Offline Imitation from Observation via Primal Wasserstein State Occupancy Matching

Kai Yan [1] **Alexander G. Schwing** [1] **Yu-Xiong Wang** [1]

Abstract

In real-world scenarios, arbitrary interactions with the environment can often be costly, and actions of expert demonstrations are not always available. To reduce the need for both, offline Learning from Observations (LfO) is extensively studied: the agent learns to solve a task given only expert states and *task-agnostic* non-expert state-action pairs. The state-of-the-art DIstribution Correction Estimation (DICE) methods, as exemplified by SMODICE, minimize the state occupancy divergence between the learner's and empirical expert policies. However, such methods are limited to either f-divergences (KL and χ^2) or Wasserstein distance with Rubinstein duality, the latter of which constrains the underlying distance metric crucial to the performance of Wasserstein-based solutions. To enable more flexible distance metrics, we propose Primal Wasserstein DICE (PW-DICE). It minimizes the primal Wasserstein distance between the learner and expert state occupancies and leverages a contrastively learned distance metric. Theoretically, our framework is a *generalization* of SMODICE, and is *the first work* that *unifies* f-divergence and Wasserstein minimization. Empirically, we find that PW-DICE improves upon several state-of-the-art methods. The code is available at `https://github.com/KaiYan289/PW-DICE`.

1. Introduction

Recent years have witnessed remarkable advances in *offline* Reinforcement Learning (RL) (Chen et al., 2021b; Kostrikov et al., 2022a;b): sequential decision-making problems are addressed with independently collected interaction data rather than an online interaction which is often costly to conduct (e.g., autonomous driving (Kiran et al., 2021)). Even without online interaction, methods achieve high sample efficiency. Such methods, however, require reward labels that are often missing when data are collected in the wild (Chang et al., 2020). In addition, an informative reward is expensive to obtain for many tasks, such as robotic manipulation, as it requires a carefully hand-crafted design (Yu et al., 2019). To bypass the need for reward labels, offline Imitation Learning (IL) has prevailed (Hakhamaneshi et al., 2022; Ho & Ermon, 2016; Kim et al., 2022a). It enables the agent to learn from existing demonstrations without reward labels. However, just like reward labels, expert demonstrations are also expensive and often scarce, as they need to be recollected repeatedly for every task of interest. Among different types of expert data shortages, there is one widely studied type: *offline Learning from Observations (LfO)*. In LfO, only the expert state, instead of both state and action, is recorded. This setting is useful when learning from experts with different embodiment (Ma et al., 2022) or from video demonstrations (Chen et al., 2021a), where the expert action is either not applicable or not available.

Many methods have thus been proposed in offline LfO, including inverse RL (Zolna et al., 2020; Torabi et al., 2019; Kostrikov et al., 2019), similarity-based reward labeling (Sermanet et al., 2017; Chen et al., 2021a), and action pseudo-labeling (Torabi et al., 2018; Kumar et al., 2019). The state-of-the-art solution to LfO is the family of DIstribution Correction Estimation (DICE) methods, which are LobsDICE (Kim et al., 2022b) and SMODICE (Ma et al., 2022): both methods perform a convex optimization in the dual space to minimize the f-divergence of the state occupancy (visitation frequency) between the learner and the empirical expert policies approximated from the dataset. Notably, DICE methods mostly focus on f-divergences (Kim et al., 2022b; Ma et al., 2022; Kostrikov et al., 2020; Kim et al., 2022a) (mainly KL-divergence and χ^2-divergence; see Appendix B for definition), metrics that ignore some underlying geometric properties of the distributions (Stanczuk et al., 2021). While there is a DICE variant, SoftDICE (Sun et al., 2021), that introduces the Wasserstein distance to DICE methods, it adopts the Kantorovich-Rubinstein duality (Kantorovich & Rubinstein, 1958; Peyré & Cuturi,

[1]The Grainger College of Engineering, University of Illinois Urbana-Champaign, Urbana, Illinois, USA. Correspondence to: Kai Yan <kaiyan3@illinois.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

2019), which limits the choice of the underlying distance metric: duality requires the underlying metric to be Euclidean (Stanczuk et al., 2021). This limitation of the distance metric is not only theoretically infavorable, but also impacts practical performance. Concretely, we find the distance metric in Wasserstein-based methods to be crucial for performance (Sec. 3.1).

To enable more flexible distance metrics, we propose Primal Wasserstein DICE (PW-DICE), a DICE method that optimizes the primal form of the Wasserstein distance. PW-DICE is illustrated in Fig. 1. With an adequate regularizer for offline pessimism (Jin et al., 2021), the joint minimization of the Wasserstein matching variable and the learner policy can be formulated as a convex optimization over the Lagrange space. The policy is then retrieved by weighted behavior cloning with weights determined by the Lagrange function. Different from SMODICE and LobsDICE, our underlying distance metric can be chosen arbitrarily, and different from all prior work, we explore the possibility of contrastively learning the metric from data. Compared to existing Wasserstein-based work that either uses Rubinstein dual or chooses simple, fixed metrics (e.g., Euclidean for PWIL (Dadashi et al., 2021) and cosine for OTR (Luo et al., 2023)), our effort endows PW-DICE with much more flexibility. Meanwhile, with specifically chosen hyperparameters, SMODICE can be obtained as a special case of PW-DICE, which theoretically guarantees our performance.

We summarize our contributions as follows: 1) to our best knowledge, this is *the first work* that sheds light on the practical importance of the underlying distance metric in LfO; 2) we propose a novel offline LfO method, PW-DICE, which uses the primal Wasserstein distance for LfO, gaining more flexibility regarding the distance metric than prior work, while removing the assumption for data coverage; 3) we theoretically prove that PW-DICE is a generalization of SMODICE, thus providing *the first unified framework* for Wasserstein-based and f-divergence-based DICE methods; 4) we empirically show that our method achieves better results than the state of the art on multiple offline LfO testbeds.

2. Preliminaries

Markov Decision Process. The Markov Decision Process (MDP) is a widely adopted formulation for sequential decision-making problems. An MDP has five components: a state space S, an action space A, a transition function T, a reward r, and a discount factor γ. An MDP evolves in discrete steps: at step $t \in \{0, 1, 2, \dots\}$, the state $s_t \in S$ is given, and an agent, following its policy $\pi(a_t|s_t) \in \Delta(A)$ (where $\Delta(A)$ is the probability simplex over A), chooses an action $a_t \in A$. After receiving a_t, the MDP transits to a new state $s_{t+1} \in S$ according to the transition probability function $T(s_{t+1}|s_t, a_t)$, and yields a reward $r(s_t, a_t) \in \mathbb{R}$ as feedback. The agent needs to maximize the discounted total reward $\sum_t \gamma^t r(s_t, a_t)$ with discount factor $\gamma \in [0, 1]$. A complete run of the MDP is defined as an episode, with the state(-action) pairs collected along the trajectory τ. The state occupancy, which is the visitation frequency of states given policy π, is $d_s^\pi(s) = (1-\gamma)\sum_t \gamma^t \Pr(s_t = s)$. See Appendix B for more rigorous definitions of the state occupancy and other occupancies.

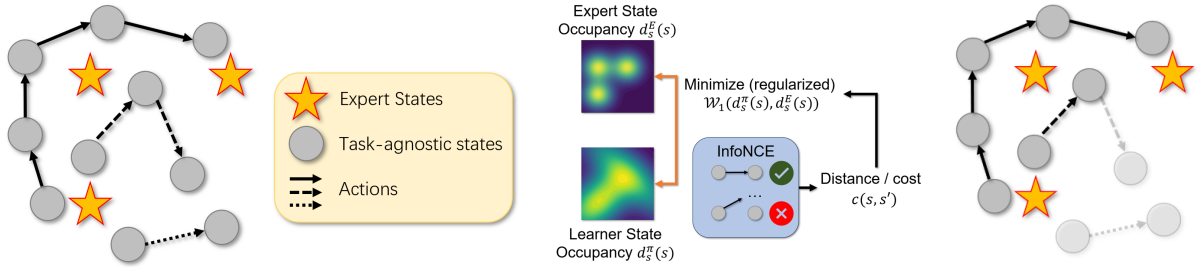
Offline Imitation Learning from Observations (LfO). In offline LfO, the agent needs to learn from two sources of data: 1) the *expert* dataset E with state-only trajectories $\tau_E = \{s_1, s_2, \dots, s_{n_1}\}$ that solve the exact target task, and 2) the *task-agnostic* non-expert dataset I consisting of less relevant state-action trajectories $\tau_I = \{(s_1, a_1), (s_2, a_2), \dots, (s_{n_2}, a_{n_2})\}$. Ideally, the agent learns the environment dynamics from I, and tries to follow the expert states in E with information about the MDP inferred from I. The state-of-the-art methods in offline LfO are SMODICE (Ma et al., 2022) and LobsDICE (Kim et al., 2022b). The two methods are in spirit similar, with the former minimizing state occupancy divergence and the latter optimizing adjacent *state-pair* occupancy divergence.

Wasserstein Distance. The Wasserstein distance, also known as Earth Mover's Distance (EMD) (Kantorovich, 1960), is widely used as the distance between two probability distributions. It captures the geometry of the underlying space better and does not require any intersection between the support sets. For two distributions $p \in \Delta(S), q \in \Delta(S)$ over state space S, the Wasserstein distance[1] with an underlying metric $c(x, y) : S \times S \to \mathbb{R}$ can be written as $\mathcal{W}(p, q) = \inf_{\Pi \in S \times S} \int_{x \in S} \int_{y \in S} \Pi(x, y) c(x, y)$, which is the *primal form* of the Wasserstein distance; Π is the matching variable between p and q. Wasserstein also has an equivalent Kantorovich-Rubinstein dual form (Kantorovich & Rubinstein, 1958), which is $\mathcal{W}(p, q) = \max_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p} f(x) - \mathbb{E}_{y \sim q} f(y)$, where $\|f\|_L \leq 1$ means that the function f is 1-Lipschitz. While this form is often adopted by the machine learning community, the Lipschitz constraint is usually implemented by a gradient regularizer in practice. As the gradient is defined using a Euclidean distance, the underlying distance metric for Rubinstein duality is also restricted to Euclidean (Stanczuk et al., 2021), which is often suboptimal.

3. Method

This section is organized as follows: in Sec. 3.1, we first validate our motivation, i.e., the importance of selecting an adequate distance metric by comparing metrics using

[1] Unless otherwise specified, we only consider 1-Wasserstein distance in this paper.



(a) Problem Setting (b) Wasserstein Optimization (c) Weighted BC

Figure 1. An illustration of our method, PW-DICE. **a)** Problem setting: different trajectories are illustrated by different styles of arrows. **b)** PW-DICE minimizes regularized 1-Wasserstein distance between the learner's state occupancy $d_s^\pi(s)$ and the expert state occupancy $d_s^E(s)$. The underlying distance function is contrastively learned to represent the reachability between the states. **c)** With the matching result, weights are calculated for downstream weighted Behavior Cloning (BC) to retrieve the policy. High transparency indicates a small weight for the state and its corresponding action.

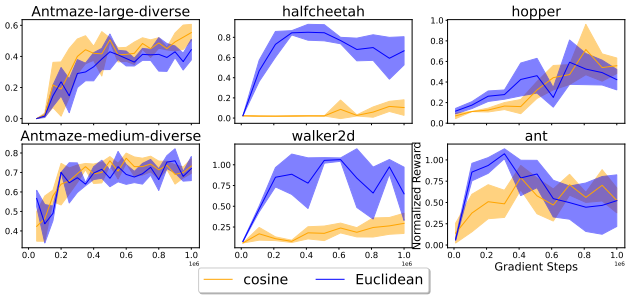


Figure 2. Performance comparison between the default (normalized cosine) distance metric and Euclidean distance metric using OTR (Luo et al., 2023) (first column), and SMODICE (Ma et al., 2022) (second and third columns). The result shows that the underlying distance metric is crucial for the performance of Wasserstein-based methods.

existing Wasserstein-based solutions; then, we detail our proposed optimization objective in Sec. 3.2; finally, we discuss our choice of the distance metric in Sec. 3.3. See Tab. 2 in Appendix F for a reference of the notation, and Appendix C for detailed derivations.

