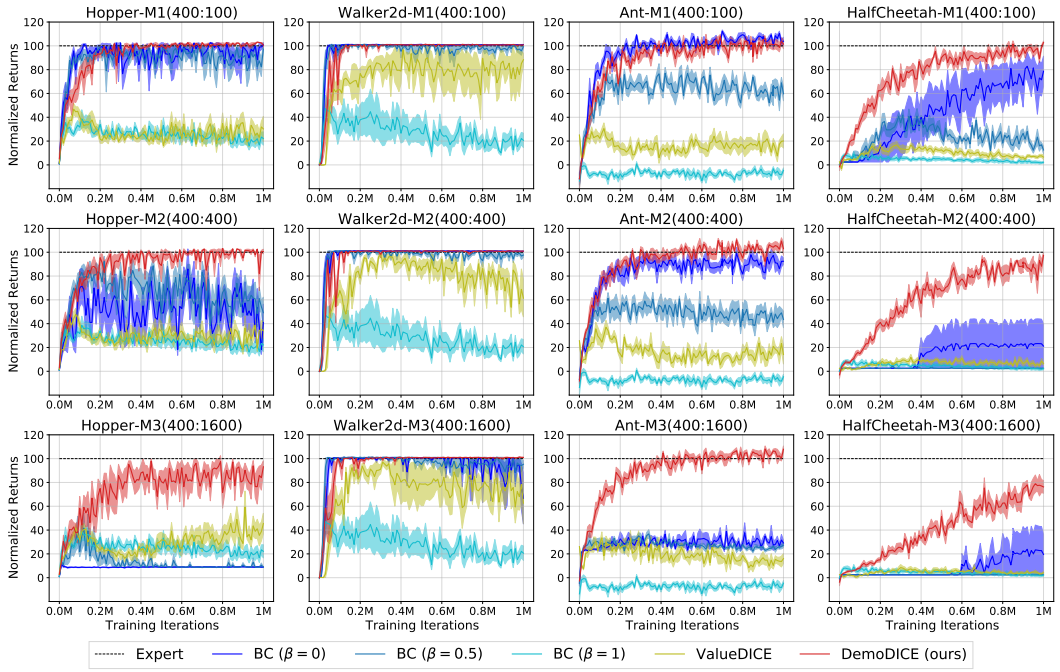

Figure 1: Performance of DemoDICE and baseline algorithms for mixed dataset tasks M1, M2, and M3. Especially, in M3 tasks, which contains lots of bad trajectories, DemoDICE maintains performance while ValueDICE and BC fail to achieve competitive performance. We plot the mean and the standard errors (shaded area) of the normalized scores over 5 random seeds.

Baselines We compare our method with two strong baseline methods, BC and ValueDICE. To consider the potential benefit of utilizing D^I, we carefully tuned BC with 5 different values of $\beta \in \{1, 0.75, 0.5, 0.25, 0\}$, which controls the balance between minimizing negative log-likelihood of D^E and minimizing that of $D^U = D^E \cup D^I$ as follows:

$$\min_{\pi} J_{BC(\beta)}(\pi) := -\beta \cdot \frac{1}{|D^E|} \sum_{(s,a)\in D^E} \log \pi(a|s) - (1-\beta) \cdot \frac{1}{|D^U|} \sum_{(s,a)\in D^U} \log \pi(a|s).$$

We denote those 5 different settings by BC(β=1), BC(β=0.75), BC(β=0.5), BC(β=0.25), BC(β=0). Lastly, we made ValueDICE to utilize imperfect datasets by plugging in all the demonstrations to replay buffer.

Evaluation metric The normalized scores for each environment are measured by $\texttt{normalized score} = 100 \times \frac{\texttt{score}-\texttt{random score}}{\texttt{expert score}-\texttt{random score}}$, where expert and random scores are average returns of trajectories in `expert-v2` and `random-v2`, respectively. We compute the average normalized score and the standard error over five random seeds. In the following subsections, we provide experimental results on two types of imperfect datasets.

