VERSATILE OFFLINE IMITATION LEARNING VIA STATE OCCUPANCY MATCHING

Jason Yecheng Ma, Andrew Shen, Dinesh Jayaraman & Osbert Bastani
Department of Computer & Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
jasonyma@seas.upenn.edu

ABSTRACT

We propose State Matching Offline DIstribution Correction Estimation (SMODICE), a novel and versatile algorithm for offline imitation learning (IL) via state-occupancy matching. Without requiring access to expert actions, SMODICE can be effectively applied to three offline IL settings: (i) imitation from observations (IfO), (ii) IfO with dynamics or morphologically mismatched expert, and (iii) example-based reinforcement learning, which we show can be formulated as a state-occupancy matching problem. We show that the SMODICE objective admits a simple optimization procedure through an application of Fenchel duality, reducing a nested optimization problem to a sequence of stable supervised learning problems. We extensively evaluate SMODICE on both gridworld environments as well as on high-dimensional offline benchmarks. Our results demonstrate that SMODICE is effective for all three problem settings and significantly outperforms prior state-of-art.

1 INTRODUCTION

The offline reinforcement learning (RL) framework Lange et al. (2012); Levine et al. (2020) aims to use pre-collected, reusable offline data—without further interaction with the environment—for sample-efficient, scalable, and practical data-driven decision-making. However, this assumes that the offline dataset comes with reward labels, which may not always be possible. To address this, offline *imitation* learning (IL) Zolna et al. (2020); Chang et al. (2021); Kim et al. (2022) has recently been proposed as an alternative where the learning algorithm is provided with a small set of expert demonstrations and a separate set of offline data of unknown quality. The goal is to learn a policy that mimics the provided expert data while avoiding test-time distribution shift Ross et al. (2011) by using the offline dataset.

Expert demonstrations are often much more expensive to acquire than offline data; thus, offline IL benefits significantly from minimizing assumptions about the expert data. In this work, we aim to remove two assumptions about the expert data in current offline IL algorithms: (i) expert action labels must be provided for the demonstrations, and (ii) the expert demonstrations are performed with identical dynamics (same embodiment, actions, and transitions) as the imitator agent. These requirements preclude applications to important practical problem settings, including (i) imitation from observations, (ii) imitation with mismatched expert that obeys different dynamics or embodiment (e.g., learning from human videos), and (iii) learning only from examples of successful outcomes rather than full expert trajectories Eysenbach et al. (2021).

For these reasons, many algorithms for *online* IL have already sought to remove these assumptions Torabi et al. (2018; 2019); Liu et al. (2019); Radosavovic et al. (2020); Eysenbach et al. (2021), but extending them to offline IL remains an open problem.

We propose **S**tate **M**atching **O**ffline **DI**stribution **C**orrection **E**stimation (SMODICE), a general offline IL framework that can be applied to all three problem settings described above. At a high level, SMODICE is based on a state-occupancy matching view of IL; in particular, it optimizes a tractable offline upper bound of the KL-divergence of the state-occupancy d between the imitator π

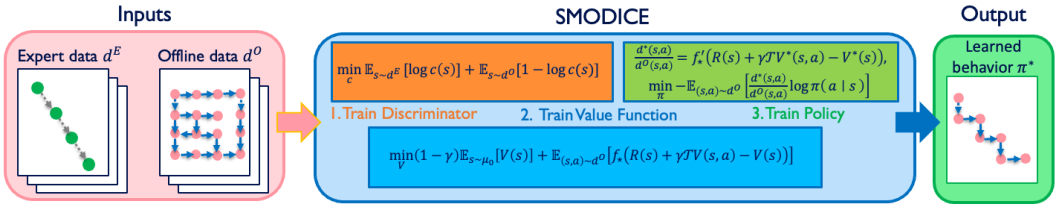
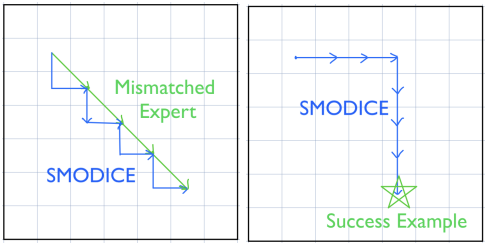


Figure 1: Diagram of SMODICE. First, a state-based discriminator is trained using the offline dataset d^O and expert observations (resp. examples) d^E. Then, the discriminator is used to train the Lagrangian value function. Finally, the value function provides the importance weights for policy training, which outputs the learned policy d^*.

and the expert E:

$$\min_\pi \mathrm{D}_{\mathrm{KL}}(d^\pi(s) \| d^E(s)). \tag{1}$$

This state-occupancy matching objective allows SMODICE to infer the correct actions from the offline data in order to match the state-occupancy of the provided expert demonstrations. This naturally enables imitation when expert actions are unavailable, and even when the expert's embodiment or dynamics are different, as long as there is a shared task-relevant state. Finally, we show that example-based RL Eysenbach et al. (2021), where only examples of successful states are provided as supervision, can be formulated as a state-occupancy matching problem between the imitator and a "teleporting" expert that is able to reach success states in one step. Hence, SMODICE can also be used as an offline example-based RL[1] method without any modification.



(a) Mismatched experts (b) IL from examples

Figure 2: Illustrations of tabular SMODICE.

Naively optimizing the offline upper bound on (1) would result in an actor-critic style IL algorithm akin to prior work Ho & Ermon (2016); Kostrikov et al. (2018; 2020); however, these algorithms suffer from training instability in the offline regime Kumar et al. (2019); Lee et al. (2021); Kim et al. (2022) due to the entangled nature of actor and critic learning, leading to erroneous value bootstrapping Levine et al. (2020). SMODICE bypasses this issue and achieves "actor-free" training by directly estimating the importance weight ratio of the occupancy measures between the optimal policy and the empirical behavior policy of the offline data, leveraging the stationary distribution correction estimation Nachum et al. (2019a); Nachum & Dai (2020) (DICE) paradigm. Specifically, by formulating the policy optimization problem via its dual (i.e., optimizing over the space of valid state-action occupancy distributions) and applying Fenchel duality, SMODICE obtains an unconstrained convex optimization problem over a value function arising from Lagrangian duality, which admits closed-form solutions in the tabular case and can be easily optimized using stochastic gradient descent (SGD) in the deep RL setting. Then, SMODICE projects the optimal value function onto the dual space to extract the optimal importance weights, and learns the optimal policy via weighted Behavior Cloning. Note that SMODICE does not learn a policy until the value function has converged.

Through extensive experiments, we show that SMODICE is effective for all three problem settings we consider and outperforms all state-of-art methods in each respective setting. A benefit of the improved optimization is that SMODICE is substantially more stable compared to prior methods—we obtain all SMODICE results using a *single* set of hyperparameters, modulo a choice of f-divergence which can be tuned *offline*. In contrast, prior methods suffer from much greater performance fluctuation across tasks and settings. Altogether, our proposed method SMODICE can serve as a versatile offline IL algorithm that is suitable for a wide range of assumptions on expert data.

Pedagogical examples. To illustrate SMODICE's versatility, we have applied it to two gridworld tasks, testing offline IL from mismatched experts and examples, respectively. Figure 2(a) shows

[1] We refer to this problem as "offline imitation learning from examples" to unify nomenclature with the other two problems.

an expert agent that can move diagonally in any direction, whereas the imitator can only move horizontally or vertically. In Figure 2(b), only a success state (the star) is provided as supervision. An offline dataset collected by a random agent is given to SMODICE for training in both cases. As shown, SMODICE recovers an optimal policy (i.e. minimum state-occupancy divergence to that of the expert) in both cases. See Appendix D.1 for details.

2 PRELIMINARIES

Markov decision processes. We consider a time-discounted Markov decision process (MDP) Puterman (2014) $\mathcal{M} = (S, A, R, T, \mu_0, \gamma)$ with state space S, action space A, deterministic rewards $R(s, a)$, stochastic transitions $s' \sim T(s, a)$, initial state distribution $\mu_0(s)$, and discount factor $\gamma \in (0, 1]$. A policy $\pi : S \to \Delta(A)$ determines the action distribution conditioned on the state.

The state-action occupancies (also known as stationary distribution) $d^\pi(s, a) : \mathcal{S} \times \mathcal{A} \to [0, 1]$ of π is

$$d^\pi(s, a) \coloneqq (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathrm{Pr}(s_t = s, a_t = a \mid s_0 \sim \mu_0, a_t \sim \pi(s_t), s_{t+1} \sim T(s_t, a_t)) \tag{2}$$

which captures the relative frequency of state-action visitations for a policy π. The state occupancies then marginalize over actions: $d^\pi(s) = \sum_a d^\pi(s, a)$. The state-action occupancies satisfy the single-step transpose Bellman equation:

$$d^\pi(s, a) = (1 - \gamma)\mu_0(s)\pi(a \mid s) + \gamma \cdot \mathcal{T}_\star^\pi d^\pi(s, a), \tag{3}$$

where $\mathcal{T}_\star^\pi$ is the adjoint policy transition operator,

$$\mathcal{T}_\star^\pi d^\pi(s, a) \coloneqq \pi(a \mid s) \sum_{\tilde{s}, \tilde{a}} T(s \mid \tilde{s}, \tilde{a}) d(\tilde{s}, \tilde{a}) \tag{4}$$

Divergences and Fenchel conjugates. Due to space constraint, their definitions are included in Appendix A.

Offline imitation learning. Many imitation learning approaches rely on minimizing the f-divergence between the state-action occupancies of the imitator and the expert Ho & Ermon (2016); Ke et al. (2020); Ghasemipour et al. (2019):

$$\min_\pi \mathrm{D}_f \left(d^\pi(s, a) \| d^E(s, a) \right) \tag{5}$$

In imitation learning, we do not have d^E; instead, we are provided with expert demonstrations $\mathcal{D}^E \coloneqq \{(s^{(i)}, a^{(i)})\}_{i=1}^N$.

In offline imitation learning, the agent further cannot interact with the MDP $\mathcal{M}$; instead, they are given a static dataset of logged transitions $\mathcal{D}^O \coloneqq \{\tau_i\}_{i=1}^M$, where each trajectory $\tau^{(i)} = (s_0^{(i)}, a_0^{(i)}, s_1^{(i)}, ...)$ with $s_0^{(i)} \sim \mu_0$; we denote the empirical state-action occupancies of $\mathcal{D}^O$ as $d^O(s, a)$.