3.1. Validation of Motivation

As mentioned in Sec. 1, our goal is to improve the idea of divergence minimization between the learner's policy and the expert policy estimated from the expert dataset. For this we suggest to use the primal Wasserstein distance which allows flexibly using an arbitrary underlying distance metric. To show the importance of distance metrics and the advantage of being able to select them, we study Optimal Transport Reward (OTR) (Luo et al., 2023), a current Wasserstein-based IL method that can be applied to our LfO setting. OTR optimizes the primal Wasserstein distance between every trajectory in the task-agnostic dataset and the expert trajectory, and uses the result to assign a reward to each state in the task-agnostic dataset. Then, offline RL is applied to retrieve the optimal policy. Fig. 2 shows results of OTR on the D4RL MuJoCo dataset (see Sec. 4.2 for more) with testbeds appearing in both SMODICE (Ma et al., 2022) and OTR. We test both the cosine-similarity-based occupancy used in Luo et al. (2023) and the Euclidean distance as the underlying distance metric. The results illustrate that distance metrics have a significant impact on outcomes. Thus, selecting a good metric is crucial for the performance of Wasserstein-based solutions.

This experiment motivates our desire to develop a method that permits the use of arbitrary distance metrics in Wasserstein-based formulations. Further, the observed performance difference inspires us to automate the selection of the metric, going beyond classic metrics such as cosine and Euclidean. We discuss the formulation of our suggested method next.

3.2. Optimization Objective

Our goal is to optimize the primal Wasserstein distance between the model-estimated state occupancy $d_s^\pi(s)$ induced by the policy π and the empirical state occupancy $d_s^E(s)$ estimated from expert data. This can be formalized via the following program:

$$\min_{\Pi,\pi} \sum_{s_i \in S} \sum_{s_j \in S} \Pi(s_i, s_j) c(s_i, s_j), \text{s.t. } d_{sa}^\pi \geq 0, \Pi \geq 0; \quad (1)$$

$$\forall s \in S, d_s^\pi(s) = (1-\gamma) p_0(s) + \gamma \sum_{\bar{s},\bar{a}} d_{sa}^\pi(\bar{s},\bar{a}) p(s|\bar{s},\bar{a});$$

$$\forall s_i, s_j \in S, \textstyle\sum_k \Pi(s_k, s_j) = d_s^E(s_j), \sum_k \Pi(s_i, s_k) = d_s^\pi(s_i).$$

In Eq. (1), we use $\Pi(s_i, s_j)$ as the matching variable between the two state occupancy distributions $d_s^\pi(s)$ and $d_s^E(s)$, and $c(s_i, s_j)$ is the distance between states s_i and

s_j. Further, d^π_{sa} is the state-action occupancy of our learned policy π, and $p_0 \in \Delta(S)$ is the distribution of the MDP's initial states. Note that there are two types of constraints in Eq. (1): the first row, together with $d^\pi_{sa} \geq 0$, is the marginal constraint for the matching variable Π. The second row and $\Pi \geq 0$ are the *Bellman flow constraints* (Ma et al., 2022) that ensure correspondence between d^π_s and a feasible policy π.

For a tabular MDP, Eq. (1) can be solved by any Linear Programming (LP) solver, as both the objective and the constraints are linear. However, using an LP solver is impractical for any MDP with continuous state or action spaces. In these cases, we need to convert the problem into a program that is easy to optimize. The common way to remove constraints is to consider the Lagrangian dual problem. However, the Lagrangian dual problem of an LP with constraints is also an LP with constraints. In order to reduce constraints in the dual program, we smooth the objective by using:

$$\Pi(s_i, s_j)c(s_i, s_j) + \epsilon_1 D_f(\Pi \| U) + \epsilon_2 D_f(d^\pi_{sa} \| d^I_{sa}). \tag{2}$$

In Eq. (2), d^I_s and d^I_{sa} are the empirical state occupancy and state-action occupancy of the task-agnostic dataset I respectively. Further, we let $U(s, s') = d^E_s(s) d^I_s(s')$, i.e., U is the product of two independent distributions d^E_s and d^I_s. Moreover, $\epsilon_1 > 0, \epsilon_2 > 0$ are hyperparameters, and D_f can be any f-divergence (we will focus on the KL-divergence in this paper). Note, despite the use of an f-divergence, different from SMODICE (Ma et al., 2022) or LobsDICE (Kim et al., 2022b), this formulation does not require data coverage of the task-agnostic data over expert data. The two regularizers are "pessimistic": they encourage the agent to stay within the support set of the dataset which is common in offline IL/RL (Jin et al., 2021).

Given the smooth objective, we apply Fenchel duality to derive a robust single-level optimization in the dual space. See Appendix C.2 for a detailed derivation.

The dual program when letting D_f refer to a KL-divergence (see Appendix E.2 for a discussion on χ^2-divergence) reads as follows:

$$\begin{aligned}
&\min_\lambda \ \epsilon_1 \log \mathbb{E}_{s_i \sim I, s_j \sim E} \exp\left(\frac{\lambda_{i+|S|} + \lambda_{j+2|S|} - c(s_i, s_j)}{\epsilon_1}\right) \\
&+\epsilon_2 \log \mathbb{E}_{(s_i, a_j) \sim I} \exp\left(\frac{-\gamma \mathbb{E}_{s_k \sim p(\cdot|s_i, a_j)} \lambda_k + \lambda_i - \lambda_{i+|S|}}{\epsilon_2}\right) \\
&- \left[(1-\gamma)\mathbb{E}_{s \sim p_0} \lambda_{:|S|} + \mathbb{E}_{s \sim E} \lambda_{2|S|:3|S|}\right].
\end{aligned} \tag{3}$$

Intuitively, the dual variables $\lambda \in \mathbb{R}^{3|S|}$ in the objective are divided into three parts, each of size $|S|$: $\lambda_i, i \in \{1, 2, \ldots, |S|\}$ can be seen as a variance of the *value function*, where $-\gamma \mathbb{E}_{s_k \sim p(\cdot|s_i, a_j)} \lambda_k + \lambda_i - \lambda_{i+|S|}$ is its Bellman residual, or negative TD(0) advantage. $\lambda_{i+|S|}$ and $\lambda_{i+2|S|}$ are costs attached to a particular state: if we compare Wasserstein matching to shipping probability mass, $\lambda_{i+|S|}$ would be the loading cost from states of π and $\lambda_{i+2|S|}$ would be the unloading cost to the states of E. Note, by Theorem C.1 (see Appendix C.2 for a detailed derivation), we have $d^\pi_{sa} = d^I_{sa} \cdot \text{softmax}\left(\frac{-\gamma \mathbb{E}_{s_k \sim p(\cdot|s_i, a_j)} \lambda_k + \lambda_i - \lambda_{i+|S|}}{\epsilon_2}\right)$ at the optimum, and the denominator of the softmax is summing over all state-action pairs.

With λ optimized, we retrieve the desired policy π by weighted behavior cloning, maximizing the following objective:

$$\begin{aligned}
&\mathbb{E}_{(s_i, a_j) \sim d^\pi_{sa}} \log \pi(a|s) = \mathbb{E}_{(s_i, a_j) \sim I} \frac{d^\pi_{sa}(s_i, a_j)}{d^I_{sa}(s_i, a_j)} \log \pi(a_j|s_i) \\
&\propto \mathbb{E}_{(s_i, a_j) \sim I} \exp\left(\frac{-\gamma \mathbb{E}_{s_k} \lambda_k + \lambda_i - \lambda_{i+|S|}}{\epsilon_2}\right) \log \pi(a_j|s_i).
\end{aligned} \tag{4}$$

In practice, we use 1-sample estimation for $p(\cdot|s_i, a_j)$, a method found to be simple and effective in prior work (Ma et al., 2022; Kim et al., 2022b). That is, we sample $(s_i, a_j, s_k) \sim I$ from the dataset instead of (s_i, a_j), and use λ_k corresponding to s_k as an estimation for $\mathbb{E}_{s_k \sim p(\cdot|s_i, a_j)} \lambda_k$. Since the number of states can be infinite in practice, we use a 3-head neural network to estimate $\lambda_s, \lambda_{s+|S|}$ and $\lambda_{s+2|S|}$ given state s. See Appendix A for pseudo-code of our algorithm where we iteratively optimize the dual by Eq. (3) and obtain the policy π by Eq. (4).

Importantly, note that our formulation can be seen as a generalization of SMODICE (Ma et al., 2022). It is not hard to see why and we point this out next. SMODICE's objective with KL divergence reads as follows:

$$\begin{aligned}
&\min_V \log E_{(s,a) \sim I}\left[\exp(R(s) + \gamma \mathbb{E}_{s' \sim (s,a)} - V(s))\right] + \\
&\quad (1-\gamma)\mathbb{E}_{s \sim p_0}\left[V(s)\right],
\end{aligned} \tag{5}$$

where $V(s)$ is a value function and $R(s)$ is the reward assigned for states. It is easy to see that λ_i corresponds to $V(s)$ in SMODICE, and SMODICE is a special case of PW-DICE with $\epsilon_1 \to 0, \epsilon_2 = 1$, and $c(s, s') = -R(s)$. We highlight that in the SMODICE setting, the distance $c(s, s')$ only depends on the first state s. Thus, the total matching cost is fixed for any matching plan given particular state occupancy of π, i.e., d^π_s. Meanwhile, the pessimistic regularizer with large coefficient ϵ_2 dominates. This generalization property also holds for other divergences such as χ^2. See Appendix C for a more rigorous derivation.

3.3. Underlying Distance Metric

Given Eq. (3) and Eq. (4), it remains to choose the distance metric $c(s_i, s_j)$. For tabular cases, one could use the simplest distance, i.e., $c(s_i, s_j) = 1$ if $s_i \neq s_j$, and 0 otherwise. However, such a distance only provides "sparse" information in the continuous case. The distance will mostly

be 0, and will degrade to all zeros if there is no common state in the expert dataset E and the task-agnostic dataset I. To address this, prior work has explored many heuristic choices, such as cosine similarity (Luo et al., 2023) or a Euclidean (Sun et al., 2021) distance. However, such choices are often suboptimal for particular environments, as shown when validating our motivation in Sec. 3.1.

In this work, inspired by both CURL (Laskin et al., 2020) and SMODICE (Ma et al., 2022), we propose a weighted sum of $R(s) = \log \frac{d_s^E(s)}{(1-\alpha)d_s^I(s)+\alpha d_s^E(s)}$ and the Euclidean distance between an embedding learned by the InfoNCE (Wan et al., 2021) loss. To be more specific, we let the distance metric c be

$$c(s_i, s_j) = R(s_i) + \beta \|g(s_i) - g(s_j)\|_2^2, \tag{6}$$

where $g(s_i), g(s_j)$ are learned embeddings for the states s_i, s_j respectively, α is a positive constant close to 0, and $\beta \geq 0$ is a hyperparameter.

The distance function consists of two parts. The first part, $R(s_i)$, is a modified version of the SMODICE reward function $\log \frac{d_s^E(s)}{d_s^I(s)}$. Intuitively, high $\log \frac{d_s^E(s)}{d_s^I(s)}$ indicates that the state s is more frequently visited by the expert than agents generating the task-agnostic data, which is probably desirable. Such reward can be obtained by training a discriminator $h(s)$ that takes expert states from E as label 1 and non-expert ones as label 0. If h is optimal, i.e., $h(s) = h^*(s) = \frac{d_s^E(s)}{d_s^E(s)+d_s^I(s)}$, then we have $\frac{d_s^E(s)}{d_s^I(s)} = \log \frac{h^*(s)}{1-h^*(s)}$. Based on this, we change the denominator $d_s^E(s)$ to $(1-\alpha)d_s^I(s) + \alpha d_s^E(s)$ to lift the theoretical assumption that the task-agnostic dataset I covers the expert dataset E, i.e., $d_s^I(s) > 0$ wherever $d_s^E(s) > 0$.