5.1 Mixed Dataset

DemoDICE mainly aims to overcome distributional drift caused by the lack of expert demonstrations. When DemoDICE successfully learns the optimal stationary distribution correction, we can expect that DemoDICE distinguishes expert trajectories from non-expert ones to update its own policy. Based on this intuition, we hypothesize that if the same sufficiently-many expert trajectories are included in imperfect demonstrations, DemoDICE achieves the optimal performance invariant to the number of non-expert trajectories in imperfect demonstrations. To see if it is the case, we use mixed datasets which have the same expert trajectories but have different numbers of non-expert trajectories in imperfect demonstrations.

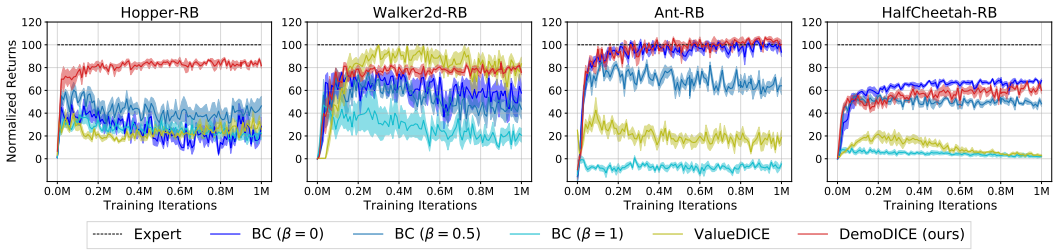

Figure 2: Performance of DemoDICE and baseline algorithms for replay buffer task RB. In RB tasks, DemoDICE achieves better or similar performance than ValueDICE. We plot the mean and the standard error (shaded area) of the normalized scores over 5 random seeds.

Experimental setup Across all environments, we consider 3 tasks, each of which is called one of M1, M2, or M3 and is provided with expert and imperfect demonstrations. While sharing the expert demonstrations composed of the same single trajectory, imperfect demonstrations in each task are composed of expert and random demonstrations with different ratios such as 1:0.25 (M1), 1:1 (M2), and 1:4 (M3). Specifically, imperfect demonstrations in M1, M2, and M3 are composed of 400 expert trajectories sampled from expert-v2 and 100, 400, and 1600 random trajectories sampled from random-v2, respectively.

Result We report the result of DemoDICE and comparing methods in Figure 1. For simplicity, BC($\beta = 0.25$) and BC($\beta = 0.75$) are in Appendix E. Note that imperfect demonstrations in the tasks M1, M2, and M3 share the same expert trajectories, while having the different number of random trajectories. Ideally, if an algorithm uses only expert trajectories from imperfect demonstrations, the algorithm should have the same performance in the tasks M1, M2, and M3, as discussed at the beginning of this subsection, except convergence speed. While ValueDICE and all the variants of BC catastrophically fail to learn expert policies in task M3 on all the environments except Walker2d, DemoDICE reaches the expert performance regardless of environments and tasks.

The result strongly implies that DemoDICE succeeds to learn appropriate stationary distribution correction, which is possible only when the algorithm effectively leverages expert trajectories in imperfect demonstrations ignoring the rest. Furthermore, it is remarkable that DemoDICE reaches the expert performance in HalfCheetah in the end, which can be found in Appendix E.1. To summarize, we have empirically shown that DemoDICE is robust to any choices of the environments and tasks and significantly outperforms existing methods on offline IL.

5.2 Replay Buffer Dataset

When we use mixed datasets as imperfect demonstration sets, ValueDICE poorly performs in all environments except Walker2d as we observed in Section 5.1. Since `full_replay-v2` consists of the replay buffer's data during off-policy training, we believe it is one of the most practically-relevant datasets to fill ValueDICE's replay buffer. Therefore, we conduct additional experiments using this dataset as the imperfect demonstrations.

Experimental setup Applying the same expert demonstrations used in Section 5.1, we employ another task named RB, which uses `full_replay-v2` directly as its imperfect dataset.