3 THE SMODICE ALGORITHM

SMODICE aims to minimize a state-occupancy matching objective based on the KL-divergence:

$$\min_\pi \mathrm{D}_{\mathrm{KL}}(d^\pi(s) \| d^E(s)) \tag{6}$$

Later, we show that we can optimize any f-divergence that upper bounds KL-divergence, as the conjugate of the KL-divergence may be numerically unstable.

Minimizing (6) requires on-policy samples from π, as the expectation is over d^π. To derive an offline objective, we first derive an upper bound involving the offline dataset.

First, we assume expert coverage of the offline data:

Assumption 1. $d^O(s) > 0$ whenever $d^E(s) > 0$.

This assumption ensures that the offline dataset has coverage over the expert state-marginal, and is necessary for imitation learning to succeed. Whereas prior offline RL approaches Kumar et al. (2020); Ma et al. (2021) assume full coverage of the state-action space, our assumption[2] is considerably weaker since it only requires expert coverage. Given this assumption, we proceed to derive the offline upper bound on state-occupancy matching:

Theorem 1. *Given Assumption 1, we have*

$$\mathrm{D_{KL}}(d^\pi(s)\|d^E(s)) \leq \mathbb{E}_{s\sim d^\pi}\left[\log\left(\frac{d^O(s)}{d^E(s)}\right)\right] + \mathrm{D_{KL}}(d^\pi(s,a)\|d^O(s,a)) \tag{7}$$

Furthermore, for any f-divergence such that $\mathrm{D}_f \geq \mathrm{D_{KL}}$,

$$\mathrm{D_{KL}}(d^\pi(s)\|d^E(s)) \leq \mathbb{E}_{s\sim d^\pi}\left[\log\left(\frac{d^O(s)}{d^E(s)}\right)\right] + \mathrm{D}_f(d^\pi(s,a)\|d^O(s,a)) \tag{8}$$

Proofs are in Appendix A. Intuitively, this says that offline state-occupancy matching can be achieved by matching states in the offline data that resemble expert states (the first term), while remaining in the support of the offline state-action distribution (the second term). Replacing KL-divergence with other divergences can be useful since the conjugate of KL divergence involves a log-sum-exp, which has been found to be numerically unstable in many RL tasks Zhu et al. (2020); Lee et al. (2021); Rudner et al. (2021).

Note that (8) requires samples from d^π, so it still cannot be easily optimized without online interaction. To address this, we first rewrite it as an optimization problem over the space of valid state-action occupancies Puterman (2014):

$$\max_{d(s,a)\geq 0} \mathbb{E}_{s\sim d(s,a)}\left[\log\left(\frac{d^E(s)}{d^O(s)}\right)\right] - \mathrm{D}_f(d(s,a)\|d^O(s,a)) \tag{9}$$

$$\text{s.t.} \sum_a d(s,a) = (1-\gamma)\mu_0(s) + \gamma\mathcal{T}_\star d(s), \forall s \in S \tag{10}$$

where $\mathcal{T}_\star d(s) = \sum_{\bar{s},\bar{a}} T(s \mid \bar{s},\bar{a})d(\bar{s},\bar{a})$; here, (10) ensures that d is the occupancy distribution for some policy. We assume that (9) is *strictly feasible*.

Assumption 2. There exists at least one $d(s,a)$ such that constraints (10) are satisfied and $\forall s \in \mathcal{S}, d(s) > 0$.

This assumption is mild and can be satisfied in practice for any MDP for which every state is reachable from the initial state distribution. Next, we can form the dual of (9):

$$\max_{d(s,a)\geq 0}\min_{V(s)\geq 0} \mathbb{E}_{s\sim d}\left[\log\left(\frac{d^E(s)}{d^O(s)}\right)\right] - \mathrm{D}_f(d(s,a)\|d^O(s,a)) + \sum_s V(s)\left((1-\gamma)\mu_0(s) + \gamma\mathcal{T}_\star d(s) - \sum_a d(s,a)\right) \tag{11}$$

where $V(s)$ are the Lagrangian multipliers. Now, because $\mathcal{T}_\star$ is the adjoint of $\mathcal{T}$, we have the following:

$$\sum_s V(s)\cdot\mathcal{T}_\star d(s) = \sum_{s,a} d(s,a)\cdot(\mathcal{T}V)(s,a) \tag{12}$$

Using this equation, we can write (11) as

$$\max_{d(s,a)\geq 0}\min_{V(s)\geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \mathbb{E}_{(s,a)\sim d}\left[\log\left(\frac{d^E(s)}{d^O(s)}\right) + \gamma\mathcal{T}V(s,a) - V(s)\right] - \mathrm{D}_f(d(s,a)\|d^O(s,a)) \tag{13}$$

We note that the original problem (9) is convex Lee et al. (2021). By Assumption 2, it is strictly feasible, so by strong duality, we can change the order of optimization in (13):

$$= \min_{V(s)\geq 0}\max_{d(s,a)\geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \mathbb{E}_{(s,a)\sim d}\left[(R(s) + \gamma\mathcal{T}V(s,a) - V(s))\right] - \mathrm{D}_f(d(s,a)\|d^O(s,a)) \tag{14}$$

[2]Furthermore, it is not needed in practice, and is only required for our technical development to ensure that all state-occupancy quantities are well-defined (i.e., no division-by-zero).

where $R(s) = \log \frac{d^E(s)}{d^O(s)}$. We briefly discuss how to compute $R(s)$. In the tabular case, $R(s)$ can be computed using empirical estimates of $d^E(s)$ and $d^O(s)$. In the continuous case, we can train a discriminator $c : \mathcal{S} \to (0, 1)$:

$$\min_c \mathbb{E}_{s\sim d^E} [\log c(s)] + \mathbb{E}_{s\sim d^O} [\log 1 - c(s)] \tag{15}$$

The optimal discriminator is $c^\star(s) = \frac{d^O(s)}{d^E(s)+d^O(s)}$ Goodfellow et al. (2014), so we can use $R(s) = -\log\left(\frac{1}{c^\star(s)} - 1\right)$.

Finally, using the Fenchel conjugate, (14) can be reduced to a single unconstrained optimization problem over $V : \mathcal{S} \to \mathbb{R}_{\geq 0}$ that depends on samples from only d^O and not d; we also obtain the importance weight of the state-occupancy of the optimal policy with respect to the offline data.

Theorem 2. *The optimization problem* (14) *is equivalent to*

$$\min_{V(s)\geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \mathbb{E}_{(s,a)\sim d^O} [f_\star (R(s) + \gamma\mathcal{T}V(s,a) - V(s))] \tag{16}$$

Furthermore, given the optimal solution V^*, the optimal state-occupancy importance weights are

$$\frac{d^*(s,a)}{d^O(s,a)} = f'_\star(R + \gamma\mathcal{T}V^*(s,a) - V^*(s)) \tag{17}$$

This result can be viewed as using Fenchel duality to generalize prior DICE-based offline approaches Lee et al. (2021); Kim et al. (2022). In particular, the inner maximization problem in (14) is precisely the Fenchel conjugate of $D_f(d(s,a)\|d^O(s,a))$ at $R(s) + \gamma\mathcal{T}V(s,a) - V(s)$ (compare (14) to (23)). Similarly, (17) can be derived from leveraging the relationship between the optimal solutions of a pair of Fenchel primal-dual problems. This generality allows us to choose problem-specific f-divergences that improve stability during optimization. In Appendix C, we specialize the SMODICE objective for the KL- and χ^2-divergences, which we use in our experiments.

Finally, using the optimal importance weights, we can extract the optimal policy π using weighted Behavior Cloning:

$$\min_\pi -\mathbb{E}_{(s,a)\sim d^*}[\log \pi(a \mid s)] = \min_\pi -\mathbb{E}_{(s,a)\sim d^O}[\xi^*(s,a) \log \pi(a \mid s)] \tag{18}$$

where $\xi^*(s,a) = \frac{d^*(s,a)}{d^O(s,a)}$. Here, $V(s)$ can be viewed as the value function—it is trained by minimizing a convex function of the Bellman residuals and the values of the initial states. Then, it can be used to inform policy learning.

Putting everything together, SMODICE can achieve stable policy learning through a sequence of three *disjoint* supervised learning problems, summarized in Figure 1. The full pseudo-code is in Algorithm 2 in Appendix 2.

SMODICE for Tabular MDPs. An appealing property of SMODICE is that it admits closed-form analytic solution in the tabular case. The derivation is in Appendix D.

4 Offline Imitation Learning from Examples

Next, we describe how SMODICE can be applied to offline imitation learning from examples. Starting from the original problem objective from Eysenbach et al. (2021), we derive a state-occupancy matching objective, enabling us to apply SMODICE without any modification.

Problem setting. We assume given success examples $S^* = \{s^* \sim p_U(s_t \mid e_t = 1)\}$, where $e \in \{0, 1\}$ indicates whether the current state is a success outcome, and offline data $\mathcal{D} = \{(s, a, s')\}$. Here, U is the state distribution of the "user" providing success examples. Then, Eysenbach et al. (2021) proposes the example-based RL objective

$$\arg\max_\pi \log p^\pi(e_{t+} = 1) = \log \mathbb{E}_{s\sim\mu_0} [p^\pi(e_{t+} = 1|s_0)] \tag{19}$$

That is, we want a policy that maximizes the probability of reaching success states in the future. To tackle this problem in the offline setting, our strategy is to convert (19) into an optimization problem over the state-occupancy space.

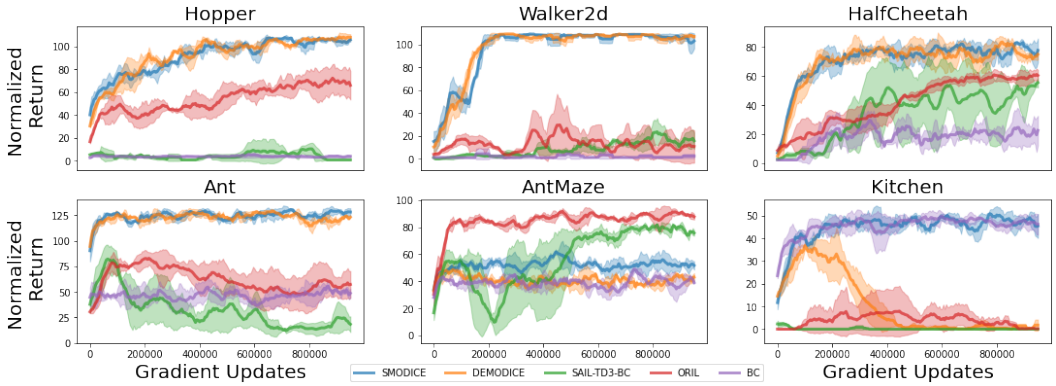


Figure 3: **Offline imitation learning from observations results.**

Derivation. We first transform the problem into state-occupancy space—i.e.,

$$\max_{\pi} \log \mathbb{E}_{s\sim\mu_0}\left[p^{\pi}(e_{t+}=1|s_0)\right] = \max_{d\geq 0} \log \mathbb{E}_{s\sim d(s)}\left[p(e|s)\right] \tag{20}$$

which is valid given that the original objective can be thought of as a regular RL problem with reward function $r(s) = p(e \mid s)$ Eysenbach et al. (2021).