The second part uses the embedding $g(s)$ learned with InfoNCE (Wan et al., 2021), which is also adopted in CURL (Laskin et al., 2020) and FIST (Hakhamaneshi et al., 2022). Different from CURL, where the contrastive learning is an auxiliary loss in addition to RL for better extraction of features, and FIST, which tries to find the similarity between the current state and a state in the dataset, we want $g(s)$ and $g(s')$ to be similar if and only if they are reachable along trajectories in the task-agnostic dataset. For this, we sample a batch of consecutive state pairs $(s_i, s_i'), i \in \{1, 2, \dots\}$, and use the following loss function:

$$\log \frac{\exp\left(g(s_i)^T W g(s_i')\right)}{\exp\left(g(s_i)^T W g(s_i')\right) + \sum_{j \neq i} \exp\left(g(s_i)^T W g(s_j')\right)}. \tag{7}$$

Here, $g(s_i)$ can be seen as an *anchor* in contrastive learning, W is a learned matrix, $g(s_i')$ is its *positive key*, and $g(s_j'), j \neq i$ is its *negative key*. Intuitively, the idea is to learn a good embedding space where the vicinity of a state can be assessed by the Euclidean distance between the embedding vectors. We define the vicinity as the "reachability" between states: if one state can reach the other through a trajectory in the task-agnostic data, then states should be close, otherwise they are far from each other. This definition groups states that lead to success in the embedding space (see Fig. 6 for a visualization), while being robust to actual numerical values of the state (see Sec. 4.2 for empirical evaluations).

4. Experiments

We evaluate PW-DICE across multiple environments. We strive to answer two main questions: 1) can the Wasserstein objective indeed lead to a closer match between the learner's and the expert policies (Sec. 4.1)?; and 2) can PW-DICE improve upon f-divergence based methods on more complicated environments, and does a flexible underlying distance metric indeed help (Sec. 4.2)?

4.1. Primal Wasserstein vs. f-Divergence

Baselines. We compare to the two major state-of-the-art baselines, SMODICE (Ma et al., 2022) and LobsDICE (Kim et al., 2022b). We test two variants of our method: 1) Linear Programming (LP) by directly solving Eq. (1); and 2) Regularized (Reg) which solves Eq. (2). As the environment is tabular, all methods are implemented with CVXPY (Agrawal et al., 2019)[2] for optimal numerical solutions. The mean and standard deviation are obtained from 10 independent runs with different seeds. We evaluate all methods with the **regret**, i.e., the gap between rewards gained by the learner's and expert policies (*lower is better*). To be consistent with LobsDICE, in Appendix E.1, we also compare the Total Variation (TV) distance for the state and state-pair occupancies, i.e., $\text{TV}(d_s^\pi \| d_s^E)$ and $\text{TV}(d_{ss}^\pi \| d_{ss}^E)$.

Environment Setup. Following the random MDP experiment in LobsDICE (Kim et al., 2022b), we randomly generate an MDP with $|S| = 20$ states, $|A| = 4$ actions, and $\gamma = 0.95$. The stochasticity of the MDP is controlled by $\eta \in [0, 1]$, where $\eta = 0$ is deterministic and $\eta = 1$ is highly stochastic. Agents always start from one particular state, and aim to reach another particular state with reward $+1$, which is the only source of reward. We report the regret for different η, expert dataset sizes, and task-agnostic dataset sizes. The only difference from LobsDICE is: the expert policy is deterministic instead of being softmax, as we found the high connectivity of the MDP states to lead to a near-uniform value function. Thus, the softmax expert policy is highly suboptimal and near-uniform. See Appendix D for explanation and Appendix E.1 for results.

Experimental Setup. As the environment is tabular, as

[2] In our experiments, CVXPY usually invokes Gurobi (Gurobi Optimization, LLC, 2023) for linear programming and MOSEK (ApS, 2019) for other objectives during optimization.

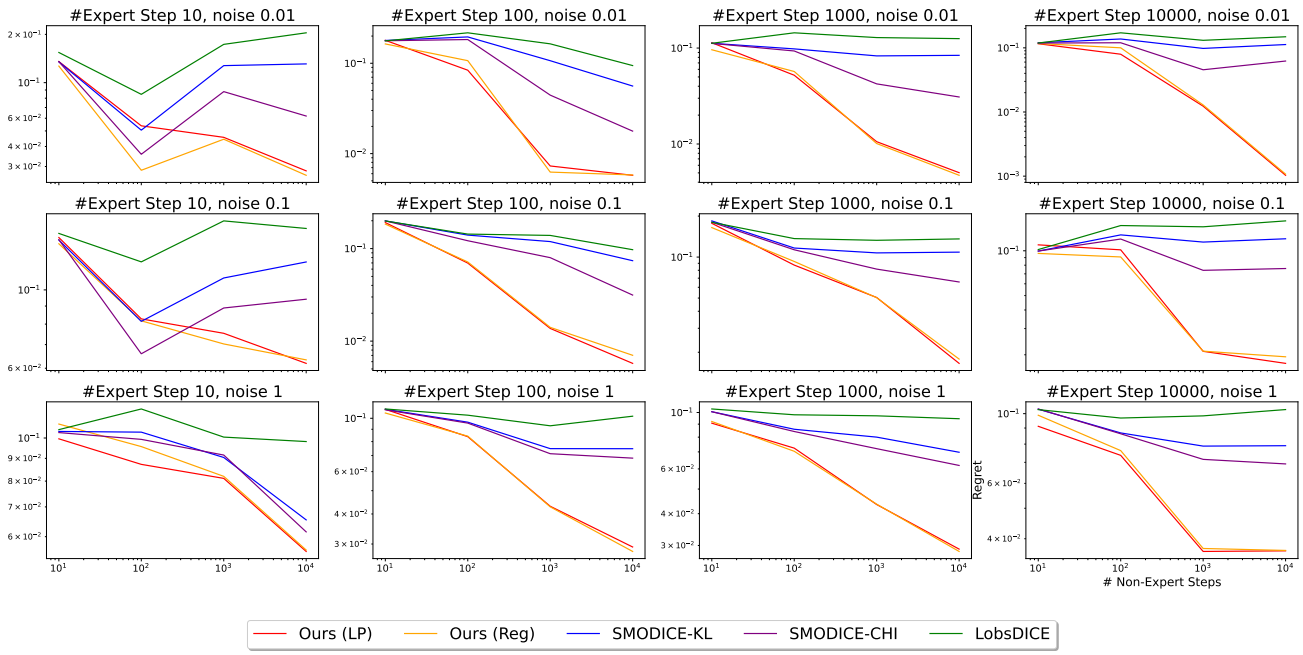


Figure 3. The regret (reward gap between learner and expert) of each method on a tabular environment. We observe our method to work the best, regardless of the presence of a regularizer; the regularizer is more important in continuous MDPs.

mentioned above, we use CVXPY (Agrawal et al., 2019) to solve for the optimal policy for each method using the *primal* formulation. For example, we directly solve Eq. (1) to get the learner's policy π. Following SMODICE, for estimating transition function and the task-agnostic average policy π^I, we simply count the state-action pair and transitions from the task-agnostic dataset I, i.e., the transition probability $p(s'|s,a) = \frac{\#[(s,a,s')\in I]}{\#[(s,a)\in I]}$, and $\pi^I(a|s) = \frac{\#[(s,a)\in I]}{\#[s\in I]}$ ($\#$ stands for "the number of"). Similarly, the expert state occupancy d_s^E is estimated by $d_s^E(s) = \frac{\#[s\in E]}{|E|}$, where $|E|$ is the size of the expert dataset E. Notably, if the denominator is 0, the distribution will be estimated as uniform. As the environment is tabular, we use the simplest distance metric, described in the beginning of Sec. 3.3, i.e., $c(s_i, s_j) = 1$ if $s_i \neq s_j$ and 0 otherwise.

Main Results. Fig. 3 shows the regret of each method. We observe our method with or without regularizer to perform similarly and to achieve the lowest regret across expert dataset sizes in $\{10, 100, 1000, 10000\}$, task-agnostic (non-expert) dataset sizes in $\{10, 100, 1000, 10000\}$, and noise levels $\eta \in \{0.01, 0.1, 1\}$. The gap increases with the task-agnostic dataset size, which shows that our method works better when the MDP dynamics are more accurately estimated. LobsDICE struggles in this scenario, albeit being the best in minimizing the divergence to the softmax expert, which is more stochastic and suboptimal (see Appendix E.1 for details).

4.2. More Complex Environments

Baselines. We adopt seven baselines in our study: state-of-the-art DICE methods SMODICE (Ma et al., 2022), LobsDICE (Kim et al., 2022b), and ReCOIL (Sikchi et al., 2024), non-DICE method ORIL (Zolna et al., 2020), Wasserstein-based method OTR (Luo et al., 2023), DWBC (Xu et al., 2022) with extra access to the expert action, and the plain Behavior Cloning (BC). As we have no access to the ReCOIL code, we directly report the final numbers from their paper. Mean and standard deviation are obtained from 3 independent runs with different seeds. We measure the performance using the average reward (*higher is better*).

Environment and Experimental Setup. Following SMODICE (Ma et al., 2022), we test PW-DICE on four standard OpenAI gym MuJoCo environments: hopper, halfcheetah, ant, and walker2d, as well as two more challenging MuJoCo testbeds, antmaze and Franka kitchen. The datasets that we use are identical to those in SMODICE (see Appendix D for details). The metric we use is the normalized average reward, where higher reward indicates better performance[3]. If the final reward is similar, the algorithm with fewer gradient step updates is better. We plot the reward curve, which illustrates the change of the mean and standard deviation of the reward with the number of gradient steps. See Appendix D for hyperparameters.

[3]We use the same normalization standard as D4RL (Fu et al., 2020b) and SMODICE (Ma et al., 2022).

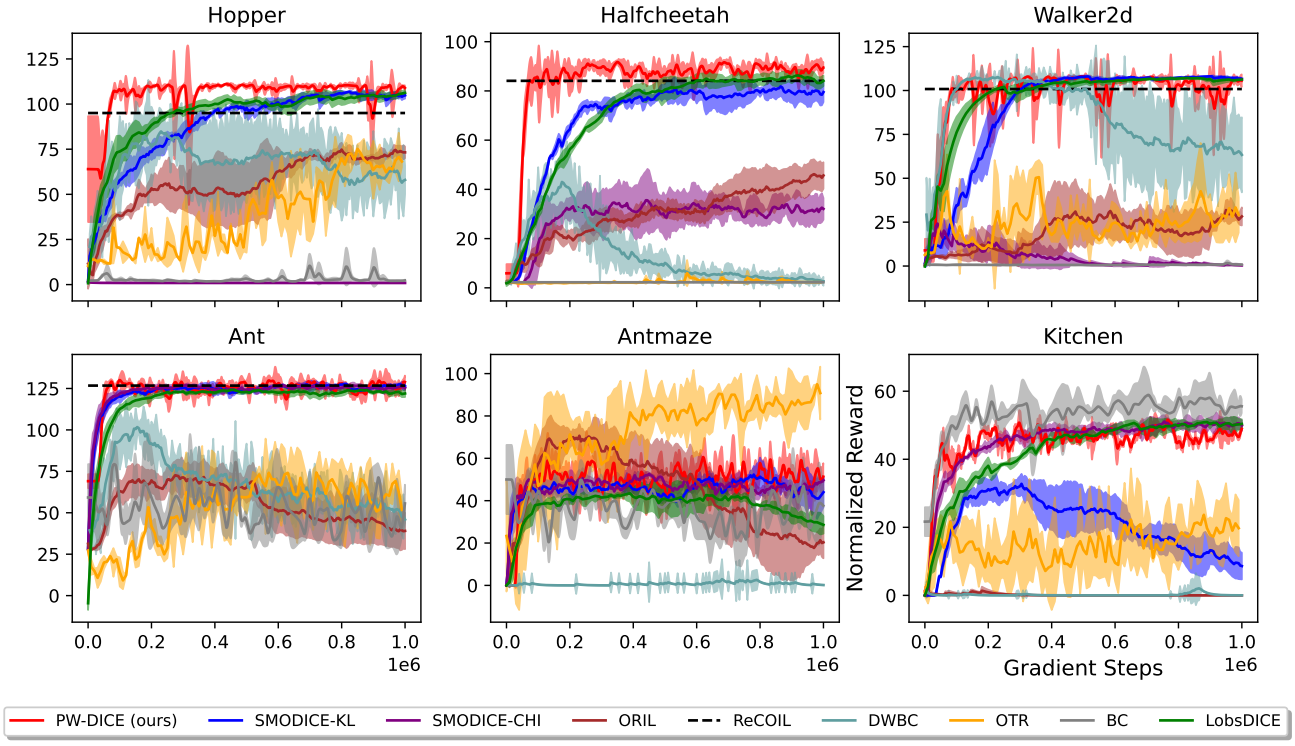


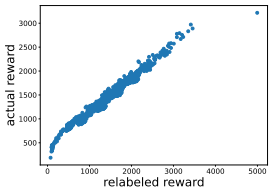
Figure 4. Performance comparison on the MuJoCo testbed. SMODICE-KL and SMODICE-CHI stand for variants of SMODICE using different f-divergences (KL or χ^2). Our method generally works the best (i.e., has the highest normalized reward) among all baselines.