Result Figure 2 summarizes the result of DemoDICE and baseline methods. We observe that DemoDICE performs on par with ValueDICE in Walker2d. However, on the other environments, DemoDICE strictly outperforms ValueDICE. While showing the performance competitive to the best performing baseline methods in Walker2d, Ant, and HalfCheetah, DemoDICE outperforms all the methods in Hopper by a significant margin.

Since the replay buffer dataset is collected by the policy which evolves (in policy training phase), it can be regarded as a set of imperfect demonstration from a wide range of sources. We emphasize that DemoDICE performs well not only when bad trajectories make up majority of the given imperfect demonstrations but also when they are generated from multiple sources.

6 CONCLUSION

We have presented DemoDICE, an algorithm for offline IL from expert and imperfect demonstrations that achieves state-of-the-art performance on various offline IL tasks. We first introduced a regularized offline IL objective and reformulated the objective so as to make it natural to apply OptiDICE (Lee et al., 2021). We then tackled the instability coming from the naive application of OptiDICE by the alternative objective which yields not only the same optimal stationary distribution but also a stable convex optimization. Furthermore, we presented the method to extract an optimal policy even without introducing any additional neural network. Lastly, our extensive empirical evaluations showed that DemoDICE achieves remarkable performance close to the expert by exploiting imperfect demonstrations effectively.

REFERENCES

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning (ICML)*, pp. 783–792. PMLR, 2019.

Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning (CoRL)*, pp. 330–359. PMLR, 2020.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning, 2020.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning (ICML)*, pp. 2052–2062. PMLR, 2019.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. *arXiv preprint arXiv:1704.00028*, 2017.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Liyiming Ke, Matt Barnes, Wen Sun, Gilwoo Lee, Sanjiban Choudhury, and Siddhartha Srinivasa. Imitation learning as f-divergence minimization. *arXiv preprint arXiv:1905.12888*, 2019.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. In *International Conference on Learning Representations (ICLR)*, 2020.

Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with Fisher divergence critic regularization. In *International Conference on Machine Learning*, pp. 5774–5783. PMLR, 2021.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy Q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Jongmin Lee, Wonseok Jeon, Byung-Jun Lee, Joelle Pineau, and Kee-Eung Kim. OptiDICE: Offline policy optimization via stationary distribution correction estimation. In *International Conference on Machine Learning (ICML)*, 2021.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. DualDICE: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019a.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. AlgaeDICE: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019b.

Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2000.

Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Stéphane Ross, Geoffrey Gordon, and J Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 627–635, 2011.

Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2020.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Voot Tangkaratt, Bo Han, Mohammad Emtiyaz Khan, and Masashi Sugiyama. Variational imitation learning with diverse-quality demonstrations. In *International Conference on Machine Learning (ICML)*, pp. 9407–9417. PMLR, 2020.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Qing Wang, Jiechao Xiong, Lei Han, Peng Sun, Han Liu, and Tong Zhang. Exponentially weighted imitation learning for batched historical data. In *NeurIPS*, pp. 6291–6300, 2018.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning (ICML)*, pp. 6818–6827, 2019.

A PROOF OF PROPOSITION 1

We remind the objectives in Equation 17 and Equation 19:

$$L(w_\nu^*, \nu; r) := (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + (1+\alpha)\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha} - 1\right)\right],$$
$$\widetilde{L}(\nu; r) := (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + (1+\alpha)\log\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha}\right)\right],$$

where

$$A_\nu(s,a) := r(s,a) + \gamma(\mathcal{T}\nu)(s,a) - \nu(s),$$
$$w_\nu^*(s,a) := \exp\left(\frac{A_\nu(s,a)}{1+\alpha} - 1\right).$$

Note that for any constant C, the advantage for $\nu + C$ is:

$$\begin{aligned}
A_{(\nu+C)}(s,a) &= r(s,a) + \gamma(\mathcal{T}(\nu+C))(s,a) - (\nu(s)+C) \\
&= r(s,a) + \gamma\big((\mathcal{T}\nu)(s,a) + C\big) - (\nu(s)+C) \\
&= r(s,a) + \gamma(\mathcal{T}\nu)(s,a) - \nu(s) - (1-\gamma)C \\
&= A_\nu(s,a) - (1-\gamma)C
\end{aligned}$$

Lemma 1. *For arbitrary function ν and constant C, the equality*

$$\widetilde{L}(\nu; r) = \widetilde{L}(\nu + C; r)$$

holds.