Given this formulation, we can derive a tractable lower bound to (20) through Jensen's inequality and Bayes' rule:

$$\log \mathbb{E}_{s\sim d(s)}\left[p_U(e \mid s)\right] \geq \mathbb{E}_{s\sim d(s)}\left[\log p_U(e \mid s)\right] \geq -\mathrm{D}_{\mathrm{KL}}\left(d(s) \| p_U(s \mid e)\right) + \text{const.}$$

We can optimize the original objective by maximizing this lower bound. Doing so is equivalent to solving $\min_{d\geq 0} \mathrm{D}_{\mathrm{KL}}\left(d(s) \| p_U(s \mid e)\right)$, which is exactly in the form of the state-occupancy matching objective (6) that SMODICE optimizes. Furthermore, this objective admits an intuitive explanation from a purely imitation learning lens. We can think of $p_U(s \mid e)$ as the state-occupancy distribution of an expert agent who can "teleport" to any success state in one time-step. Therefore, we have shown that example-based RL can be understood as a state-occupancy minimization problem between a MDP-dynamics abiding imitator and a teleporting expert agent. Consequently, SMODICE can be used in the offline setting without any algorithmic modification.

5 Experiments

We experimentally demonstrate that SMODICE is effective for offline IL from observations, mismatched experts, and examples. We give additional experimental details in Appendices G, H, and I.

5.1 Offline Imitation Learning from Observations

Datasets. We utilize the D4RL Fu et al. (2021) offline RL dataset. The dataset compositions for all tasks are listed in Table 3 in Appendix G. We consider the following standard Mujoco environments: Hopper, Walker2d, HalfCheetah, and Ant. For each, we take a single expert trajectory from the respective "expert-v2" dataset as the expert dataset and omit the actions. For the offline dataset, following Kim et al. (2022), we use a mixture of small number of expert trajectories (≤ 200 trajectories) and a large number of low-quality trajectories from the "random-v2" dataset (we use the full random dataset, consisting of around 1 million transitions). This dataset composition is particularly challenging as the learning algorithm must be able to successfully distinguish expert from low-quality data in the offline dataset.

(a) Mujoco (b) AntMaze (c) Kitchen

Figure 4: **Evaluation environments.**

We also include two more challenging environments from D4RL: AntMaze and Franka Kitchen. Dur to space constraint, we describe their dataset compositions in Appendix G.

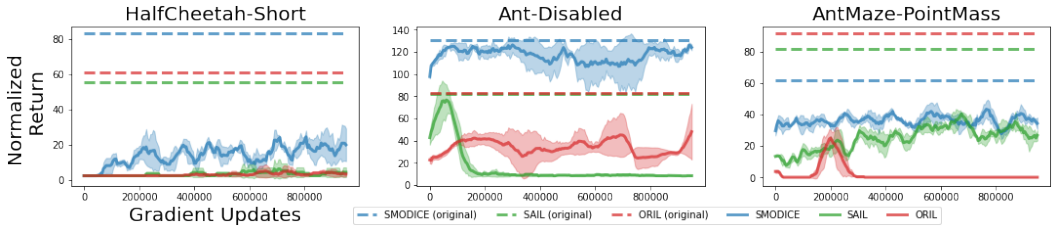


Figure 5: **Offline imitation learning from mismatched experts results.**

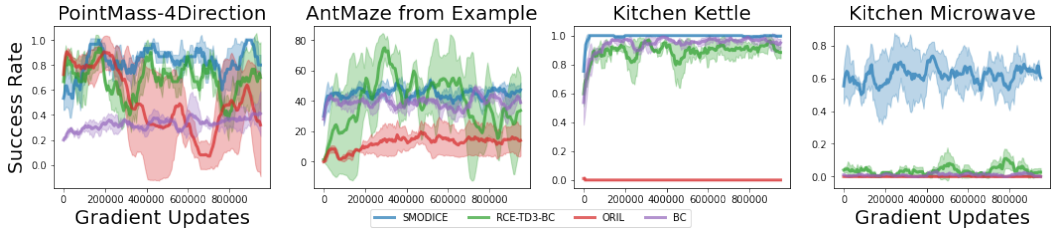


Figure 6: **Offline imitation learning from examples results.**

Method and baselines. We use SMODICE with χ^2-divergence for all tasks (in other problem settings as well) except Hopper, Walker, and Halfcheetah, where we find SMODICE with KL-divergence to perform better; in Appendix E.2, we explain how to choose the appropriate f-divergence *offline* by monitoring SMODICE's policy loss. For comparisons, we consider both IfO and regular offline IL methods, which make use of expert actions. For the former, we compare against (i) **SAIL-TD3-BC**, which combines a state-of-art state-matching based *online* IL algorithm (SAIL) Liu et al. (2019) with a state-of-art offline RL algorithm (TD3-BC) Fujimoto & Gu (2021),[3] (ii) **Offline Reinforced Imitation Learning (ORIL)** Zolna et al. (2020), which adapts GAIL Ho & Ermon (2016) to the offline setting by using an offline RL algorithm for policy optimization; we implement ORIL using the same state-based discriminator as in SMODICE, and TD3-BC as the offline RL algorithm. For the latter, we consider the state-of-art **DEMODICE** Kim et al. (2022) as well as **Behavior Cloning (BC)**. We train all algorithms for 1 million gradient steps and keep track of the normalized score (i.e., 100 is expert performance, 0 is random-action performance) during training; the normalized score is averaged over 10 independent rollouts. All methods are evaluated over 3 seeds, and one standard-deviation confidence intervals are shaded.

Results. As shown in Figure 3, only SMODICE achieves stable and good performance in all six tasks. It achieves (near) expert performance in all the Mujoco environments, performing on-par with DEMODICE and doing so without the privileged information of expert actions. SMODICE's advantage over DEMODICE is more apparent in AntMaze and Kitchen. In the former, SMODICE outperforms BC, while DEMODICE cannot; in the latter, DEMODICE quickly collapses due to its use of KL-divergence, which may be numerically unstable in high-dimensional environments. BC is a strong baseline for tasks where the offline dataset contains (near) expert data (i.e., AntMaze and Kitchen); however, as the dataset becomes more diverse, BC's performance drops significantly.

SAIL-TD3-BC and ORIL both fail to learn in some environments and otherwise converge to a worse policy than SMODICE. The only exception is AntMaze; however, in Appendix G.2, we show that both methods collapse with a more diverse version of the AntMaze offline dataset, indicating that unlike SMODICE, these methods are highly sensitive to the composition of the offline dataset, and work best with task-aligned offline data. The sub-par performances of SAIL and ORIL highlight the challenges of adapting online IL methods to the offline setting; we hypothesize that it is not sufficient to simply equip the original methods (i.e., SAIL and GAIL) with a strong base offline RL algorithm. Together, these results demonstrate that SMODICE is stable, scalable, and robust, and significantly outperforms prior methods. Finally, in Appendix G.2, we ablate SMODICE by zeroing out its discriminator-based reward to validate that SMODICE's empirical performance comes from its ability to discriminate expert data in the offline dataset.

[3]We chose TD3-BC due to its simplicity and stability.

5.2 Offline IL from Mismatched Experts

Datasets and baselines. Due to space constraint, they are described in Appendix H.

Results. The training curves are shown in Figure 5; we illustrate the original maximum performance attained by each method (i.e., using the original expert trajectory, Section 5.1) using dashed lines as points of reference. As can be seen, SMODICE is significantly more robust to mismatched experts than either SAIL-TD3-BC or ORIL. On AntMaze, the task where SAIL-TD3-BC and ORIL originally outperform SMODICE, learning from a PointMass expert significantly deteriorates their performances, and the learned policies are noticably worse than that of SMODICE, which has the smallest performance drop. The other two tasks exhibit similar trends; SMODICE is able to learn an expert level policy for the original Ant embodiment using a disabled Ant expert, and is the only method that shows any progress on the hardest HalfCheetah-Short task. Despite using the same discriminator for reward supervision, SMODICE is substantially more robust than ORIL, likely due to the occupancy-constraint $\mathrm{D}_f(d(s,a)\|d^O(s,a))$ term in its objective (8), which ensures that the learned policy is supported by the offline data as it attempts to match the expert states. In Appendix H.2, we provide additional quantitative analysis of Figure 5.

5.3 Offline Imitation Learning from Examples

Tasks. We use the AntMaze and Kitchen environments and create example-based task variants. For AntMaze, we replace the full demonstration with a small set of success states (i.e., Ant in the goal region) extracted from the offline data. For Kitchen, we consider two subtasks in the environment: Kettle and Microwave. and define task success to be only whether the specified object is correctly placed (instead of all objects as in the original task); the success states are extracted from the offline data accordingly. Examples of the success states are illustrated in Figure 13 in Appendix I. Note that the kitchen dataset contains many trajectories where the kettle is moved first. Thus, the kettle task is easy even for Behavior Cloning (BC), since cloning the offline data can lead to success. This is not the case for the microwave task, making it much more difficult to solve using only success examples. In addition, we introduce the PointMass-4Direction environment. Here, a 2D PointMass agent is tasked with navigating to the middle point of a specified edge of the square that encloses the agent (see Figure 13(a)). The offline dataset is generated using a waypoint navigator controlling the agent to each of the four possible goals and contains equally many trajectories for each goal. At training and evaluation time, we set the left edge to be the desired edge and collect success states from the offline data accordingly. This task is low-dimensional but consists of multi-task offline data, making it challenging for algorithms such as BC that do not solve the example-based RL objective.

Approaches. We make no modification to SMODICE; the only difference is that the discriminator is trained using success states instead of full expert state trajectories. Our main comparison is **RCE-TD3-BC**, which combines RCE Eysenbach et al. (2021), the state-of-art online example-based RL method, and TD3-BC. We also compare against ORIL Zolna et al. (2020), using the same architecture as in Section 5.1. Finally, we also include BC.