Main Results. Fig. 4 shows the results on the MuJoCo testbed, where our method achieves performance comparable to or better than baselines on all four testbeds. SMODICE with KL-divergence and LobsDICE work decently well, while the other methods struggle.

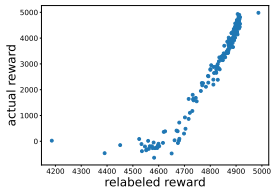
Note, OTR (Luo et al., 2023) struggles on most environments despite using the primal Wasserstein distance, which is probably because the assigned reward calculated by the Wasserstein distance is not always reasonable. See Fig. 5 for examples.

Is our design of distance metric useful? We illustrate the importance and effectiveness of our distance metric design through qualitative and quantitative studies. For a qualitative evaluation, we draw 4 different trajectories from the D4RL dataset of the MuJoCo hopper environment, and compare the t-SNE (Van der Maaten & Hinton, 2008) visualization result (for better readability, we only plot 150 steps of the trajectory). The result in Fig. 6 shows that our embedding successfully learns the topology of reachability between the states, which separates different trajectories and connects states in the same trajectory irrespective of their distance.

For a quantitative evaluation, we conduct an ablation study on the distance metric used in PW-DICE. Specifically, we test the result of PW-DICE with $c(s, s') = R(s)$, $c(s, s') = \|s - s'\|_2^2$ (Euclidean), $c(s, s') = 1 - \frac{s^T s'}{|s||s'|}$ (cosine similarity), $c(s, s')$ from contrastive learning and their combinations. The result is illustrated in Fig. 7. The result shows that both our design of distance and the combination of cosine similarity and $R(s)$ works well, while distance metrics with a single component fail (including

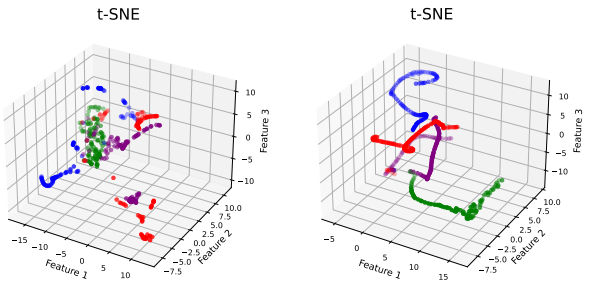


(a) hopper (success)



(b) halfcheetah (failure)

Figure 5. An illustration of successful (coherent with the OTR paper) and failing reward assignment in OTR (Luo et al., 2023). OTR performs Wasserstein matching between uniform distributions over the states of each trajectory in the task-agnostic dataset and the expert dataset, instead of between policy distributions. The reward is calculated from the matching result. Such a solution may fail to differentiate good and bad trajectories by giving similar rewards, as shown in the failure case **b)**.



(a) State (b) Contrastive Embedding

Figure 6. The t-SNE visualization of **a)** the states and **b)** their corresponding contrastive embeddings. Different colors stand for different trajectories. Note that the trajectories in the embedding space are separated despite their proximity in the original state space, and states along the same trajectory are connected despite being separated in the original state space.

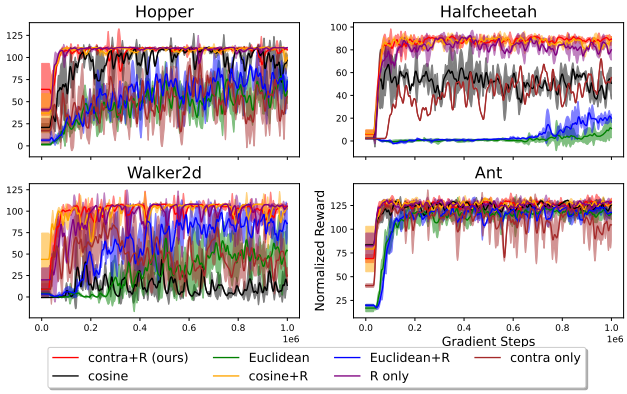


Figure 7. Ablations on the choice of distance metrics. Our choice of $c(s, s')$, which combines a contrastively learned distance and the discriminator-based component R, performs the best. The Euclidean distance fails in our scenario, which further proves the importance of using the primal form instead of the Rubinstein dual form.

Euclidean distance implied by Rubinstein duality).

Ablations on ϵ_1 and ϵ_2. In order to show the robustness of PW-DICE to the choice of ϵ_1 and ϵ_2, we conduct an ablation study on the MuJoCo environment. Specifically, we test $\epsilon_1 \in \{0.1, 0.5, 1\} \times \epsilon_2 \in \{0.1, 0.5, 1\}$. The result is shown in Fig. 8. While some choice of hyperparameters leads to failure, PW-DICE is generally robust to the selection of ϵ_1 and ϵ_2. Generally, ϵ_1 should be small to maintain good performance. See more ablations in Appendix E.

Robustness against distorted state representations. One important motivation for using a learned distance metric is that a fixed distance metric might be limited to the state representation. For example, the Euclidean distance could perform well when learning navigation for a point-mass, where coordinates are given as states. However, the Euclidean distance will no longer be accurate when some of the dimensions undergo scaling (e.g., due to metric changes from inches to meters). While scaling each dimension independently could be alleviated by state normalization, in this experiment we consider a more complicated *distortion* to the state representations.

More specifically, for state $s \in \mathbb{R}^{1\times n}$, we randomly generate a distortion matrix $D = 0.1I + D' \in \mathbb{R}^{n\times n}$, where each element of D' is independently and randomly sampled from $\mathcal{N}(0, 4^2)$. The new state exposed to the agent (both in the dataset and evaluation) is calculated as $s' = D's$. We compare our method against SMODICE on several MuJoCo environments. Results are shown in Fig. 9. We observe that our method is generally more robust to poor state representations than SMODICE.

5. Related Work

Wasserstein Distance for Imitation Learning. As a metric which is capable of leveraging geometric properties of distributions and which yields gradients for distributions with different support sets, the Wasserstein distance (also known as *Optimal Transport*) (Kantorovich, 1960) is a popular choice when studying distribution divergence minimization. It is widely used in IL/RL (Agarwal et al., 2021; Fickinger et al., 2022; Xiao et al., 2019; Dadashi et al., 2021; Garg et al., 2021). Among them, SoftDICE (Sun et al., 2021) is the most similar work to our PW-DICE, which also optimizes the Wasserstein distance under the DICE framework. However, SoftDICE and most Wasserstein-based IL algorithms (Sun et al., 2021; Xiao et al., 2019; Zhang et al., 2020; Liu et al., 2020) use Rubinstein-Kantorovich duality (Kantorovich & Rubinstein, 1958; Peyré & Cuturi, 2019), which limits the underlying distance metric to be Euclidean. There are a few methods optimizing the primal Wasserstein distance. For example, OTR (Luo et al., 2023) computes the primal Wasserstein distance between two trajectories and assigns rewards accordingly for offline RL. PWIL (Dadashi et al., 2021) uses greedy coupling to simplify the computation of the Wasserstein distance. However, the former struggles in our experimental settings, and the latter only optimizes an upper bound of the Wasserstein distance. Moreover, both methods only use fixed heuristic distance metrics, such as Euclidean (Dadashi et al., 2021) and cosine (Luo et al., 2023). Our PW-DICE addresses these issues.

Offline Imitation Learning from Observation. Offline Learning from Observation (LfO) aims to learn from expert observations with no labeled action, which is useful in robotics where the expert action is either not available (e.g.,

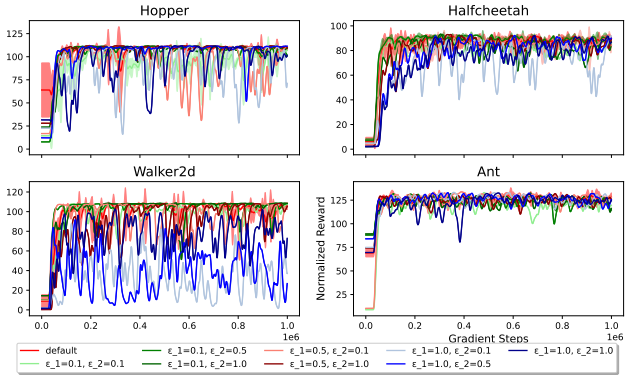


Figure 8. Ablation of ϵ_1 and ϵ_2 on the MuJoCo testbed: $\epsilon_1 = 0.1$ is marked in green, $\epsilon_1 = 0.5$ is marked in red, and $\epsilon_1 = 1.0$ is marked in blue. The deeper the color is, the larger ϵ_2 is. Our method is generally robust to hyperparameter changes, though some choice results in failure. Generally, large ϵ_1 leads to worse performance.

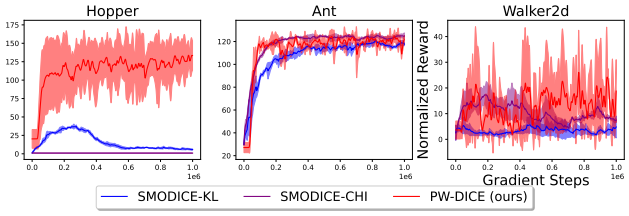


Figure 9. Performance comparison between our proposed method, PW-DICE, and SMODICE under distorted state representations. Our method generally outperforms SMODICE.

in videos (Pari et al., 2022)) or not applicable (e.g., from a different embodiment (Sermanet et al., 2017)). Three major directions are present in this area: 1) offline planning or RL with assigned, similarity-based reward (Torabi et al., 2018; Kumar et al., 2019); 2) occupancy divergence minimization, which includes iterative inverse-RL methods (Zolna et al., 2020; Xu & Denil, 2019; Torabi et al., 2019) and DICE (Ma et al., 2022; Kim et al., 2022a;b; Lee et al., 2021; Zhu et al., 2020); 3) action pseudo-labeling, where the missing actions are predicted with an inverse dynamic model (Sermanet et al., 2017; Chen et al., 2019; Wu et al., 2019). Our PW-DICE falls in the second category but generalizes over SMODICE, unifies f-divergence and Wasserstein, and empirically improves upon existing methods.

Contrastive Learning for State Representations. Contrastive learning methods, such as InfoNCE (Wan et al., 2021) and SIMCLR (Chen et al., 2020), aim to find a good representation that satisfies similarity and dissimilarity constraints between particular pairs of data points. Contrastive learning is widely used in reinforcement learning, especially with visual input (Laskin et al., 2020; Pari et al., 2022; Sermanet et al., 2017) and for meta RL (Fu et al., 2020a) to improve the generalizability of agents and mitigate the curse of dimensionality. In these methods, similarity constraints can come from different augmentations of the same state (Laskin et al., 2020; Pari et al., 2022), multiview alignment (Sermanet et al., 2017), consistency after reconstruction (Zhu et al., 2023), or task context (Fu et al., 2020a). Different from prior work, PW-DICE uses contrastive learning to identify a good distance metric considering state reachability, while still adopting the reward from the DICE work.

6. Conclusion

In this paper, we propose PW-DICE, a DICE method that uses the primal form of the Wasserstein distance and a contrastively learned distance metric. By adding adequate pessimistic regularizers, we formulate an unconstrained convex optimization and retrieve the policy using weighted behavior cloning. Our method is a generalization of SMODICE, unifying f-divergence and Wasserstein minimization in imitation learning. This generalization enables better performance than multiple baselines, such as SMODICE (Ma et al., 2022) and LobsDICE (Kim et al., 2022b).