Proof. From the definition of $\widetilde{L}(\nu; r)$, we derive the following equation:

$$\begin{aligned}
&\widetilde{L}(\nu + C; r) \\
&= (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s) + C] + (1+\alpha)\log\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_{(\nu+C)}(s,a)}{1+\alpha}\right)\right] \\
&= (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s) + C] + (1+\alpha)\log\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a) - (1-\gamma)C}{1+\alpha}\right)\right] \\
&= (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s) + C] + (1+\alpha)\log\left[\exp\left(-\frac{(1-\gamma)C}{1+\alpha}\right)\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha}\right)\right]\right] \\
&= (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s) + C] + (1+\alpha)\left[\log\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha}\right)\right] - \frac{(1-\gamma)C}{1+\alpha}\right] \\
&= (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + (1+\alpha)\log\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha}\right)\right] \\
&= \widetilde{L}(\nu; r)
\end{aligned}$$

□

Lemma 2. *For any function ν, the following inequality always holds:*

$$L(w_\nu^*, \nu; r) \geq \widetilde{L}(\nu; r).$$

Equality holds if and only if

$$\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha} - 1\right)\right] = 1.$$

Proof. For any $y \geq 0$,

$$y - 1 \geq \log y,$$

and equality holds if and only if $y = 1$. From this inequality, we can derive the inequality directly:

$$\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha}-1\right)\right]-1 \geq \log \mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha}-1\right)\right],$$

and equality holds if and only if

$$\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha}-1\right)\right]=1.$$

Finally, we obtain the following results:

$$\begin{aligned}
L(w_\nu^*,\nu;r) &= (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + (1+\alpha)\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha}-1\right)\right] \\
&\geq (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + (1+\alpha)\left\{\log \mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha}-1\right)\right]+1\right\} \\
&\geq (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + (1+\alpha)\log\left\{\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha}-1\right)\right]\exp(1)\right\} \\
&= (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + (1+\alpha)\log \mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha}\right)\right] \\
&= \widetilde{L}(\nu;r).
\end{aligned}$$

The equality holds if and only if

$$\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha}-1\right)\right]=1$$

□

Lemma 3. *For any optimal solution $\tilde{\nu}^* = \arg\min_\nu \widetilde{L}(\nu;r)$, there is a constant C such that $\tilde{\nu}^* + C$ is an optimal solution of $\min_\nu L(w_\nu^*,\nu;r)$.*

Proof. Let $\tilde{\nu}^*$ be an optimal solution of $\arg\min_\nu \widetilde{L}(\nu;r)$ and

$$C^* := \frac{1+\alpha}{1-\gamma}\log \mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_{\tilde{\nu}^*}(s,a)}{1+\alpha}-1\right)\right]$$

Then, $\hat{\nu} := \tilde{\nu}^* + C^*$ satisfies

$$\begin{aligned}
&\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_{\tilde{\nu}}(s,a)}{1+\alpha}-1\right)\right] \\
&= \mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_{(\tilde{\nu}^*+C^*)}(s,a)}{1+\alpha}-1\right)\right] \\
&= \mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_{\tilde{\nu}^*}(s,a)-(1-\gamma)C^*}{1+\alpha}-1\right)\right] \\
&= \mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_{\tilde{\nu}^*}(s,a)}{1+\alpha}-\frac{1-\gamma}{1+\alpha}C^*-1\right)\right] \\
&= \mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_{\tilde{\nu}^*}(s,a)}{1+\alpha}-1\right)\right]\exp\left(-\frac{1-\gamma}{1+\alpha}C^*\right) \\
&= \mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_{\tilde{\nu}^*}(s,a)}{1+\alpha}-1\right)\right]\left\{\mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_{\tilde{\nu}^*}(s,a)}{1+\alpha}-1\right)\right]\right\}^{-1} \\
&= 1.
\end{aligned}$$