Results. As shown in Figure 6, SMODICE is the best performing method on all four tasks and is the only one that can solve the Microwave task. RCE-TD3-BC is able to solve the first three tasks, but achieves worse solutions and exhibits substantial performance fluctuation during training; we posit that the optimization for RCE, which requires alternate updates to a recursive classifier and a policy, is substantially more difficult than that of SMODICE. ORIL is unstable and fails to make progress in most tasks. Interestingly, as in the mismatched expert setting, on AntMaze, ORIL's performance is far below that of SMODICE, despite attaining better results originally (Figure 3). This comparison demonstrates the versatility of SMODICE afforded by its state-occupancy matching objective; in contrast, ORIL treats offline IL from examples as an offline RL task with discriminator-based reward and cannot solve the task. See Appendix I for additional analysis.

6 Conclusion

We have proposed SMODICE, a simple, stable, and versatile algorithm for offline imitation learning from observations, mismatched experts, and examples. Leveraging Fenchel duality, SMODICE optimizes a state-occupancy matching objective that enjoys closed-form tabular solution and stable optimization with deep neural networks. Through extensive experiments, we have shown that SMODICE significantly outperforms prior state-of-art methods in all three settings.

REFERENCES

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

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.

Jonathan D. Chang, Masatoshi Uehara, Dhruv Sreenivas, Rahul Kidambi, and Wen Sun. Mitigating covariate shift in imitation learning via offline data without great coverage, 2021.

Bo Dai, Niao He, Yunpeng Pan, Byron Boots, and Le Song. Learning from conditional distributions via dual embeddings, 2016.

Bo Dai, Ofir Nachum, Yinlam Chow, Lihong Li, Csaba Szepesvári, and Dale Schuurmans. Coindice: Off-policy confidence interval estimation. arXiv preprint arXiv:2010.11652, 2020.

Benjamin Eysenbach, Sergey Levine, and Ruslan Salakhutdinov. Replacing rewards with examples: Example-based policy search via recursive classification. In Thirty-Fifth Conference on Neural Information Processing Systems, 2021. URL https://openreview.net/forum?id=VXeoK3fJZhW.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. arXiv preprint arXiv:2106.06860, 2021.

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods, 2019.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. arXiv preprint arXiv:1910.11956, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. Nature, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning, 2016.

Liyiming Ke, Sanjiban Choudhury, Matt Barnes, Wen Sun, Gilwoo Lee, and Siddhartha Srinivasa. Imitation learning as f-divergence minimization, 2020.

Geon-Hyeong Kim, Seokin Seo, Jongmin Lee, Wonseok Jeon, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. DemoDICE: Offline imitation learning with supplementary imperfect demonstrations. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=BrPdX1bDZkQ.

Kuno Kim, Yihong Gu, Jiaming Song, Shengjia Zhao, and Stefano Ermon. Domain adaptive imitation learning. In International Conference on Machine Learning, pp. 5286–5295. PMLR, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

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. arXiv preprint arXiv:1809.02925, 2018.

Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=Hyg-JC4FDr.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. arXiv preprint arXiv:1906.00949, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. arXiv preprint arXiv:2006.04779, 2020.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In Reinforcement learning, pp. 45–73. Springer, 2012.

Jongmin Lee, Wonseok Jeon, Byung-Jun Lee, Joelle Pineau, and Kee-Eung Kim. Optidice: Offline policy optimization via stationary distribution correction estimation. arXiv preprint arXiv:2106.10783, 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.

Fangchen Liu, Zhan Ling, Tongzhou Mu, and Hao Su. State alignment-based imitation learning. arXiv preprint arXiv:1911.10947, 2019.

Yecheng Ma, Dinesh Jayaraman, and Osbert Bastani. Conservative offline distributional reinforcement learning. Advances in Neural Information Processing Systems, 34, 2021.

Ofir Nachum and Bo Dai. Reinforcement learning via fenchel-rockafellar duality, 2020.

Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. arXiv preprint arXiv:1906.04733, 2019a.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience, 2019b.

Martin L Puterman. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014.

Ilija Radosavovic, Xiaolong Wang, Lerrel Pinto, and Jitendra Malik. State-only imitation learning for dexterous manipulation, 2020.

Dripta S. Raychaudhuri, Sujoy Paul, Jeroen van Baar, and Amit K. Roy-Chowdhury. Cross-domain imitation from observations, 2021.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011.

Tim G. J. Rudner, Cong Lu, Michael Osborne, Yarin Gal, and Yee Whye Teh. On pathologies in KL-regularized reinforcement learning from expert demonstrations. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), Advances in Neural Information Processing Systems, 2021. URL https://openreview.net/forum?id=sS8rRmgAatA.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation, 2018.

Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation, 2019.

Chao Yang, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Huaping Liu, Junzhou Huang, and Chuang Gan. Imitation learning from observations by minimizing inverse dynamics disagreement. arXiv preprint arXiv:1910.04417, 2019.

Ruiyi Zhang*, Bo Dai*, Lihong Li, and Dale Schuurmans. Gendice: Generalized offline estimation of stationary values. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=HkxlcnVFwB.

Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. Off-policy imitation learning from observations. Advances in Neural Information Processing Systems, 33, 2020.

Konrad Zolna, Alexander Novikov, Ksenia Konyushkova, Caglar Gulcehre, Ziyu Wang, Yusuf Aytar, Misha Denil, Nando de Freitas, and Scott Reed. Offline learning from demonstrations and unlabeled experience. arXiv preprint arXiv:2011.13885, 2020.

A PROOFS

A.1 PRELIMINARIES

Definition 1 (f-divergence)**.** Given a continuous, convex function f and two probability distributions $p, q \in \Delta(\mathcal{X})$ over a domain $\mathcal{X}$, the f-divergence of p at q is

$$\mathrm{D}_f(p\|q) = \mathbb{E}_{x\sim q}\left[f\left(\frac{p(x)}{q(x)}\right)\right] \tag{21}$$

A common f-divergence in machine learning is the KL-divergence, which corresponds to $f(x) = x\log x$. Now, we introduce Fenchel conjugate for f-divergences.

Definition 2 (Fenchel conjugate)**.** Given a vector space Ω with inner-product $\langle\cdot,\cdot\rangle$, the *Fenchel conjugate* $f_\star : \Omega_\star \to \mathbb{R}$ of a function $f : \Omega \to \mathbb{R}$ is

$$f_\star(y) \coloneqq \max_{x\in\Omega} \langle x, y\rangle - f(x) \tag{22}$$

For an f-divergence, under mild realizability assumptions Dai et al. (2016) on f, the Fenchel conjugate of $D_f(p\|q)$ at $y : \mathcal{X} \to \mathbb{R}$ is

$$\mathrm{D}_{\star,f}(y) = \max_{p\in\Delta(\mathcal{X})} \mathbb{E}_{x\sim p}[y(x)] - \mathrm{D}_f(p\|q) = \mathbb{E}_{x\sim q}[f_\star(y(x))] \tag{23}$$

A.2 TECHNICAL LEMMAS

Lemma 1. *We have*

$$\mathrm{D}_{\mathrm{KL}}(d^\pi(s)\|d^E(s)) \le \mathrm{D}_{\mathrm{KL}}(d^\pi(s,a)\|d^E(s,a))$$

Proof. We first state and prove a related lemma, which first appeared in Yang et al. (2019).

Lemma 2.

$$\mathrm{D}_{\mathrm{KL}}\left(d^\pi(s,a,s')\|d^E(s,a,s')\right) = \mathrm{D}_{\mathrm{KL}}\left(d^\pi(s,a)\|d^E(s,a)\right).$$

Proof.

$$\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}\left(d^\pi(s,a,s')\|d^E(s,a,s')\right)\\
=&\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}} d^\pi(s,a,s')\log\frac{d^\pi(s,a)\cdot T(s'\mid s,a)}{d^E(s,a)\cdot T(s'\mid s,a)}ds'dads\\
=&\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}} d^\pi(s,a,s')\log\frac{d^\pi(s,a)}{d^E(s,a)}ds'dads\\
=&\int_{\mathcal{S}\times\mathcal{A}} d^\pi(s,a)\log\frac{d^\pi(s,a)}{d^E(s,a)}dads\\
=&\mathrm{D}_{\mathrm{KL}}\left(d^\pi(s,a)\|d^E(s,a)\right)
\end{aligned}$$

□

Using this result, we can show the desired upper bound:

$$
\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}\left(d^{\pi}(s,a) \| d^{E}(s,a)\right) \\
=&\mathrm{D}_{\mathrm{KL}}\left(d^{\pi}(s,a,s') \| d^{E}(s,a,s')\right) \\
=&\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}} d^{\pi}(s,a,s') \log \frac{d^{\pi}(s,a)\cdot T(s' \mid s,a)}{d^{E}(s,a)\cdot T(s' \mid s,a)} ds' dads \\
=&\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}} d^{\pi}(s)\pi(a \mid s)T(s' \mid s,a) \log \frac{d^{\pi}(s,a)\cdot T(s' \mid s,a)}{d^{E}(s,a)\cdot T(s' \mid s,a)} ds' dads \\
=&\int d^{\pi}(s)\pi(a \mid s)T(s' \mid s,a) \log \frac{d^{\pi}(s)}{d^{E}(s)} ds' dads + \int d^{\pi}(s)\pi(a \mid s)T(s' \mid s,a) \log \frac{\pi(a \mid s)T(s' \mid s,a)}{\pi^{E}(a \mid s)T(s' \mid s,a)} ds' dads \\
=&\int d^{\pi}(s) \log \frac{d^{\pi}(s)}{d^{E}(s)} ds + \int d^{\pi}(s)\pi(a \mid s) \log \frac{\pi(a \mid s)}{\pi^{E}(a \mid s)} dads \\
=&\mathrm{D}_{\mathrm{KL}}\left(d^{\pi}(s) \| d^{E}(s)\right) + \mathrm{D}_{\mathrm{KL}}\left(\pi(a \mid s) \| \pi^{E}(a \mid s)\right) \\
\geq&\mathrm{D}_{\mathrm{KL}}\left(d^{\pi}(s) \| d^{E}(s)\right)
\end{aligned}
$$

□

A.3 Proof of Theorem 1

Proof.