Limitations and Future Directions. In order to obtain an unconstrained optimization formulation, we add KL terms to the objective. This introduces a logsumexp into the final objective. Some studies argue that logsumexp adds instability to the optimization, due to the use of minibatches: minibatch gradient estimates for logarithms and exponentials of expectations are biased (Sun et al., 2021). Although we did not observe this and it has been found to be tolerable in prior work (Ma et al., 2022), this may be a potential shortcoming for PW-DICE in even more challenging environments. Thus, one next step is to find a more robust formulation while maintaining the beneficial properties of PW-DICE.

Acknowledgements

This work was supported in part by NSF under Grants 2008387, 2045586, 2106825, MRI 1725729, NIFA award 2020-67021-32799, the Jump ARCHES endowment through the Health Care Engineering Systems Center at Illinois and the OSF Foundation, and the IBM-Illinois Discovery Accelerator Institute.

Impact Statement

Our work automates decision-making processes by utilizing expert observations as well as past experience data. While our effort improves the efficiency of automated task-solving, it could also lead to negative societal impacts in several aspects. For example, since our work is mostly tested on

locomotion tasks, there exists a potential risk of harmful applications (e.g., military) of our proposed decision-making techniques. Also, the improvement of automated decision-making may potentially result in a reduction of job opportunities.

References

Agarwal, R., Machado, M. C., Castro, P. S., and Bellemare, M. G. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *ICLR*, 2021.

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. Differentiable convex optimization layers. In *NeurIPS*, 2019.

ApS, M. *The MOSEK optimization toolbox for CVXPY manual.*, 2019. URL https://docs.mosek.com/9.0/faq/index.html.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge University Press, 2004.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI gym, 2016.

Chang, M., Gupta, A., and Gupta, S. Semantic visual navigation by watching YouTube videos. In *NeurIPS*, 2020.

Chen, A. S., Nair, S., and Finn, C. Learning generalizable robotic reward functions from "in-the-wild" human videos. In *RSS*, 2021a.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 2021b.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *NeurIPS*, 2020.

Chen, X., Li, S., Li, H., Jiang, S., Qi, Y., and Song, L. Generative adversarial user model for reinforcement learning based recommendation system. In *ICML*, 2019.

Dadashi, R., Hussenot, L., Geist, M., and Pietquin, O. Primal wasserstein imitation learning. In *ICLR*, 2021.

Dai, B., He, N., Pan, Y., Boots, B., and Song, L. Learning from conditional distributions via dual embeddings. In *AISTATS*, 2017.

Fickinger, A., Cohen, S., Russell, S., and Amos, B. Cross-domain imitation learning via optimal transport. In *ICLR*, 2022.

Fu, H., Tang, H., Hao, J., Chen, C., Feng, X., Li, D., and Liu, W. Towards effective context for meta-reinforcement learning: An approach based on contrastive learning. In *AAAI*, 2020a.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4RL: Datasets for deep data-driven reinforcement learning. *ArXiv:2004.07219*, 2020b.

Garg, D., Chakraborty, S., Cundy, C., Song, J., and Ermon, S. IQ-Learn: Inverse soft-Q learning for imitation. In *NeurIPS*, 2021.

Ghasemipour, S., Zemel, R., and Gu, S. A divergence minimization perspective on imitation learning methods. In *CoRL*, 2019.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2023. URL https://www.gurobi.com.

Hakhamaneshi, K., Zhao, R., Zhan, A., Abbeel, P., and Laskin, M. Hierarchical few-shot imitation with skill transition models. In *ICLR*, 2022.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *NIPS*, 2016.

Jin, Y., Yang, Z., and Wang, Z. Is pessimism provably efficient for offline RL? In *ICML*, 2021.

Kantorovich, L. and Rubinstein, G. S. On a space of totally additive functions. *Vestnik Leningradskogo Universiteta*, 1958.

Kantorovich, L. V. Mathematical methods of organizing and planning production. *Management science*, 1960.

Kim, G., Seo, S., Lee, J., Jeon, W., Hwang, H., Yang, H., and Kim, K. DemoDICE: Offline imitation learning with supplementary imperfect demonstrations. In *ICLR*, 2022a.

Kim, G.-H., Lee, J., Jang, Y., Yang, H., and Kim, K. LobsDICE: Offline learning from observation via stationary distribution correction estimation. In *NeurIPS*, 2022b.

Kiran, B., Sobh, I., Talpaert, V., Mannion, P., Sallab, A., Yogamani, S., and Perez, P. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

Kostrikov, I., Agrawal, K. K., Dwibedi, D., Levine, S., and Tompson, J. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *ICLR*, 2019.

Kostrikov, I., Nachum, O., and Tompson, J. Imitation learning via off-policy distribution matching. In *ICLR*, 2020.

Kostrikov, I., Nair, A., and Levine, S. Conservative Q-learning for offline reinforcement learning. In *ICLR*, 2022a.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit Q-learning. In *ICLR*, 2022b.

Kumar, A., Gupta, S., and Malik, J. Learning navigation subroutines from egocentric videos. In *CoRL*, 2019.

Laskin, M., Srinivas, A., and Abbeel, P. CURL: Contrastive unsupervised representations for reinforcement learning. In *ICML*, 2020.

Lee, J., Jeon, W., Lee, B.-J., Pineau, J., and Kim, K.-E. OptiDICE: Offline policy optimization via stationary distribution correction estimation. In *ICML*, 2021.

Liu, F., Ling, Z., Mu, T., and Su, H. State alignment-based imitation learning. In *ICLR*, 2020.

Luo, Y., Jiang, Z., Cohen, S., Grefenstette, E., and Deisenroth, M. P. Optimal transport for offline imitation learning. In *ICLR*, 2023.

Ma, Y. J., Shen, A., Jayaraman, D., and Bastani, O. Smodice: Versatile offline imitation learning via state occupancy matching. In *ICML*, 2022.

Nachum, O., Chow, Y., Dai, B., and Li, L. DualDICE: Behavior-agnostic estimation of discounted stationary distribution corrections. In *NeurIPS*, 2019.

Pari, J., Shafiullah, N. M. M., Arunachalam, S. P., and Pinto, L. The surprising effectiveness of representation learning for visual imitation. In *RSS*, 2022.

Peyré, G. and Cuturi, M. Computational optimal transport. *Foundations and Trends in Machine Learning*, 2019.

Polyanskiy, Y. f-divergences, 2020. URL https://people.lids.mit.edu/yp/homepage/data/LN_fdiv.pdf.

Sermanet, P., Lynch, C., Hsu, J., and Levine, S. Time-contrastive networks: Self-supervised learning from multi-view observation. In *CVPRW*, 2017.

Sikchi, H. S., Zhang, A., and Niekum, S. Imitation from arbitrary experience: A dual unification of reinforcement and imitation learning methods. In *ICLR*, 2024.

Stanczuk, J., Etmann, C., Kreusser, L., and Schonlieb, C.-B. Wasserstein GANs work because they fail (to approximate the wasserstein distance). *ArXiv:2103.01678*, 2021.

Sun, M., Mahajan, A., Hofmann, K., and Whiteson, S. Softdice for imitation learning: Rethinking off-policy distribution matching. *ArXiv:2106.03155*, 2021.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. In *IJCAI*, 2018.

Torabi, F., Warnell, G., and Stone, P. Generative adversarial imitation from observation. In *ICML Workshop on Imitation, Intent, and Interaction*, 2019.

Van der Maaten, L. and Hinton, G. Visualizing data using t-SNE. *JMLR*, 2008.

Wan, C., Zhang, T., Xiong, Z., and Ye, H. Representation learning for fault diagnosis with contrastive predictive coding. In *CAA Symposium on Fault Detection, Supervision, and Safety for Technical Processes (SAFEPROCESS)*, 2021.

Wu, A., Piergiovanni, A., and Ryoo, M. S. Model-based behavioral cloning with future image similarity learning. In *CoRL*, 2019.

Xiao, H., Herman, M., Wagner, J., Ziesche, S., Etesami, J., and Linh, T. H. Wasserstein adversarial imitation learning. *arXiv preprint arXiv:1906.08113*, 2019.

Xu, D. and Denil, M. Positive-unlabeled reward learning. In *CoRL*, 2019.

Xu, H., Zhan, X., Yin, H., and Qin, H. Discriminator-weighted offline imitation learning from suboptimal demonstrations. In *NeurIPS*, 2022.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2019.

Zhang, M., Wang, Y., Ma, X., Xia, L., Yang, J., Li, Z., and Li, X. Wasserstein distance guided adversarial imitation learning with reward shape exploration. In *Data Driven Control and Learning Systems (DDCLS)*, 2020.

Zhu, J., Xia, Y., Wu, L., Deng, J., Zhou, W., Qin, T., Liu, T.-Y., and Li, H. Masked contrastive representation learning for reinforcement learning. *IEEE TPAMI*, 2023.

Zhu, Z., Lin, K., Dai, B., and Zhou, J. Off-policy imitation learning from observations. In *NeurIPS*, 2020.

Zolna, K., Novikov, A., Konyushkova, K., Gulcehre, C., Wang, Z., Aytar, Y., Denil, M., de Freitas, N., and Reed, S. E. Offline learning from demonstrations and unlabeled experience. In *NeurIPS Workshop on Offline Reinforcement Learning*, 2020.

Algorithm 1 PW-DICE

Input: Expert state-only dataset E, task-agnostic state-action dataset I, distance metric $c : |S| \times |S| \to \mathbb{R}$
Input: Triple head network for dual variable $\lambda_{\theta_1}(s, h) : |S| \times \{0, 1, 2\} \to \mathbb{R}$ parameterized by θ_1, actor $\pi_{\theta_2}(a|s)$ parameterized by θ_2, the number of epoches N, batch size B, learning rate η.
Initialize initial state dataset $I_{\text{ini}} = \{\}$.
for $\tau \in I$ **do**
  Add first state of τ to I_{ini}
end for
for epoch $= 1$ **to** N **do**
  for $(s, a, s') \in I$ **do**
    Sample state s^I from I, state s^E from E
    $l_1 \leftarrow \frac{1}{\epsilon_1}(\lambda_{\theta_1}(s^I, 1) + \lambda_{\theta_1}(s^E, 2) - c(s^I, s^E)) - \log B$
    $l_2 \leftarrow \frac{1}{\epsilon_2}(-\gamma\lambda_{\theta_1}(s', 0) + \lambda_{\theta_1}(s, 0) - \lambda_{\theta_1}(s, 1)) - \log B$
    Sample initial states s_{ini} from I_{ini}
    $l_3 \leftarrow -\frac{1}{B}[(1-\gamma)\lambda_{\theta_1}(s_{\text{ini}}, 0) + \lambda_{\theta_1}(s^E, 2)]$
    $l \leftarrow \text{logsumexp}(l_1) + \text{logsumexp}(l_2) + l_3$
    $\theta_1 \leftarrow \theta_1 - \eta\frac{\partial l}{\partial \theta_1}$
  end for
  for $(s, a, s') \in I$ **do**
    $v \leftarrow \frac{1}{\epsilon_2}(-\gamma\lambda_{\theta_1}(s', 0) + \lambda_{\theta_1}(s, 0) - \lambda_{\theta_1}(s, 1))$
    $l \leftarrow \exp(v) \log \pi_{\theta_2}(a|s)$
    $\theta_2 \leftarrow \theta_2 - \eta\frac{\partial l}{\partial \theta_2}$
  end for
end for

Appendix: Offline Imitation from Observation via Primal Wasserstein State Occupancy Matching

The appendix is organized as follows. We first present the pseudo-code of PW-DICE in Sec. A. Then, we rigorously introduce the most important mathematical concepts of our work in Sec. B, which include state, state-action, and state-pair occupancy, as well as f-divergences and Fenchel conjugate. After that, in Sec. C, we provide detailed derivations omitted in the main paper, as well as the corresponding proofs. In Sec. D, we provide a detailed description of our experiments. In Sec. E, we provide additional experimental results, including auxiliary metrics, experimental results using an identical softmax expert (which is actually suboptimal, see Sec. E.1.2) as LobsDICE (Kim et al., 2022b) in the tabular experiment, and ablations in MuJoCo environments. In Sec. F, we summarize our notation. Finally, in Sec. G, we list the computational resource that we use during the training process.