Furthermore, $\hat{\nu}$ is also an optimal solution of $\min_\nu \widetilde{L}(\nu;r)$ by Lemma 1. Then, by equality condition in Lemma 2,

$$L(w_{\hat{\nu}}^*,\hat{\nu};r) = \widetilde{L}(\hat{\nu};r) = \min_\nu \widetilde{L}(\nu;r) \leq \min_\nu L(w_\nu^*,\nu;r).$$

It means, $\hat{\nu}$ is an optimal solution of $\min_\nu L(w_\nu^*,\nu;r)$ □

Lemma 4. *An optimal solution* $\nu^* = \arg\min_\nu L(w^*_\nu, \nu; r)$ *is also an optimal solution of* $\min_\nu \widetilde{L}(\nu; r)$

Proof. From the Lemma 2,

$$\min_\nu L(w^*_\nu, \nu; r) = L(w^*_{\nu^*}, \nu^*; r) \geq \widetilde{L}(\nu^*; r).$$

Lemma 3 show that $\min_\nu L(w^*_\nu, \nu; r)$ and $\min_\nu \widetilde{L}(\nu; r)$ have the same minimum value,

$$\widetilde{L}(\nu^*; r) \leq L(w^*_{\nu^*}, \nu^*; r) = \min_\nu L(w^*_\nu, \nu; r) = \min_\nu \widetilde{L}(\nu; r)$$

holds. It means that $\widetilde{L}(\nu^*; r)$ is the minimum value of $\widetilde{L}$ and therefore, ν^* is an optimal solution of $\min_\nu \widetilde{L}(\nu; r)$. □

Finally, we ready to prove the proposition:

Proposition 1. *Define the objective* $\widetilde{L}(\nu; r)$ *as*

$$\widetilde{L}(\nu; r) := (1-\gamma)\mathbb{E}_{s\sim p_0}[\nu(s)] + (1+\alpha)\log \mathbb{E}_{(s,a)\sim d^U}\left[\exp\left(\frac{A_\nu(s,a)}{1+\alpha}\right)\right]. \tag{18}$$

Then, for sufficiently large families of ν*, the following equality holds:*

$$\min_\nu L(w^*_\nu, \nu; r) = \min_{\tilde{\nu}} \widetilde{L}(\tilde{\nu}; r).$$

In particular, there is a constant C *such that the following equation holds for two optimal functions* $\nu^* := \arg\min_\nu L(w^*_\nu, \nu; r)$ *and* $\tilde{\nu}^* := \arg\min_\nu \widetilde{L}(\nu; r)$*: (The proof can be found in Appendix A.)*

$$\nu^*(s,a) = \tilde{\nu}^*(s,a) + C \quad \forall s, a. \tag{19}$$

Proof. Lemma 3 shows that

$$\min_\nu L(w^*_\nu, \nu; r) = L(w^*_{\nu^*}, \nu^*; r) \geq \widetilde{L}(\nu^*; r).$$

Furthermore, Lemma 3 and Lemma 4 shows that the last equation holds.

□

B Proof of Proposition 2

Proposition 2. *The objective* $\widetilde{L}(\nu; r)$ *is convex with respect to* ν*. (The proof is in Appendix B.)*