$$
\begin{aligned}
&D_{\mathrm{KL}}\left(d^{\pi}(s) \| d^{E}(s)\right) \\
=&\int d^{\pi}(s) \log \frac{d^{\pi}(s)}{d^{E}(s)} \cdot \frac{d^{O}(s)}{d^{O}(s)} ds, \quad \text{we assume that } d^{O}(s) > 0 \text{ whenever } d^{E}(s) > 0. \\
=&\int d^{\pi}(s) \log \frac{d^{O}(s)}{d^{E}(s)} ds + \int d^{\pi}(s) \log \frac{d^{\pi}(s)}{d^{O}(s)} ds \\
\leq&\mathbb{E}_{s\sim d^{\pi}}\left[\log \frac{d^{O}(s)}{d^{E}(s)}\right] + \mathrm{D}_{\mathrm{KL}}\left(d^{\pi}(s,a) \| d^{E}(s,a)\right)
\end{aligned}
$$

where the last step follows from Lemma 1. Then, for any $\mathrm{D}_f \geq \mathrm{D}_{\mathrm{KL}}$, we have that

$$
D_{\mathrm{KL}}\left(d^{\pi}(s) \| d^{E}(s)\right) \leq \mathbb{E}_{s\sim d^{\pi}}\left[\log \frac{d^{O}(s)}{d^{E}(s)}\right] + \mathrm{D}_f\left(d^{\pi}(s,a) \| d^{E}(s,a)\right)
$$

□

A.4 Proof of Theorem 2

Proof. We begin with

$$
\min_{V(s)\geq 0} \max_{d(s,a)\geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \mathbb{E}_{(s,a)\sim d}\left[(R(s) + \gamma\mathcal{T}V(s,a) - V(s))\right] - \mathrm{D}_f(d(s,a) \| d^{O}(s,a)) \tag{24}
$$

We have that

$$
\min_{V(s)\geq 0} \max_{d(s,a)\geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \mathbb{E}_{(s,a)\sim d}\left[(R(s) + \gamma\mathcal{T}V(s,a) - V(s))\right] - \mathrm{D}_f(d(s,a) \| d^{O}(s,a)) \tag{25}
$$

$$
= \min_{V(s)\geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \max_{d(s,a)\geq 0} + \mathbb{E}_{(s,a)\sim d}\left[(R(s) + \gamma\mathcal{T}V(s,a) - V(s))\right] - \mathrm{D}_f(d(s,a) \| d^{O}(s,a)) \tag{26}
$$

$$
= \min_{V(s)\geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \mathbb{E}_{(s,a)\sim d^{O}}\left[f_{\star}\left(R(s) + \gamma\mathcal{T}V(s,a) - V(s)\right)\right] \tag{27}
$$

where the last step follows from recognizing that the inner-maximization is precisely the Fenchel conjugate of $D_f(d(s,a) \| d^{O}(s,a))$ at $R(s) + \gamma\mathcal{T}V(s,a) - V(s)$.

To show the relationship among $V^{\star}$ and $\xi^{\star}$, we recognize that (27) and (9) are a pair of Fenchel primal-dual problems.

Lemma 3.

$$\min_{V(s)\geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \mathbb{E}_{(s,a)\sim d^O}\left[f_\star\left(R(s)+\gamma\mathcal{T}V(s,a)-V(s)\right)\right]$$

is the Fenchel dual to

$$\max_{d(s,a)\geq 0} \mathbb{E}_{s\sim d}\left[\log\left(\frac{d^E(s)}{d^O(s)}\right)\right] - \mathrm{D}_f(d(s,a)\|d^O(s,a)) \tag{28}$$

$$\textit{s.t.} \sum_a d(s,a) = (1-\gamma)\mu_0(s) + \gamma\mathcal{T}_\star d(s), \forall s\in S \tag{29}$$

Proof. We define the indicator function $\delta_{\mathcal{X}}(x)$ as

$$\delta_{\mathcal{X}}(x) = \begin{cases} 0 & x\in\mathcal{X} \\ \infty & \text{otherwise} \end{cases}$$

Then, we define $g:\mathbb{R}^{|\mathcal{S}|}\to\mathbb{R}$ as $g(\cdot) \coloneqq \delta_{\{(1-\gamma)\mu_0\}}(\cdot)$. Then, it can be shown that the Fenchel conjugate of g is $g_\star(\cdot) = (1-\gamma)\mathbb{E}_{\mu_0}[\cdot]$. In addition, we denote $h(\cdot) \coloneqq \mathrm{D}+f(\cdot\|d^O)$; then, $h_\star(\cdot) = \mathbb{E}_{(s,a)\sim d^O}[f_\star(\cdot)]$. Finally, define matrix operator $A \coloneqq \gamma\mathcal{T}_\star - I$. Using these notations, we can write (27) as

$$\min_V g_\star(V) + h_\star(A_\star V + R) \tag{30}$$

Then, we proceed to derive the Fenchel dual of (30):

$$\begin{aligned}
&\min_V g_\star(V) + h_\star(A_\star V + R) && (31)\\
=&\min_V \max_d g_\star(V) + \langle d, A_\star V + R\rangle - h(d) && (32)\\
=&\min_V \max_d g_\star(V) + \langle d, A_\star V\rangle + \langle d, R\rangle - h(d) && (33)\\
=&\max_d \left(\min_V g_\star(V) + \langle d, A_\star V\rangle\right) + \langle d, R\rangle - h(d) && (34)\\
=&\max_d \left(\min_V g_\star(V) + \langle Ad, V\rangle\right) + \langle d, R\rangle - h(d) && (35)\\
=&\max_d \left(\max_V -g_\star(V) + \langle -Ad, V\rangle\right) + \langle d, R\rangle - h(d) && (36)\\
=&\max_d g(-Ad) + \langle d, R\rangle - h(d) && (37)
\end{aligned}$$

where (32) follows applying Fenchel conjugacy to $h_\star$, (34) follows from strong duality, (35) follows from the property of an adjoint operator, and (37) follows from applying Fenchel conjugacy to $g_\star$. Here, we recognize that (37) is precisely the optimization problem (28)-(29), where we have moved the constraint (29) to the objective as the indicator function $g(-Ad)$:

$$\begin{aligned}
&g(-Ad) = \delta_{\{(1-\gamma)\mu_0\}}\left(d-\gamma\mathcal{T}_\star d\right)\\
\Leftrightarrow &\sum_a d(s,a) = (1-\gamma)\mu_0(s) + \gamma\mathcal{T}_\star d(s), \forall s\in S
\end{aligned}$$

□

Giving Lemma 3, we use the fact that d^* and V^* admit the following relationship:

$$d^* = h'_\star(-A_\star V^* + R) \tag{38}$$

This follows from the characterization of the optimal solutions for a pair of Fenchel primal-dual problems with convex g, h and linear operator A Nachum & Dai (2020). In this case, assuming that we can exchange the order of expectation and derivative (e.g, conditions of Dominated Convergence Theorem hold), we have

$$d^* = \mathbb{E}_{(s,a)\sim d^O}\left[f_\star\left((R(s)+\gamma\mathcal{T}V(s,a)-V(s)\right)\right], \tag{39}$$

or equivalently,

$$d^*(s,a) = f_\star\left(R(s)+\gamma\mathcal{T}V(s,a)-V(s)\right)\cdot d^O(s,a), \forall s,a\in\mathcal{S}\times\mathcal{A}, \tag{40}$$

as desired. □

B Related Work

Offline imitation learning. The closest work is concurrent work, DEMODICE Kim et al. (2022), a state-action based offline IL method, also using the DICE paradigm to estimates the occupancy ratio between the expert and the imitator; we overview the DICE literature in Appendix B. Due to its dependence on expert actions, DEMODICE cannot be applied to the three problem settings we study. At a technical level, a key limitation of DEMODICE is that it does not exploit the form of general Fenchel duality and only support the KL-divergence, forgoing other f-divergences that can lead to more stable optimization Ghasemipour et al. (2019); Ke et al. (2020); Zhu et al. (2020). Another related work is ORIL Zolna et al. (2020), which adapts GAIL Ho & Ermon (2016) to the offline setting. Finally, there has been recent work learning a pessimistic dynamics model using the offline dataset and then performs imitation learning by minimizing the state-action occupancy divergence with respect to the expert inside this learned model Chang et al. (2021). As with DEMODICE, this approach requires expert actions and cannot be applied to the settings we study.

Imitation from observations, imitation with mismatched experts, and example-based RL All three of these problems have been studied in the online setting. IfO is often achieved through training an additional inverse dynamics model to infer the expert actions Torabi et al. (2018; 2019); Liu et al. (2019); Radosavovic et al. (2020); in contrast, SMODICE matches the expert observations by identifying the correct actions supported in the offline data. To handle experts with dynamics mismatch, some work explicitly learns a correspondence between the expert and the imitator MDPs Kim et al. (2020); Raychaudhuri et al. (2021); however, these approaches make much stronger assumptions on access to the expert MDP that are difficult to satisfy in the offline setting, such as demonstrations from auxillary tasks. In contrast, SMODICE falls under the category of state-only imitation learning Liu et al. (2019); Radosavovic et al. (2020), which overcomes expert dynamics differences by only matching the shared task-relevant state space (e.g., xy coordinates for locomotion tasks). Finally, example-based RL was first studied in Eysenbach et al. (2021); they introduce a recursive-classifier based off-policy actor critic method to solve it. By casting this problem as state-occupancy matching between an imitator and a "teleporting" expert agent, SMODICE can solve the offline variant of this problem without modification.

Stationary distribution correction estimation. Estimating the optimal policy's stationary distribution using off-policy data was introduced by Nachum et al. (2019a) as the DICE trick. This technique has been shown to be effective for off-policy evaluation Nachum et al. (2019a); Zhang* et al. (2020); Dai et al. (2020), policy optimization Nachum et al. (2019b); Lee et al. (2021), online imitation learning Kostrikov et al. (2020); Zhu et al. (2020), and concurrently, offline imitation learning Kim et al. (2022). Within the subset of DICE-based policy optimization methods, none has tackled state-occupancy matching or directly apply Fenchel Duality to its full generality to arrive at the form of value function objective we derive.