A. Pseudo-code

Alg. 1 details the training process of our main algorithm. Upon implementation, we normalize coefficient v over the whole task-agnostic dataset I for better stability. See Sec. D.2 for the implementation detail of contrastive learning for the distance metric.

B. Mathematical Concepts

In this section, we introduce three important concepts used in the paper, which are state/state-action/state-pair occupancy, f-divergence, and Fenchel conjugate. The first one is the key concept used throughout this work, the second is used in our motivations, and the last is used in Sec. C.2.

B.1. State, State-Action, and State-Pair Occupancy

Consider an MDP (S, A, T, r, γ) with initial state distribution p_0 and infinite horizon; at the t-th timestep, we denote the current state as s_t and the action as a_t. Then, with a fixed policy π, the probability of $\Pr(s_t = s)$ and $\Pr(a_t = a)$

for any s, a are determined. Based on this, the *state occupancy*, which is the state visitation frequency under policy π, is defined as $d_s^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \Pr(s_t = s)$. Similarly, we define the *state-action occupancy* as $d_{sa}^\pi(s,a) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \Pr(s_t = s, a_t = a)$. Some work such as LobsDICE also uses *state-pair occupancy*, which is defined as $d_{ss}^\pi(s,s') = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \Pr(s_t = s, s_{t+1} = s')$. In this work, we denote the average policy that generates the task-agnostic dataset I as π^I with state occupancy d_s^I and state-action occupancy d_{sa}^I, and the expert policy that generates the expert dataset E as π^E with state occupancy d_s^E.

B.2. f-divergence

The f-divergence is a measure of distance between probability distributions p, q and is widely used in the machine learning community (Ghasemipour et al., 2019). For two probability distributions p, q on domain $\mathcal{X}$ based on any continuous and convex function f, the f-divergence between p and q is defined as

$$D_f(p\|q) = \mathbb{E}_{x\sim q}\left[f\left(\frac{p(x)}{q(x)}\right)\right]. \tag{8}$$

For instance, when $f(x) = x\log x$, we have $D_f(p\|q) = \mathbb{E}_{x\sim q}\frac{p(x)}{q(x)}\log\frac{p(x)}{q(x)} = \mathbb{E}_{x\sim p}\log\frac{p(x)}{q(x)}$, which induces the KL-divergence. When $f(x) = (x-1)^2$, we have $D_f(p\|q) = \mathbb{E}_{x\sim q}((\frac{p(x)-q(x)}{q(x)})^2)$, which induces the χ^2-divergence.

B.3. Fenchel Conjugate

Fenchel conjugate is widely used in DICE methods for either debiasing estimations (Nachum et al., 2019) or solving formulations with stronger constraints to get numerically more stable objectives (Ma et al., 2022). PW-DICE uses the Fenchel conjugate for the latter. For a vector space Ω and a convex, differentiable function $f : \Omega \to \mathbb{R}$, the Fenchel conjugate of $f(x)$ is defined as

$$f_*(y) = \max_{x\in\Omega}\langle x, y\rangle - f(x), \tag{9}$$

where $\langle\cdot,\cdot\rangle$ is the inner product over Ω.

C. Mathematical Derivations

In this section, we provide the detailed derivations omitted in the main paper due to the page limit. In Sec. C.1, we briefly introduce SMODICE to clarify the motivation of using Wasserstein distance and how SMODICE is related to our proposed PW-DICE. In Sec. C.2, we provide a detailed derivation of our objective, omitted in Sec. 3.2. Sec. C.3 and Sec. C.4 are complements to Sec. C.2: in Sec. C.3, we provide a detailed derivation on the elements of condensed representation of constraints, A and b, and in Sec. C.4, we explain why additional constraints are applied during one step of the derivation process while the optimal solution remains the same. Finally, in Sec. C.5, we provide a detailed proof for the claim that our method is a generalization of SMODICE in Sec. 3.2.

C.1. SMODICE

SMODICE (Ma et al., 2022) is a state-of-the-art offline LfO method. It minimizes the f-divergence between the state occupancy of the learner's policy π and the expert policy π^E, i.e., the objective is

$$\min_\pi D_f(d_s^\pi(s)\|d_s^E(s)), \text{s.t. } \pi \text{ is feasible.} \tag{10}$$

Here, the feasibility of π is the same as the Bellman flow constraint (the second row of constraints in Eq. (1)) in the main paper. To take the only information source of environment dynamics, which is the task-agnostic dataset I, into account, the objective is relaxed to

$$\max_\pi \mathbb{E}_{s\sim d^\pi}\log\frac{d_s^E(s)}{d_s^I(s)} - D_f(d_{sa}^\pi(s,a)\|d_{sa}^I(s,a)), \text{s.t. } \pi \text{ is a feasible policy.} \tag{11}$$

Here, D_f can be any divergence that is not smaller than KL-divergence (SMODICE mainly studies χ^2-divergence). The first term, $\log \frac{d_s^E(s)}{d_s^I(s)}$ indicates the relative importance of the state. The more often the expert visits a particular state s than non-expert policies, the more possible that s is a desirable state. Reliance on such a ratio introduces a theoretical limitation: the assumption that $d_s^I(s) > 0$ wherever $d_s^E(s) > 0$ must be made, which does not necessarily hold in a high-dimensional space. Thus, we introduce a hyperpamater of α to mix the distribution in the denominator in our reward design.

By transforming the constrained problem into an unconstrained problem in the Lagrange dual space, SMODICE optimizes the following objective (assuming the use of KL-divergence):

$$\min_V (1-\gamma)\mathbb{E}_{s\sim p_0}[V(s)] + \log \mathbb{E}_{(s,a,s')\sim I} \exp\left[\log \frac{d_s^E(s)}{d_s^I(s)} + \gamma V(s') - V(s)\right], \tag{12}$$

where p_0 is the initial state distribution and γ is the discount factor. As stated in Sec. 3.2, this objective is a special case of PW-DICE with $c(s,s') = \log \frac{d_s^E(s)}{d_s^I(s)}$, $\epsilon_2 = 1, \epsilon_1 \to 0$. LobsDICE (Kim et al., 2022b) is similar in spirit; however, it minimizes the state-pair divergence $\mathrm{KL}(d_{ss}^\pi \| d_{ss}^E)$ instead.

C.2. Detailed Derivation of Our Objective

As mentioned in Eq. (1) in Sec. 3.2, our primal objective is

$$\begin{aligned}
&\min_{\Pi,\pi} \sum_{s_i\in S}\sum_{s_j\in S} \Pi(s_i,s_j)c(s_i,s_j), \text{s.t. } d_{sa}^\pi \geq 0, \Pi \geq 0;\\
&\forall s \in S, d_s^\pi(s) = (1-\gamma)p_0(s) + \gamma \sum_{\bar{s},\bar{a}} d_{sa}^\pi(\bar{s},\bar{a})p(s|\bar{s},\bar{a});\\
&\forall s_i, s_j \in S, \sum_k \Pi(s_k,s_j) = d_s^E(s_j), \sum_k \Pi(s_i,s_k) = d_s^\pi(s_i).
\end{aligned} \tag{13}$$

Before smoothing our objective, for readability, we rewrite our main objective in Eq. (13) as an LP problem over a single vector $x = \begin{bmatrix} \Pi \\ d_{sa}^\pi \end{bmatrix} \in \mathbb{R}^{|S|\times(|S|+|A|)}$, where $\Pi \in \mathbb{R}^{|S|\times|S|}$ and $d_{sa}^\pi \in \mathbb{R}^{|S|\times|A|}$ are flattened in a row-first manner. Correspondingly, we extend the distance c between states to $c' : (|S|(|S|+|A|)) \times (|S|(|S|+|A|)) \to \mathbb{R}$, such that $c' = c$ on the original domain of c and $c' = 0$ otherwise. Further, we summarize all linear equality constraints in $Ax = b$. Eq. (1) is then equivalent to

$$\min_{x\geq 0} (c')^T x \quad \text{s.t.} \quad Ax = b, x \geq 0. \tag{14}$$

It is easy to see that the Lagrange dual form of Eq. (14) is also a constrained optimization. In order to convert the optimization to an unconstrained one, we modify the objective as follows (same as Eq. (2) in the main paper):

$$\begin{aligned}
\min_x (c')^T x + \epsilon_1 D_f(\Pi \| U) + \epsilon_2 D_f(d_{sa}^\pi \| d_{sa}^I),\\
\text{s.t. } Ax = b, x \geq 0,
\end{aligned} \tag{15}$$

where $U(s,s') = d_s^E(s) d_s^I(s')$, i.e., U is the product of two independent distributions d_s^E and d_s^I. $\epsilon_1 > 0, \epsilon_2 > 0$ are hyperparameters, and D_f can be any f-divergence. Note, although an f-divergence is used, unlike SMODICE (Ma et al., 2022) or LobsDICE (Kim et al., 2022b), such formulation does not require data coverage of the task-agnostic data over the expert data. The two regularizers are "pessimistic," which encourages the agents to stay within the support set of the dataset. This has been used in offline IL/RL (Jin et al., 2021).

With the regularized objective in Eq. (2), we now consider its Lagrange dual problem:

$$\max_\lambda \min_{x\geq 0} L(\lambda, x), \text{ where } L(\lambda,x) = (c')^T x + \epsilon_1 D_f(\Pi\|U) + \epsilon_2 D_f(d_{sa}^\pi \| d_{sa}^I) - \lambda^T(Ax - b). \tag{16}$$

While Eq. (16) only has the non-negativity constraint, its domain is the non-negative numbers. Thus the objective can be optimized as being unconstrained. To obtain a practical and stable solution, a single-level optimization is preferred. To do

so, one could consider using the KKT condition (Boyd & Vandenberghe, 2004), and set the derivative of the inner-level optimization to 0. However, such an approach will lead to an exp function in the objective (Polyanskiy, 2020; Kim et al., 2022b), which is numerically unstable (Kim et al., 2022b). To avoid this, we first rewrite Eq. (16) *with negated* $L(\lambda, x)$ to separate Π and d^π_{sa} in x:

$$\begin{aligned}&\min_{\lambda} \max_{x \geq 0} -L(\lambda, x) \\ &\min_{\lambda} \left\{ \epsilon_1 \max_{\Pi \in \Delta(S^2)} \left[\frac{(A_1^T \lambda - c)}{\epsilon_1}^T \Pi - D_f(\pi \| U) \right] + \epsilon_2 \max_{d^\pi_{sa} \in \Delta(S \cdot A)} \left[\frac{(A_2^T \lambda)}{\epsilon_2}^T d^\pi_{sa} - D_f(d^\pi_{sa} \| d^I_{sa}) \right] - b^T \lambda \right\}.\end{aligned} \tag{17}$$

In Eq. (17), we have $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$, where $A_1 \in \mathbb{R}^{(|S|\times|S|)\times M}$, $A_2 \in \mathbb{R}^{(|S|\times|A|)\times M}$, and $M = 3|S|$ is the number of equality constraints in the primal form. See Sec. C.3 for elements in A, A_1, A_2, and b. Two points are worth noting in Eq. (17).

First, we append two extra constraints, which are $\Pi \in \Delta, d^\pi_{sa} \in \Delta$. These constraints do not affect the final result for the following fact:

Lemma 1. *For any MDP and feasible expert policy π^E, the inequality constraints in Eq. (1) with $\Pi \geq 0, d^\pi_{sa} \geq 0$ and $\Pi \in \Delta, d^\pi_{sa} \in \Delta$ are equivalent.*

The detailed proof of Lemma 1 is given in Appendix C.4. In a word, the optimal solution of Eq. (16), as long as it satisfies all constraints in the primal form, must have $\Pi \in \Delta, d^\pi_{sa} \in \Delta$.