Proof. For any functions ν, ν' and constant $\lambda \in [0,1]$, we can derive the following equality:

$$\begin{aligned}
&A_{(\lambda\nu+(1-\lambda)\nu')}(s,a) \\
&= r(s,a) + \gamma(\mathcal{T}(\lambda\nu + (1-\lambda)\nu'))(s,a) - (\lambda\nu(s) + (1-\lambda)\nu'(s)) \\
&= r(s,a) + \gamma\lambda(\mathcal{T}\nu)(s,a) + \gamma(1-\lambda)(\mathcal{T}\nu')(s,a) - (\lambda\nu(s) + (1-\lambda)\nu'(s)) \\
&= \lambda(r(s,a) + \gamma(\mathcal{T}\nu)(s,a) - \lambda\nu(s)) + (1-\lambda)(r(s,a) + \gamma(\mathcal{T}\nu')(s,a) - \lambda\nu'(s)) \\
&= \lambda A_\nu(s,a) + (1-\lambda)A_{\nu'}(s,a).
\end{aligned}$$

It means, A_ν is the linear function with respect to ν. Furthermore, using the convexity of log-sum-exp,

$$\begin{aligned}
&\log \mathbb{E}_{(s,a)\sim d^U}\left[\exp(A_{(\lambda\nu+(1-\lambda)\nu')}(s,a))\right] \\
&= \log \mathbb{E}_{(s,a)\sim d^U}\left[\exp(\lambda A_\nu(s,a) + (1-\lambda)A_{\nu'}(s,a))\right] \\
&\leq \lambda \log \mathbb{E}_{(s,a)\sim d^U}\left[\exp(A_\nu(s,a))\right] + (1-\lambda)\log \mathbb{E}_{(s,a)\sim d^U}\left[A_{\nu'}(s,a))\right]
\end{aligned}$$

Therefore, $\widetilde{L}(\nu; r)$ is convex function with respect to ν. □

C POLICY EXTRACTION

By the definition of A_ν,

$$\begin{aligned}
&\exp\left(\frac{A_{(\tilde{\nu}^*+C)}(s,a)}{1+\alpha} - 1\right) \\
&= \exp\left(\frac{A_{\tilde{\nu}^*}(s,a) - (1-\gamma)C}{1+\alpha} - 1\right) \\
&= \exp\left(\frac{A_{\tilde{\nu}^*}(s,a)}{1+\alpha}\right) \exp\left(-\frac{1-\gamma}{1+\alpha}C - 1\right).
\end{aligned}$$

Therefore,

$$\exp\left(\frac{A_{\tilde{\nu}^*}(s,a)}{1+\alpha}\right) \propto \exp\left(\frac{A_{(\tilde{\nu}^*+C)}(s,a)}{1+\alpha} - 1\right) = \frac{d^*(s,a)}{d^U(s,a)}.$$

D EXPERIMENTAL DETAILS

Implementation detail For fair comparison, we use the same learning rate to train actors of BC and DemoDICE. In DemoDICE, we applied the same regularization coefficient $\alpha = 0.05$ in Equation 18 across all the tasks and dataset configurations. We implement our network architectures for BC and DemoDICE based on the implementation of OptiDICE[1]. For ValueDICE, we use its official implementation[2] without any modification to network architectures or hyperparameters. Based on DAC (Kostrikov et al., 2019), we treat terminal states as absorbing states in ValueDICE and DemoDICE.

For stable discriminator learning, we use gradient penalty regularization on the $r(s, a)$ function, which was proposed in Gulrajani et al. (2017) to enforce 1-Lipschitz constraint. To stabilize critic training, we add gradient L2-norm to the critic loss for the regularization. Detailed hyperparameter configurations used for our main experiments are summarized in Table 1.

Hyperparameters	BC	DemoDICE
γ (discount factor)	0.99	0.99
α (regularization coefficient)	-	0.05
learning rate (actor)	3×10^{-5}	3×10^{-5}
network size (actor)	[256, 256]	[256, 256]
learning rate (critic)	-	3×10^{-4}
network size (critic)	-	[256,256]
learning rate (discriminator)	-	3×10^{-4}
network size (discriminator)	-	[256,256]
gradient L2-norm coefficient (critic)	-	1×10^{-4}
gradient penalty coefficient (discriminator)	-	10
batch size	256	256
# of expert trajectories	1	1
# of training iterations	1,000,000	1,000,000

Table 1: Configurations of hyperparameters used in our experimental results.