C SMODICE with common f-divergences

Example 1 (SMODICE with χ^2-divergence)**.** *Suppose $f(x) = \frac{1}{2}(x-1)^2$, corresponding to χ^2-divergence. Then, we can show that $f_\star(x) = \frac{1}{2}(x+1)^2$ and $f'_\star(x) = x+1$. Hence, the SMODICE objective amounts to*

$$\min_{V(s)\geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \frac{1}{2}\mathbb{E}_{(s,a)\sim d^O}\left[\left(R(s) + \gamma\mathcal{T}V(s,a) - V(s) + 1\right)^2\right] \tag{41}$$

and

$$\xi^*(s,a) = \frac{d^*(s,a)}{d^O(s,a)} = \max\left(0, R(s,a) + \gamma\mathcal{T}V^*(s,a) - V^*(s) + 1\right) \tag{42}$$

Example 2 (SMODICE with KL-divergence)**.** *We have $f(x) = x\log x$. Using the fact that the conjugate of the negative entropy function, restricted to the probability simplex, is the log-sum-exp function Boyd et al. (2004), it follows that $\mathrm{D}_{\star,f}(y) = \log \mathbb{E}_{x\sim q}[\exp y(x)]$. Hence, the KL-divergence SMODICE objective is*

$$\min_{V(s)\geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \frac{1}{2}\log \mathbb{E}_{(s,a)\sim d^O}\left[\exp\left(R(s) + \gamma\mathcal{T}V(s,a) - V(s)\right)\right] \tag{43}$$

and

$$\xi^*(s,a) = \frac{d^*(s,a)}{d^O(s,a)} = \exp\left(R + \gamma \mathcal{T} V^*(s,a) - V^*(s)\right) \tag{44}$$

D SMODICE FOR TABULAR MDPS

In this section, we derive the closed-form expression of SMODICE for tabular MDPs. For simplicity, we assume that the expert state occupancies are given, $d^E(s) \in \Delta(|\mathcal{S}|)$. A behavior policy π_b is used to collect the offline dataset $\mathcal{D}^O$. Then, we can construct a surrogate MDP $\hat{\mathcal{M}}$ using maximum likelihood estimation (i.e., $\hat{T}(s,a,s') = \frac{n(s,a,s')}{n(s,a)}$). Using $\hat{\mathcal{M}}$, we can extract the empirical estimate of the behavior policy occupancies $d^O \in \Delta(|\mathcal{S}||\mathcal{A}|)$ using linear programming. Then, we can define the reward vector $R \in \mathbb{R}_+^{|\mathcal{S}|}$ as $R(s) = \log \frac{d^E(s)}{d^O(s)}$. Using the χ^2-divergence version of SMODICE, we can write down the objective for $V(s) \in \mathbb{R}_+^{|\mathcal{S}|}$:

$$\min_{V(s)\geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \frac{1}{2}\mathbb{E}_{(s,a)\sim d^O}\left[\left(R(s) + \gamma\mathcal{T}V(s,a) - V(s) + 1\right)^2\right] \tag{45}$$

We rewrite this expression in vector-matrix form to derive the closed-form solution. To this end, we define $\mathcal{T} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}|}$ and $\mathcal{B} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}|}$ such that $(\mathcal{T}V)(s,a) = \sum_{s'} T(s'|s,a)V(s')$ and $(\mathcal{B}V)(s,a) = V(s)$. Additionally, we denote $\mu_0 \in \Delta(|\mathcal{S}|)$ and $D = \mathrm{diag}(d^O) \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}||\mathcal{A}|}$. Then, we can rewrite (45):

$$\begin{aligned}
&\min_{V(s)\geq 0} (1-\gamma)\mathbb{E}_{s\sim\mu_0}[V(s)] + \frac{1}{2}\mathbb{E}_{(s,a)\sim d^O}\left[\left(R(s) + \gamma\mathcal{T}V(s,a) - V(s) + 1\right)^2\right] \\
\Rightarrow &\min_{V(s)} (1-\gamma)\mu_0^\top V + \frac{1}{2}\mathbb{E}_{(s,a)\sim d^O}\left[\left(\underbrace{\mathcal{B}R(s,a) + \gamma\mathcal{T}V(s,a) - \mathcal{B}V(s,a)}_{r_V(s,a)} + 1\right)^2\right] \\
\Rightarrow &\min_{V(s)} (1-\gamma)\mu_0^\top V + \frac{1}{2}(r_V + I)^\top D(r_V + I)
\end{aligned} \tag{46}$$

where $r_V \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ and I is the all-one vector in $\mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$. Denoting $J(V) := (1-\gamma)\mu_0^\top V + \frac{1}{2}(r_V + I)^\top D(r_V + I)$, it is clear that $J(V)$ is a convex program in V. Therefore, we can find its optimal solution by solving the first-order stationary point. We have:

$$\begin{aligned}
\frac{\partial J(V)}{\partial V} =& \frac{\partial}{\partial V}\left((1-\gamma)\mu_0^\top V + \frac{1}{2}(r_V + I)^\top D(r_V + I)\right) \\
=& \frac{\partial}{\partial V}\left((1-\gamma)\mu_0^\top V + \frac{1}{2}r_V^\top D r_V + r_V^\top D I + I^\top D I\right) \\
=& (1-\gamma)\mu_0 + (\gamma\mathcal{T} - \mathcal{B})^\top D r_V + (\gamma\mathcal{T} - \mathcal{B})^\top D I \\
=& (1-\gamma)\mu_0 + (\gamma\mathcal{T} - \mathcal{B})^\top D(\mathcal{B}R + (\gamma\mathcal{T} - \mathcal{B})V) + (\gamma\mathcal{T} - \mathcal{B})^\top D I
\end{aligned}$$

Then, by setting this expression to zero and solving for V gives the optimal V^*:

$$\begin{aligned}
(\gamma\mathcal{T} - \mathcal{B})^\top D(\gamma\mathcal{T} - \mathcal{B})V &= (\gamma - 1)\mu_0 + (\mathcal{B} - \gamma\mathcal{T})^\top D(I + BR) \\
\Rightarrow V^* &= \left((\gamma\mathcal{T} - \mathcal{B})^\top D(\gamma\mathcal{T} - \mathcal{B})\right)^{-1}\left((\gamma - 1)\mu_0 + (\mathcal{B} - \gamma\mathcal{T})^\top D(I + BR)\right)
\end{aligned} \tag{47}$$

and we can recover $\xi^*(s,a) = \frac{d^*(s,a)}{d^O(s,a)}$:

$$\xi^*(s,a) = \mathcal{B}R(s,a) + \gamma\mathcal{T}V^*(s,a) - \mathcal{B}V^*(s,a) + 1 \tag{48}$$

Pythonic pseudo-code using NumPy Harris et al. (2020) is given in Algorithm 1.

D.1 GRIDWORLD EXPERIMENTS

In this subsection, we provide more experimental details and analysis of the tabular SMODICE experiments shown in Figure 1.

To generate the offline dataset, a random policy (i.e., a policy that chooses each action with equal probabilities) is executed in the MDP for 10000 epsiodes. We use this dataset to compute the approximate MDP. Then, this MDP is used as an input to SMODICE (see Algorithm 1). The data collection procedure for the offline imitation learning from examples setting is identical.

Offline IL from mismatched experts. In this task, we consider an expert agent that can move one grid cell diagonally in any direction, whereas the imitator is only able to move one grid cell horizontally or vertically. The expert policy is shown in black in Figure 2(a). Using purely an offline dataset collected by a random agent, we compute the closed-form tabular SMODICE solution (**??**) using Algorithm 1 and obtain the zig-zagging policy shown in blue. Indeed, this solution is one of the two correct solutions that minimize the state-occupancy divergence (the other one mirrors this path along the expert demo), while being feasible under the imitator dynamics.

Offline IL from examples. We arbitrarily select a state to be the success state denoted by the green star in Figure 2(b). In this case, the expert's state occupancies is simply a one-hot vector with weight 1 at the success state. Then, we again use the tabular version of SMODICE to compute the policy whose state occupancies is as close to this one-hot vector as possible; the solution is illustrated in blue. As can be seen, this policy successfully reaches the goal. Furthermore, it is easy to see that in this task, a policy that minimizes state-occupancy divergence to the expert (i.e., the one-hot vector) is one that reaches the goal with the fewest steps. The policy learned by SMODICE is indeed among the set of optimal policies.

Furthermore, we compute the state occupancies of all states in the gridworld. For the success state, $d(s) \approx 0.915$, whereas the second largest state occupancy is 0.01. This validates the intuition that $\sum_{s \in \mathcal{S}^*} d^\pi(s) \gg \sum_{s \notin \mathcal{S}^*} d^\pi(s)$.

E SMODICE WITH DEEP NEURAL NETWORKS

For high-dimensional MDP with continuous state and action spaces, we instantiate SMODICE using deep neural networks. In particular, we parameterize V_θ and π_ϕ using DNNs with weights θ and ϕ, respectively.

Remark. We note that the sample-based estimation of Equation (16) (Line 9) is biased because $\mathcal{T}V$ is itself an expectation that is inside a (non-linear) convex function f Baird (1995); however, as in several prior works Nachum et al. (2019b); Nachum & Dai (2020); Lee et al. (2021), we do not find this biased estimate to impact empirical performance and keep it for simplicity.

E.1 HYPERPARAMETERS AND ARCHITECTURE

We use the same hyperparameters for all SMODICE experiments in this paper modulo the choice of f-divergences (explained in the next section). In terms of architecture, we use a simple 2-layer ReLU network with hidden size 256 to parameterize the value network. For the policy network, we use the same architecture to parameterize a Gaussian output distribution; the mean and the log standard deviation are ouputs of two separate heads. In addition, we use an tanh function on the Gaussian samples to enforce bounded actions, as in Haarnoja et al. (2018). The discriminator uses the same architecture. Table 1 summarizes the hyperparameters as well as the architecture.

E.2 CHOOSING f-DIVERGENCE IN PRACTICE

In our experiments, SMODICE is implemented using χ^2-divergence for all tasks except Hopper, Walker2d, and HalfCheetah. Here, we show that a suitable choice of f-divergence can be chosen *offline* by observing the initial direction of the SMODICE policy loss on the offline dataset. More specifically, on the environments in which SMODICE exhibited largest performance discrepancies between using KL-divergence or χ^2-divergence, we have found that SMODICE returns are *negatively* correlated with the policy loss. As shown in Figure 7, the poor performing variant of SMODICE

Table 1: SMODICE Hyperparameters.