Second, we decompose the max operator into two independent maximizations, as the equality constraints that correlate Π and d^π_{sa} are all relaxed in the dual. With Eq. (17), we now apply the following theorem from SMODICE (Ma et al., 2022):

Theorem C.1. *With mild assumptions (Dai et al., 2017), for any f-divergence D_f, probability distribution p, q on domain $\mathcal{X}$ and function $y : \mathcal{X} \to \mathbb{R}$, we have*

$$\max_{p \in \Delta(\mathcal{X})} \mathbb{E}_{x\sim p}[y(x)] - D_f(p\|q) = \mathbb{E}_{x\sim q}[f_*(y(x))]. \tag{18}$$

For maximizer $p^*(x) = \arg\max_{p\in\Delta(\mathcal{X})} \mathbb{E}_{x\sim q}[f_*(y(x))]$, we have $p^*(x) = q(x) f'_*(y(x))$, where $f_*(\cdot)$ is the Fenchel conjugate of f, and f'_* is its derivative.

A complete proof can be found at theorem 7.14^* in Polyanskiy (2020). The rigorous definition of f-divergence and Fenchel conjugate are in Appendix B. For this work, we mainly consider KL-divergence as D_f, which corresponds to $f(x) = x \log x$, and $f_*(x) = \text{logsumexp}(x)$ to be the Fenchel dual function with $x \in \Delta$ (Boyd & Vandenberghe, 2004)[4]. With Thm. C.1, we set $p = \Pi, x = \lambda, y(x) = \frac{A_1^T\lambda - c}{\epsilon_1}$ for the first max operator, and set $p = d^\pi_{sa}, x = \lambda, y(x) = \frac{A_2^T\lambda}{\epsilon_2}$ for the second max operator. Then, we get the following single-level convex objective:

$$\min_{\lambda} \epsilon_1 \log \mathbb{E}_{s_i\sim I, s_j\sim E} \exp\left(\frac{(A_1^T\lambda - c)^T}{\epsilon_1}\right) + \epsilon_2 \log \mathbb{E}_{(s_i,a_j)\sim I} \exp\left(\frac{A_2^T\lambda}{\epsilon_2}\right) - b^T\lambda, \tag{19}$$

with which, by considering the components of A_1, A_2 and b in Appendix C.3, we have our final objective stated in Eq. (3) and maximizer stated in Sec. 3.2.

C.3. Components of A, b in Eq. (14)

In this subsection, we discuss in detail the entries of A, b in Eq. (14). In Eq. (14), we summarize all equality constraints in Eq. (1) as $Ax = b$, $x = \begin{bmatrix} \Pi \\ d^\pi_{sa} \end{bmatrix}$, where Π, d^π_{sa} are flattened in a row-first manner. Thus, we have $x_{:i|S|+j} = \Pi(s_i, s_j)$, and $x_{|S|^2+i|A|+j} = d^\pi_{sa}(s_i, a_j)$.

[4] χ^2-divergence does not work as well as KL-divergence in MuJoCo environments. See Appendix E.2 for details.

We further assume that in A and b, the first $|S|$ rows are the Bellman flow constraints

$$\forall s, \sum_a d^\pi_{sa}(s,a) - \gamma \sum_{\bar{s},\bar{a}} p(s|\bar{s},\bar{a}) d^\pi_{sa}(\bar{s},\bar{a}) = (1-\gamma)p_0(s). \tag{20}$$

The second $|S|$ rows are the $\sum_j \Pi(s_i, s_j) = d^\pi_s(s_i)$ marginal constraints

$$\forall s, \sum_{s'} \Pi(s,s') = \sum_a d^\pi_{sa}(s,a). \tag{21}$$

The third $|S|$ rows are the $\sum_i \Pi(s_i, s_j) = d^E_s(s_j)$ constraints

$$\forall s, \sum_{s'} \Pi(s',s) = \sum_a d^E_{sa}(s,a). \tag{22}$$

Thus, we have $A_{i,|S|^2+j|A|+k} = -\gamma p(s_i|s_j,a_k)$ for $i \in \{1,2,\dots,|S|\}$, $A_{i,|S|^2+i|A|:|S|^2+(i+1)|A|} = 1$ for $i \in \{1,2,\dots,|S|\}$ (Eq. (20)), $A_{i+|S|,i|S|+j} = 1$ for $i \in \{1,2,\dots,|S|\}$, $A_{i+|S|,|S|^2+i|A|+j} = -1$ (Eq. (21)), and $A_{i+2|S|,j|S|+i} = 1$ (Eq. (22)). Other entries of A are 0. A_1 in Eq. (17) are the first $|S| \times |S|$ rows of A, and A_2 are the last $|S| \times |A|$ rows of A.

For vector b, we have

$$b = \begin{bmatrix} (1-\gamma)p_0 \\ 0 \\ d^E_s \end{bmatrix}. \tag{23}$$

C.4. Lemma 1

In this section, we provide a proof of Lemma 1 used in Appendix C.2. The Lemma reads as follows:

Lemma 1. *For any MDP and feasible expert policy π^E, the inequality constraints in Eq. (1) with $\Pi \geq 0, d^\pi_{sa} \geq 0$ and $\Pi \in \Delta, d^\pi_{sa} \in \Delta$ are equivalent.*

Proof. According to the equality constraint, $\sum_s \Pi(s,s') = d^E_s(s')$ for any s'. Thus, we have $\sum_{s'}\sum_s \Pi(s,s') = \sum_{s'} d^E_s(s') = 1$ by the definition of state occupancy. Thus $\Pi \geq 0$ is equivalent to $\Pi \geq \Delta$. Similarly, by summing over both sides of the Bellman flow equality constraint, we have

$$\begin{aligned}
\sum_s d^\pi_s(s) &= \sum_s (1-\gamma)p_0(s) + \sum_s \gamma \sum_{\bar{s},\bar{a}} d^\pi_{sa}(\bar{s},\bar{a}) p(s|\bar{s},\bar{a}) \\
\sum_{s,a} d^\pi_{sa}(s,a) &= (1-\gamma) + \gamma \sum_s \sum_{\bar{s},\bar{a}} d^\pi_{sa}(\bar{s},\bar{a}) p(s|\bar{s},\bar{a}) \\
\sum_{s,a} d^\pi_{sa}(s,a) &= (1-\gamma) + \gamma \sum_{s'} \sum_{s,a} d^\pi_{sa}(s,a) p(s'|s,a) \\
\sum_{s,a} d^\pi_{sa}(s,a)(1-\gamma\sum_{s'} p(s'|s,a)) &= 1-\gamma \\
\sum_{s,a} d^\pi_{sa}(s,a) &= 1
\end{aligned} \tag{24}$$

given that p_0 and the transition function are legal. Thus, $d^\pi_{sa} \geq 0$ is equivalent to $d^\pi_{sa} \in \Delta$.

□

Intuitively, by adding the extra constraints, we can assume that redundant equality constraints exist in Eq. (1), and they are not relaxed in the Lagrange dual. By imposing more strict constraints over the dual form, the Fenchel conjugate yields a numerically more stable formulation.

C.5. Proof of Generalization over SMODICE

In Sec. 3.2, we claim that our proposed method, PW-DICE, is a generalization over SMODICE. More specifically, we have the following claim and corollary:

Claim 1. *If* $c(s_i, s_j) = -\log \frac{d_s^E(s_i)}{d_s^I(s_i)}, \epsilon_2 = 1$*, then as* $\epsilon_1 \to 0$*, Eq. (3) is equivalent to the SMODICE objective with KL divergence.*

Corollary C.2. *If* $c(s_i, s_j) = -\log \frac{d_s^E(s_i)}{d_s^I(s_i)}, \epsilon_2 = 1$*, then as* $\epsilon_1 \to 0$*, Eq. (17) is equivalent to SMODICE with any* f*-divergence.*

We first provide a simple proof from the primal perspective:

Proof. (Primal Perspective) According to Eq. (11) and Eq. (1), the SMODICE and PW-DICE primal objectives are as follows:

$$\begin{aligned} &\min_x (c')^T x + \epsilon_1 D_f(\Pi \| U) + \epsilon_2 D_f(d_{sa}^\pi \| d_{sa}^I), \text{s.t. } Ax = b, x \geq 0; \text{ (PW-DICE)} \\ &\max_\pi \mathbb{E}_{s\sim d^\pi} \log \frac{d_s^E(s)}{d_s^I(s)} - D_f(d_{sa}^\pi(s,a) \| d_{sa}^I(s,a)), \text{s.t. } \pi \text{ is a feasible policy. (SMODICE)} \end{aligned} \tag{25}$$

Here, $x = \begin{bmatrix} d_s^\pi \\ \Pi \end{bmatrix}$. Note: 1) $Ax = b, x \geq 0$ contains three equality constraints: the Bellman flow equation (which is the same as "π is a feasible policy"), $\sum_{s'} \Pi(s, s') = d_s^\pi(s)$, and $\sum_s \Pi(s, s') = d^E(s')$; 2) $(c')^T x = \sum_{s,s'} c(s, s')\Pi(s, s')$. Thus, we have

$$\sum_s \sum_{s'} c(s, s')\Pi(s, s') = \sum_s \log \frac{d_s^E(s)}{d_s^I(s)} \sum_{s'} \Pi(s, s') = -\mathbb{E}_{s\sim d_s^\pi} \log \frac{d_s^E(s)}{d_s^I(s)}. \tag{26}$$

Therefore, when $\epsilon_1 = 0, \epsilon_2 = 1$, $c'(s, s') = c(s, s') = -\log \frac{d_s^E(s)}{d_s^I(s)}$, the objective of PW-DICE and SMODICE are negated version of each other (with one maximizing and the other minimizing), and the constraints on d_{sa}^π are identical. Since Π is also solvable (one apparent solution is $\Pi = d_s^\pi \otimes d_s^E$), the two objectives are identical, and thus the objectives in Eq. (25) are equivalent. Both the Claim and the Corollary are proved, since we do not specify D_f. □

However, this claim is unintuitive in its dual form: as we always have $\epsilon_1 > 0, \epsilon_2 > 0$ in the dual form, the behavior of $\lim_{\epsilon_1 \to 0} \epsilon_1 \log \mathbb{E}_{s_i \sim I, s_j \sim E} \exp\left(\frac{\lambda_{i+|S|} + \lambda_{j+2|S|} - c(s_i, s_j)}{\epsilon_1}\right)$ in Eq. (3) is non-trivial. Thus, here we give another proof for Claim 1 directly from the dual perspective for KL-divergence as D_f in the continuous space:

Proof. (Dual Perspective, KL-divergence, continuous space) First, we prove by contradiction that

$$\lim_{\epsilon_1 \to 0} \epsilon_1 \log \mathbb{E}_{s\sim I, s'\sim E} \exp\left(\frac{\lambda_{s+|S|} + \lambda_{s'+2|S|} - c(s, s')}{\epsilon_1}\right) \tag{27}$$

is not the $\max$ operator, because at the optimum we have $\lambda_{s+|S|} + \lambda_{s'+2|S|} - c(s, s')$ to be equal for every $d_s^I(s) > 0, d_s^E(s') > 0$. Otherwise, assume the state pair (s, s') has the largest $\lambda_{s+|S|} + \lambda_{s'+2|S|} - c(s_0, s_0')$; because ϵ_1 can be arbitrarily close to 0, there exists ϵ_1 small enough such that there exists $s \neq s_0$ or $s' \neq s_0'$ that makes the infinitesimal increment of λ_s or λ_s' worthy (i.e., partial derivative with respect to λ_s or λ_s' greater than 0).