E ADDITIONAL EXPERIMENTAL RESULTS

In this section, we report additional experimental results for detailed empirical analysis of our approach. We focus three experiments as follows: (1) asymptotic performance of our approach on HalfCheetah environment for the mixed dataset we considered in Section 5.1, (2) behavior cloning with more diverse β selections which are omitted in Figure 1, (3) ablation study on hyperparameter α for DemoDICE, which is appeared in Equation 18.

E.1 ASYMPTOTIC PERFORMANCE OF DEMODICE ON HALFCHEETAH

To inspect asymptotic behaviors of DemoDICE on our mixed dataset tasks in HalfCheetah environment, we extend the number of training iterations into 2 million with fixing the other hyperparameter settings. As Figure 3 shows, DemoDICE converges to the expert performance for all tasks we considered in Section 5.1.

[1] https://github.com/secury/optidice

[2] https://github.com/google-research/google-research/tree/master/value_dice

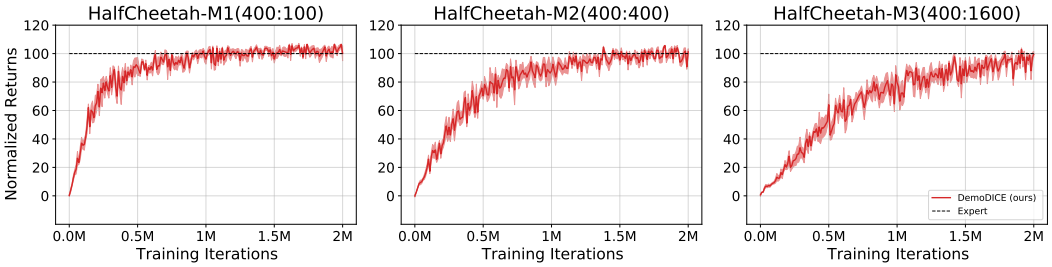

Figure 3: 1 Learning curves of DemoDICE during 2 million training iterations in HalfCheetah mixed dataset tasks (M1, M2, M3).

E.2 Behavior Cloning Performance

In Figure 1, we omitted performance of BC with $\beta = 0.25$ and $\beta = 0.75$ for simplicity. Figure 4 shows performance of entire BC configurations, i.e., $\beta \in \{0, 0.25, 0.5, 0.75, 1.0\}$.