	Hyperparameter	Value
SMODICE Hyperparameters	Optimizer	Adam Kingma & Ba (2014)
	Critic learning rate	3e-4
	Discriminator learning rate	3e-4
	Actor learning rate	3e-5
	Mini-batch size	256
	Discount factor	0.99
	Actor Mean Clipping	(-7.24, 7.24)
	Actor Log Std. Clipping	(-5,2)
Architecture	Discriminator hidden dim	256
	Discriminator hidden layers	2
	Discriminator activation function	Tanh
	Critic hidden dim	256
	Critic hidden layers	2
	Critic activation function	ReLU
	Actor hidden dim	256
	Actor hidden layers	2
	Actor activation function	ReLU

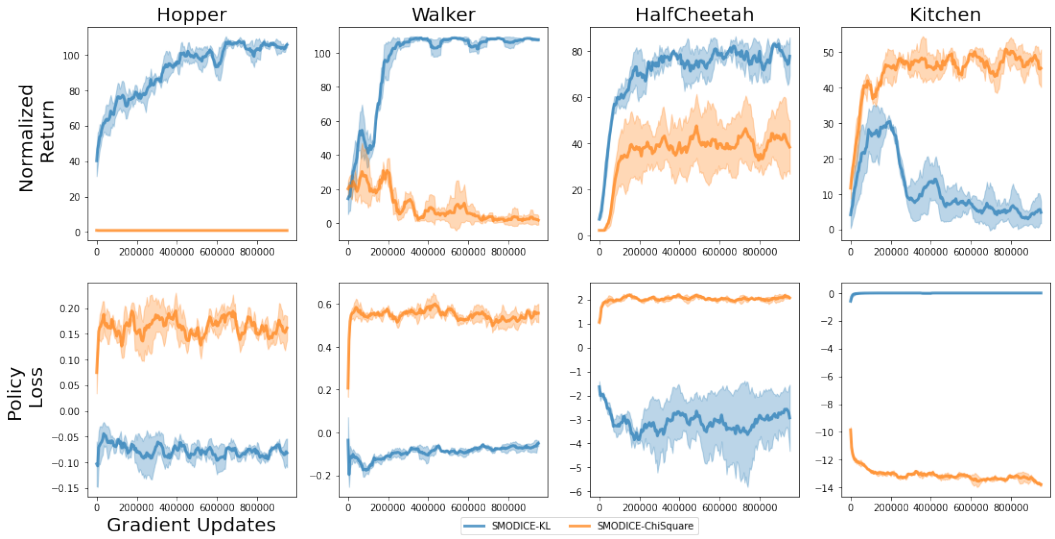


Figure 7: SMODICE returns are negatively correlated with the direction of its policy losses.

always has a policy loss that initially jumps and vice-versa. This makes intuitive sense given the composition of the offline datasets, which is a mix of small amount of expert data with a large amount of poor quality data (see Appendix G for more details). When SMODICE fails to pick out the expert data, which is often narrowly distributed, then it must have assigned relatively higher importance weights to the lower quality data, which is more diverse. This creates a more difficult supervised learning task, leading to higher training loss for the policy. Therefore, in practice, we recommend monitoring SMODICE's initial policy loss direction to determine whether the current f-divergence will lead to good performance and make changes accordingly.

F BASELINES

TD3-BC. Many of our baselines are implemented using TD3-BC as their offline policy optimizer. We use the default hyperparameters for TD3-BC provided by Fujimoto & Gu (2021), shown in Table 2.

Implementation Details. We use the official PyTorch implementation of TD3-BC, publicly available at `https://github.com/sfujim/TD3_BC`. For DEMODICE, because the code is not public available, we implement it using PyTorch, adapting from `https://github.com/secury/optidice`; we use the hyperparameters reported in the paper. Note that DEMODICE shares many architectures with SMODICE. For example, DEMODICE uses a state-action discriminator, and we implement it by simply changing the input space of the state-based discriminator used in our SMODICE implementation. For SAIL, we use the official PyTorch implementation (`https://github.com/FangchenLiu/SAIL`) and combine it with TD3-BC. We implement RCE using PyTorch, adapting from the official TensorFlow implementation `https://github.com/google-research/google-research/tree/master/rce`.

Table 2: TD3+BC Hyperparameters. This table is reproduced from Fujimoto & Gu (2021) directly.

	Hyperparameter	Value
TD3 Hyperparameters	Optimizer	Adam (Kingma & Ba, 2014)
	Critic learning rate	3e-4
	Actor learning rate	3e-4
	Mini-batch size	256
	Discount factor	0.99
	Target update rate	5e-3
	Policy noise	0.2
	Policy noise clipping	(-0.5, 0.5)
	Policy update frequency	2
Architecture	Critic hidden dim	256
	Critic hidden layers	2
	Critic activation function	ReLU
	Actor hidden dim	256
	Actor hidden layers	2
	Actor activation function	ReLU
TD3+BC Hyperparameters	α	2.5

G Offline IL from Observations Experimental Details

G.1 Datasets

For Hopper, Walker2d, HalfCheetah, Ant, and AntMaze, we construct the offline datasets by combining a small amount of expert data and a large amount of low quality random data. For the first four tasks, we leverage the respective "expert-v2" and "random-v2" datasets in the D4RL benchmark. For AntMaze, we use trajectories from "antmaze-umaze-v2" as the expert data; for the random data, we simulate the antmaze environment for 1M steps using random actions and take the resulting transitions. For the kitchen environment, we use the full "kitchen-mixed-v0" dataset as the offline dataset without further augmentation. See Table 3 for dataset breakdown.

In AntMaze (Figure 4(b)), an Ant agent is tasked with navigating an U-shaped maze from one end to the other end (i.e., the goal region). The offline dataset (i.e., "antmaze-umaze-v2") consists of trajectories ($\approx$ 300k transitions) of an Ant agent navigating to the goal region from initial states; The trajectories are not always successful; often, the Ant flips over to its legs before it reaches the goal. As above, we additionally include 1 million random-action transitions to increase the task difficulty. We take one trajectory from the offline dataset that successfully reaches the goal to be the expert trajectory. Franka Kitchen (Figure 4(c)), introduced by Gupta et al. (2019), involves controlling a 9-DoF Franka robot to manipulate common household kitchen objects (e.g., microwave, kettle, cabinet) sequentially to achieve a pre-specified configuration of objects. The dataset (i.e., "kitchen-mixed-v0") consists of *undirected* human teleoperated demonstrations, meaning that each trajectory only solves a subset of the tasks. Together, these six tasks (illustrated in Figure 5.1) require scalability to high-dimensional state-action spaces and robustness to different dataset compositions.

Table 3: Offline Dataset Compositions.

Task	State Dim	Expert Dataset	Expert Data Size	Random Data Size
Hopper	11	hopper-expert-v2	193430	999999
Walker2d	17	walker2d-expert-v2	99900	999999
HalfCheetah	17	halfcheetah-expert-v2	199800	999000
Ant	27	ant-expert-v2	192409	999427
AntMaze	29	antmaze-umaze-v2	349687	999000
Kitchen	60	kitchen-mixed-v0	136937	0

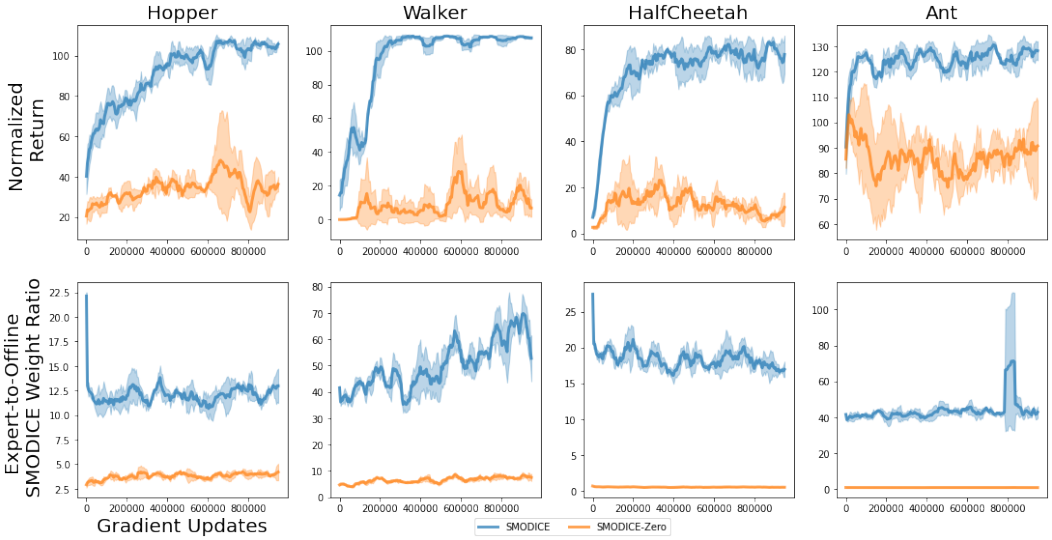


Figure 8: SMODICE vs. SMODICE-Zero. Using the discriminator-based reward, SMODICE assigns much higher

G.2 Additional Results

In this section, we present some additional results as well as ablation experiments.

Diverse AntMaze. In Section 5.1, we have found that two of the baselines (SAIL-TD3-BC and ORIL) outperform SMODICE on the AntMaze benchmark. To investigate their sources of empirical gain, we have designed a diverse version of the AntMaze dataset to test how different approaches are robust to the dataset composition on the same task. To this end, we take the AntMaze offline dataset (explained above) and reverse half of the trajectories in their directions. In other words, these reversed trajectories would navigate from the original goal to the initial state. This procedure is easy to do because the U-shaped maze is symmetric. Then, using this dataset, we have trained all approaches in Section 5.1 again. As shown in Figure 9(a), on this dataset, both SAIL-TD3-BC and ORIL quickly collapse, indicating that these methods are very brittle to the dataset composition. In contrast, SMODICE remains the best performing algorithm, despite overall drop in all methods' performances.

SMODICE with Zero Reward. We compare SMODICE with SMODICE-Zero, which simply assigns every transition zero reward (i.e., $R(s) = 0$) regardless of its similarity to an expert state. Then, we compare the ratio of the importance weights (i.e., $\xi(s, a)$) assigned to the offline expert data and the offline random data by the two SMODICE methods, respectively. As shown in Figure 8, SMODICE assigns much higher relative weights to the expert data and consequently significantly outperforms SMODICE-Zero. These results demonstrate that SMODICE's empirical performance comes from its superior ability to discriminate the offline expert data, which is a by-product of its optimization procedure.

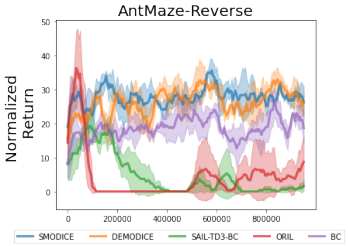
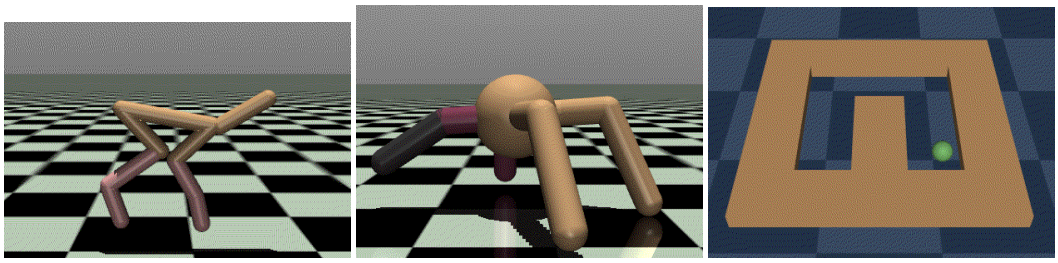


Figure 9: Offline imitation learning results on AntMaze-Reverse. SMODICE is still among the best performing methods, while both SAIL-TD3-BC and ORIL collapse, demonstrating their sensitivity to the offline dataset composition.