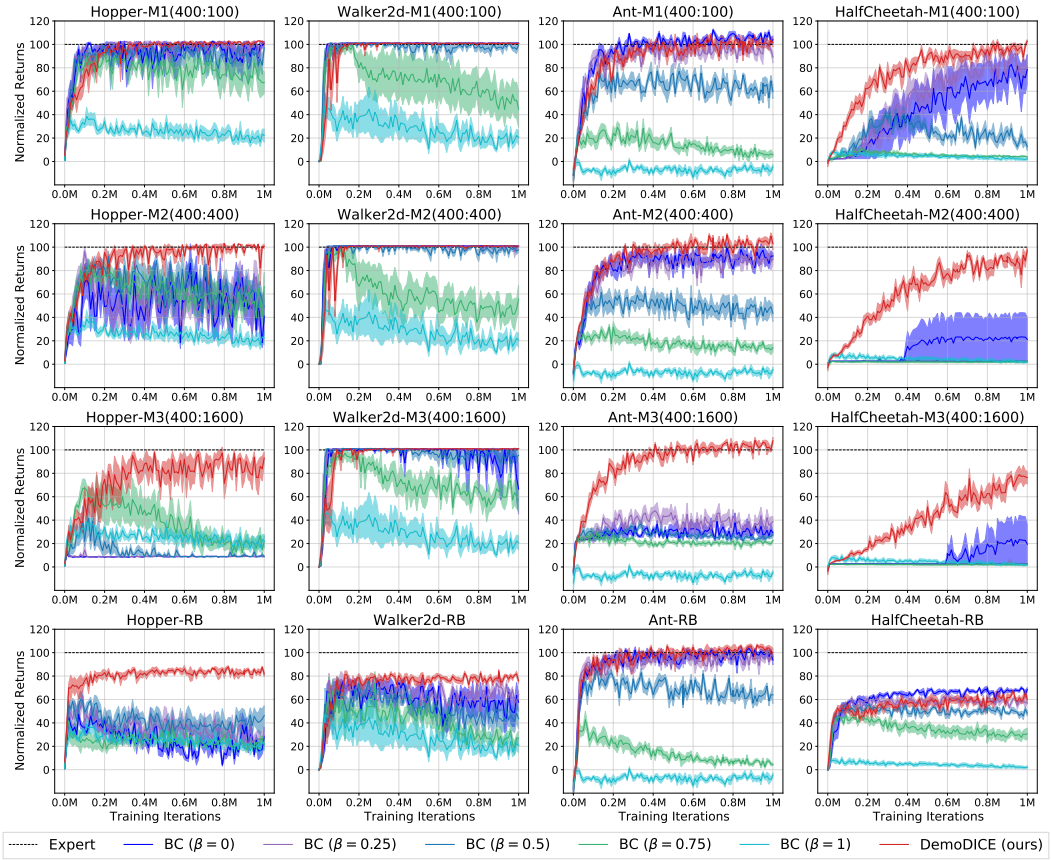

Figure 4: Performance of variants of BC with $\beta \in \{0, 0.25, 0.5, 0.75, 1.0\}$ and DemoDICE both on the mixed datasets (M1, M2, M3) and the replay buffer datasets. We plot the mean and the standard errors (shaded area) of normalized scores over 5 random seeds.

E.3 ABLATION TEST ON HYPERPARAMETER α

As we can notice in Equation 18, DemoDICE introduces a hyperparameter α, which is a regularization coefficient. In this section, we aim to study how does the choice of α effect to training processes and performance of our algorithm. We select $\alpha \in \{10, 1, 0.5, 0.05, 0\}$ and train on M3 datasets and replay buffer datasets by preserving the other settings same as before. Figure 5 describes the results of ablation study.

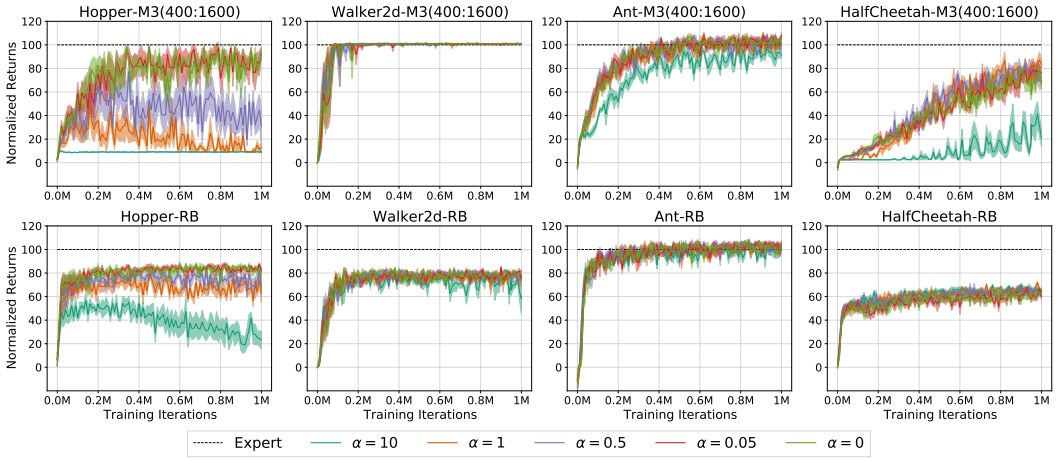

Figure 5: Performance of DemoDICE by varying $\alpha \in \{0, 0.25, 0.5, 0.75, 1.0\}$ on the mixed datasets M3 and the replay buffer datasets. Bold lines and shaded areas indicate the means and the standard errors (shaded area) of normalized scores over 5 random seeds respectively.