(a) HalfCheetah-Short (b) Ant-Disabled (c) PointMass-Maze

Figure 10: Illustrations of the mismatched experts.

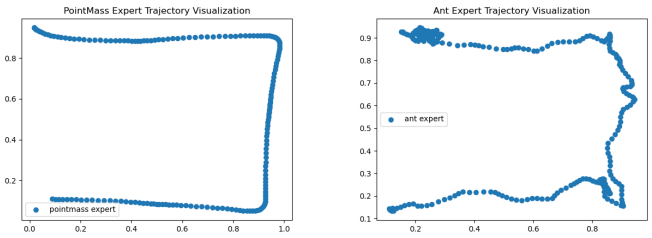
H Offline IL from Mismatched Expert Experimental Details

H.1 Continuous Control Experiments

Datasets and Baselines. We compare SMODICE to SAIL-TD3-BC and ORIL, which are both state-based offline IL methods; in particular, we note that SAIL is originally designed to be robust to mismatched experts. We consider only tasks in which both SAIL-TD3-BC and ORIL obtained non-trivial performance, including HalfCheetah, Ant, and AntMaze. Then, for each environment, we train a mismatched expert and collect one expert trajectory, replacing the original expert trajectory used in Section 5.1. The mismatched experts for the respective tasks are (i) "HalfCheetah-Short", where the torso of the cheetah agent is halved in length, (ii) "Ant-Disabled", where the front legs are shrank by a quarter in length, and (iii) a 2D PointMass agent operating in the same maze configuration. The mismatched experts are illustrated in Figure 10 in Appendix H. For the first two, we train an expert policy using SAC Haarnoja et al. (2018) and collect one expert trajectory. The latter task is already in D4RL; thus, we take one trajectory from "maze2d-umaze-v0" as the expert trajectory. Because Ant and PointMass have different state spaces, following Liu et al. (2019), we train the discriminator on the shared xy-coordinates of the two state spaces. The offline datasets are identical to the ones in Section 5.1.

Mismatched Experts. The mismatched experts are illustrated in Figure 10.

Comparison between PointMass and Ant experts for AntMaze. The trajectories of PointMass and Ant experts are illustrated in Figure 11. As can be seen, the PointMass trajectory is more regular and smooth due to its simpler dynamics and the use of a waypoint controller. In contrast, the ant trajectory is much less well-behaved because solving the maze task using the Ant agent is intrinsically a difficult task; consequently, it is difficult to provide an Ant demonstration. This example serves as a strong motivating problem for offline imitation learning with mismatched experts.



(a) PointMass Expert (b) Ant Expert

Figure 11: Trajectory visualizations of AntMaze experts.

Table 4: Relative performance drop with mismatched experts.

Algorithm	HalfCheetah	Ant	AntMaze	Average
SMODICE	**70.7%**	**3.3%**	**29.7%**	**34.5%**
SAIL-TD3-BC	88.9%	6.8%	50.6%	48.8%
ORIL	91.8%	42.2%	72.7%	68.9%

H.2 Quantitative Analysis of Figure 5

We quantitatively measure the percentage drop-in-performance for each method in Figure 5, computed as $\frac{|\text{max_original} - \text{max_mismatched}|}{\text{max_original}}$. Note that this metric favors the baselines as taking the maximum value advantages methods that are more unstable. Nevertheless, as shown in Table 4, SMODICE is still by far the most robust method overall and in each individual task. As expected, ORIL does the worst as it is not designed to handle mismatched dynamics; this shows that using a state-based discriminator in itself is not sufficient.

I Offline IL from Examples Experimental Details

I.1 Datasets

We collect 300 success-state examples for each of the tasks. The examples are randomly sampled from the subset of the offline dataset that achieves the task. Task success is verfied through a pre-defined sparse reward function (e.g., distance threshold function).

I.2 Environments.

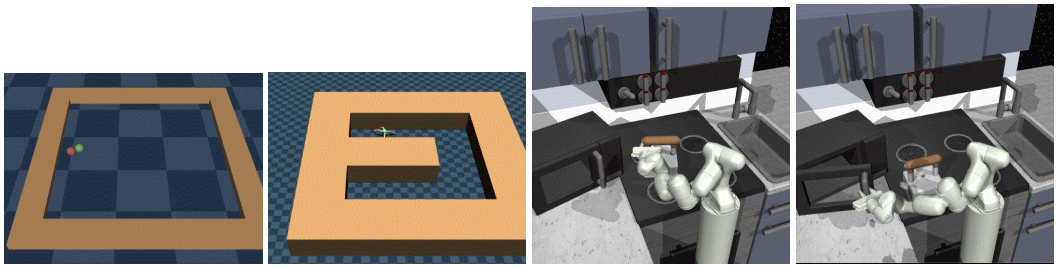
PointMass-4Direction. This environment is adapted from the "maze2d-umaze-v0" environment in D4RL by changing the map configuration. The environment termination condition is triggered when the agent successfully comes within a small radius of the specified goal.

AntMaze-Example. This environment is identical to the environments used in previous two settings.

Kettle and Microwave. These environments are adapted from the "kitchen-mixed-v0" environment in D4RL. The environments are identical as the original except the termination conditions. Both of these tasks terminate when the Franka robot places the specified object within a small radius of the desired configuration.

I.3 Examples of Success States

All success states are extracted from the offline dataset used for policy training. We illustrate one representative example from each task in Figure 13.



(a) PointMass-4Direction (b) AntMaze (c) Kettle (d) Microwave

Figure 13: Illustrations of success examples.

I.4 Additional Analysis

To better understand SMODICE, on PointMass-4Direction, we visualize the importance weights $\xi(s, a)$ it assigns to the offline dataset. As shown in Figure 12, SMODICE assigns much higher weights to transitions along the correct path from the initial state region to the success examples. Interestingly, the weights progressively decrease along this path, indicating that SMODICE has learned that it must pay more attention transitions at the beginning of the path, since making a mistake there is more likely to derail progress towards the goal. This behavior occurs *automatically* via SMODICE's state-matching objective without any additional bias.

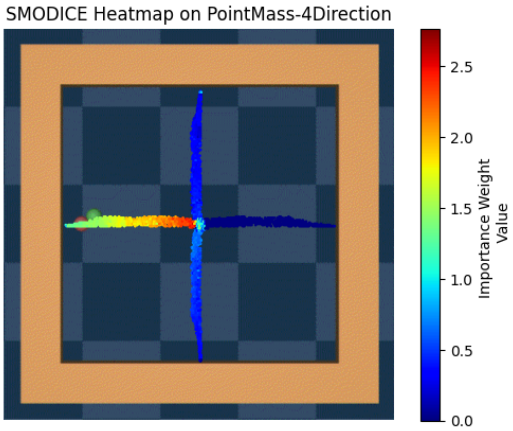


Figure 12: **SMODICE weights.**

Algorithm 1 SMODICE with χ^2-divergence for Tabular MDPs

```
# d_E: the expert state occupancies, |S|
# mdp: the empirical MDP learned using offline data
# pi_b: the behavior policy, |S||A|

def SMODICE(mdp, d_E, pi_b):
    d_O_sa = compute_policy_occupancies(mdp, pi_b)  # |S||A|
    d_O = d_O_sa.reshape(mdp.S, mdp.A).sum(axis=1) # |S|

    # compute reward function
    R = np.log(d_E/d_O) # |S|

    # define and reshape matrices
    T = mdp.T.reshape(mdp.S * mdp.A, mdp.S)  # |S||A| x |S|
    B = np.repeat(np.eye(mdp.S), mdp.A, axis=0)  # |S||A| x |S|
    I = np.ones(mdp.S * mdp.A) # |S||A|
    D = np.diag(d_O_sa)  # |S||A| x |S||A|

    # compute optimal V
    H = (mdp.gamma * P - B).T @ D @ (mdp.gamma * T - B) # |S| x |S|
    y = -((1 - mdp.gamma) * p0 + (mdp.gamma * P - B).T @ D @ (I + B @ R)) # |S|
    V_star = np.linalg.pinv(H) @ y # |S|

    # compute optimal occupancy ratios
    xi_star = B @ R + (mdp.gamma * P - B) @ V_star + 1 # |S||A|
    m = np.array(xi_star >= 0, dtype=np.float)
    xi_star = xi_star * m

    # weighted BC
    pi_star = (xi_star * d_O).reshape(mdp.S, mdp.A) # |S||A|
    pi_star /= np.sum(pi_star, axis=1, keepdims=True)

    f_divergence = d.dot(0.5 * (w_star ** 2))

    return pi_star, f_divergence, V_star
```

Algorithm 2 SMODICE for Continuous MDPs

Require: Expert demonstration(s) $\mathcal{D}^E$, offline dataset $\mathcal{D}^O$, choice of f-divergence f
Randomly initialize discriminator c_ψ, value function V_θ, and policy π_ϕ.
// Train Expert (resp. Example) Discriminator
Train c_ψ using $\mathcal{D}^E$ and $\mathcal{D}^O$ using Equation (15)
// Train Lagrangian Value Function
for number of iterations **do**
  Sample minibatch of offline data $\{s_t^i, a_t^i, s_{t+1}^i\}_{i=1}^N \sim \mathcal{D}^O, \{s_0^i\}_{i=1}^M \sim \mathcal{D}^O(\mu_0)$
  Obtain reward: $R_i = c_\theta(s_t^i), i = 1, ..., N$
  Compute value objective $\mathcal{L}(\theta) := (1-\gamma)\frac{1}{M}\sum_{i=1}^M V_\theta(s_0^i) + \frac{1}{N} f_\star\left(R^i + \gamma V(s_{t+1}^i) - V(s_t^i)\right)$
  Update V_θ using SGD: $V_\theta \leftarrow V_\theta - \eta_V \nabla \mathcal{L}(\theta)$
end for
// Policy Learning
for number of iterations **do**
  Sample minibatch of offline data $\{s_t^i, a_t^i, s_{t+1}^i\}_{i=1}^N \sim \mathcal{D}^O$
  // Compute Optimal Importance Weights
  Compute $\xi^*(s^i, a^i) = f_\star'\left(R(s^i) + \gamma V(s_{t+1}^i) - V(s_t^i)\right), i = 1, ..., N$
  // Weighted Behavior Cloning
  Update π_ψ using Equation (18)
end for