Then, we have

$$\begin{aligned} &\lim_{\epsilon_1 \to 0} \epsilon_1 \log \mathbb{E}_{s\sim I, s'\sim E} \exp\left(\frac{\lambda_{s+|S|} + \lambda_{s'+2|S|} - c(s, s')}{\epsilon_1}\right) \\ =&\mathbb{E}_{s\sim I, s'\sim E}\left(\lambda_{s+|S|} + \lambda_{s'+2|S|} - c(s, s')\right) \\ =&\mathbb{E}_{s\sim I}\left[\lambda_{s+|S|} + \log \frac{d_s^E(s)}{d_s^I(s)}\right] + \mathbb{E}_{s'\sim E}\lambda_{s'+2|S|}. \end{aligned} \tag{28}$$

Note that $\lambda_{s'+2|S|}$ in Eq. (28) is cancelled out with the linear term $-\mathbb{E}_{s'\sim E}\lambda_{s'+2|S|}$ in the objective (see Eq. (3)) later, so the value of $\lambda_{s'+2|S|}$ does not matter anymore. That means, for any $\lambda_{s'+2|S|}$, there exists an optimal solution. Therefore, without loss of generality, we let $\lambda_{s'+2|S|} = 0$. The objective then becomes

$$\begin{aligned}&\epsilon_1 \log \mathbb{E}_{s\sim I} \exp\left(\frac{\lambda_{s+|S|} + \log \frac{d_s^E(s)}{d_s^I(s)}}{\epsilon_1}\right) + \\ &\epsilon_2 \log \mathbb{E}_{(s,a,s')\sim I} \exp\left(\frac{-\gamma\lambda_{s'} + \lambda_s - \lambda_{s+|S|}}{\epsilon_2}\right) - (1-\gamma)\mathbb{E}_{s\sim p_0}\lambda_s.\end{aligned} \tag{29}$$

Then, we can use the same trick on $\epsilon_1 \to 0$ and infer that $\lambda_{s+|S|} = -\log \frac{d_s^E(s)}{d_s^I(s)} + Q$, where Q is some constant. Then, we have our optimization objective to be

$$L(\lambda) = Q + \epsilon_2 \log \mathbb{E}_{(s,a,s')\sim I} \exp\left(\frac{-\gamma\lambda_{s'} + \lambda_s + \log \frac{d_s^E(s)}{d_s^I(s)} - Q}{\epsilon_2}\right) - (1-\gamma)\mathbb{E}_{s\sim p_0}\lambda_s. \tag{30}$$

Note that Q is cancelled out again, which means that the value of Q does not matter. Without loss of generality, we set $Q = 0$, and then we obtain the SMODICE objective with KL-divergence. □

D. Experimental Details

D.1. Tabular MDP

Experimental Settings. We adopt the tabular MDP experiment from LobsDICE (Kim et al., 2022b). For the tabular experiment, there are 20 states in the MDP and 4 actions for each state s; each action a leads to four uniformly chosen states s'_1, s'_2, s'_3, s'_4. The vector of probability distribution over the four following states is determined by the formula $(p(s'_1|s,a), p(s'_2|s,a)), p(s'_3|s,a), p(s'_4|s,a) = (1-\eta)X + \eta Y$, where $X \sim$ Categorical($\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}$), and $Y \sim$ Dirichlet$(1,1,1,1)$. $\eta \in [0,1]$ controls the randomness of the transition: $\eta = 0$ means deterministic, and $\eta = 1$ means highly stochastic. The agent always starts from state s_0, and can only get a reward of $+1$ by reaching a particular state s_x. x is chosen such that the optimal value function $V^*(s_0)$ is minimized. The discount factor γ is set to 0.95.

Dataset Settings. For each MDP, the expert dataset is generated using a deterministic optimal policy with infinite horizon, and the task-agnostic dataset is generated similarly but with a uniform policy. Note that we use a different expert policy from the softmax policy of LobsDICE, because we found the value function for each state to be quite close due to the high connectivity of the MDP. Thus, the "expert" softmax policy is actually near-uniform and severely suboptimal.

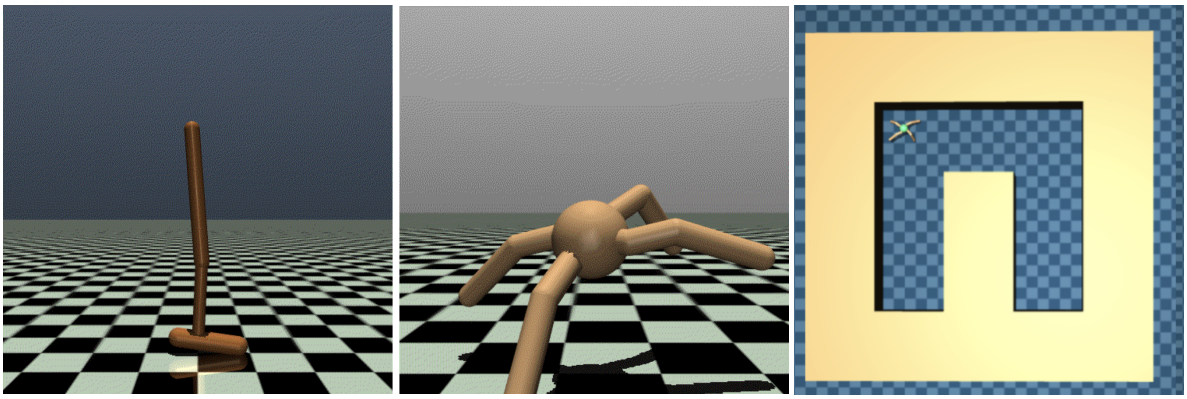
Selection of Hyperparameters. There is no hyperparameter selection for SMODICE. For LobsDICE, we follow the settings in their paper, which is $\alpha = 0.1$. For our method, we use $\epsilon_1 = \epsilon_2 = 0.01$ for the version with regularizer, and $\epsilon_1 = \epsilon_2 = 0$ for the version with Linear Programming (LP).

D.2. MuJoCo Environment

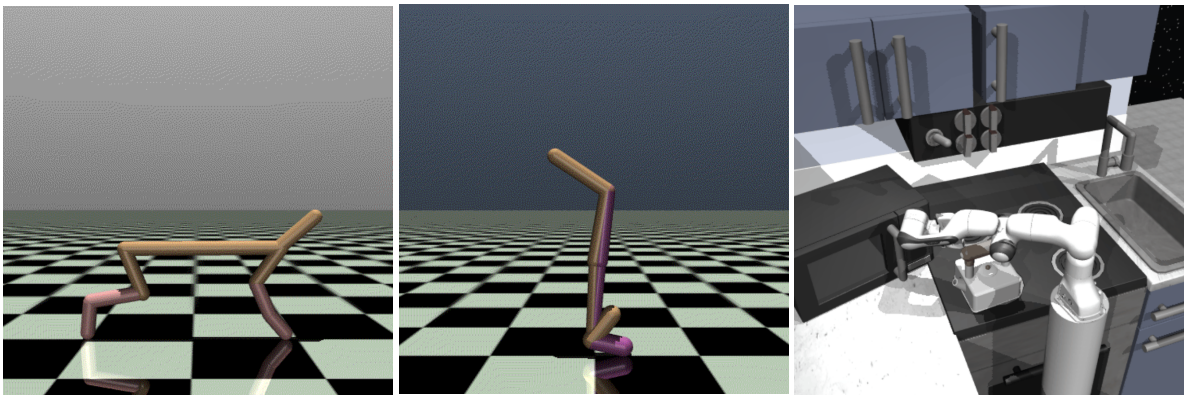
Experimental Settings. Following SMODICE (Ma et al., 2022), we test four widely adopted MuJoCo locomotion environments: hopper, halfcheetah, ant, and walker2d, and two more challenging locomotion environments, antmaze and kitchen. Below is the detailed description for each environment. See Fig. 10 for an illustration.

1. **Hopper.** Hopper is a 2D environment where the agent controls a single-legged robot to jump forward. The state is 11-dimensional, which includes the angle and velocity for each joint of the robot; the action is 3-dimensional, each of which controls the torque applied on a particular joint.

2. **Halfcheetah**. In Halfcheetah, the agent controls a cheetah-like robot to run forward. Similar to Hopper, the environment is also 2D, with 17-dimensional state space describing the coordinate and velocity and 6-dimensional action space controlling torques on its joints.



(a) Hopper (c) Ant (e) Antmaze



(b) Halfcheetah (d) Walker2d (f) Kitchen

Figure 10. Illustration of environments tested in Sec. 4.2 based on OpenAI Gym (Brockman et al., 2016) and D4RL (Fu et al., 2020b).

3. **Ant.** Ant is a 3D environment where the agent controls a quadrupedal robotic ant to move forward. The 111-dimensional state space includes the coordinate and velocity of each joint. The action space is 8-dimensional.

4. **Walker2d.** Walker2d, as its name suggests, is a 2D environment where the agent controls a two-legged robot to walk forward. The state space is 27-dimensional and the action space is 8-dimensional.

5. **Antmaze**. In this work, we consider the U-maze task, where the agent needs to manipulate a 8-DoF robotic ant with 29-dimensional state to crawl from one end of the maze to another, as illustrated in Fig. 10.

6. **Kitchen.** Franka Kitchen in D4RL is a challenging environment, where the agent manipulates a 9-DoF robotic arm and tries to complete 4 sequential subtasks. Subtask candidates include moving the kettle, opening the microwave, turning on the bottom or top burner, opening the left or right cabinet, and turning on the lights. The state space describes the status of the robot and the goal and current location of the target items. The state is 60-dimensional.

Dataset Settings. We adopt the same settings as SMODICE (Ma et al., 2022). SMODICE uses a single trajectory (1000 states) from the "expert-v2" dataset in D4RL (Fu et al., 2020b) as the expert dataset E. For the task-agnostic dataset I, SMODICE uses the concatenation of 200 trajectories ($200K$ state-action pairs) from "expert-v2" and the whole "random-v2" dataset ($1M$ state-action pairs).

Selection of Hyperparameters. Tab. 1 summarizes our hyperparameters, which are also the hyperpameters of plain Behavior Cloning if applicable. For baselines (SMODICE, LobsDICE, ORIL, OTR, and DWBC), we use the hyperparameters reported in their paper (unless the hyperparameter values in the paper and the code differ, in which case we report the values from the code).

Type	Hyperparameter	Value	Note
Disc.	Network Size	[256, 256]	
	Activation Function	Tanh	
	Learning Rate	0.0003	
	Training Length	40K steps	
	Batch Size	512	
	Optimizer	Adam	
Actor	Network Size	[256, 256]	
	Activation Function	ReLU	
	Learning Rate	0.001	
	Weight Decay	10^{-5}	
	Training length	1M steps	
	Batch Size	1024	
	Optimizer	Adam	
	Tanh-Squashed	Yes	
Critic	Network Size	[256, 256]	
	Activation Function	ReLU	
	Learning Rate	0.0003	
	Training Length	1M steps	
	Batch Size	1024	
	Optimizer	Adam	
	ϵ_1	kitchen 0.01, others 0.5	coefficient for the KL regularizer
	ϵ_2	kitchen 2, others 0.5	coefficient for the KL regularizer
	α	0.01	mixing coefficient to the denominator of $R(s)$
	β	5	coefficient for combination of distance metric
	γ	0.998	discount factor in our formulation

Table 1. Our selection of hyperparameters. We use the same network architecture and optimizer as SMODICE (Ma et al., 2022).

Implementation of Contrastively Learned Distance. We build our contrastive learning module on the implementation of FIST (Hakhamaneshi et al., 2022). The contrastive learning embedding framework is trained for 200 epochs over the task-agnostic dataset. In each epoch, we use a batch size of 4096, where each data point consists of a consecutive state pair (s_i, s_i'), $i \in \{1, 2, \ldots, 4096\}$, and the states are n-dimensional. An encoding layer $g(s) : \mathbb{R}^n \to \mathbb{R}^{M\times 1}$ with $M = 32$ embedding dimensions is first applied on each s_i and s_i'. Then, the score matrix $L \in \mathbb{R}^{4096\times 4096}$ is calculated, such that $L_{i,j} = g(s_i)^T W g(s_j')$. $W \in \mathbb{R}^{M\times M} = \text{softplus}(W_0)\text{softplus}(W_0^T)$ is a learnable matrix, and it is structured for better stability. Finally, each row of L is viewed as the score for a 4096-way classification. The label is the identity matrix. Such training paradigm gives us the loss also provided in Eq. (7):

$$L_c = \log \frac{\exp\left(q^T W k_+\right)}{\exp\left(q^T W k_+\right) + \sum_{k_-} \exp\left(q^T W k_-\right)}. \tag{31}$$

Here, $q = g(s_i)$ is the query (anchor), W is the weight matrix, $k_+ = g(s_i')$ is a positive key, and $k_- \in \{g(s_j')|j \neq i\}$ are negative keys. This objective essentially amounts to a 4096-way classification task, where for the i-th sample the correct label is i.

E. Additional Experimental Results

E.1. Supplementary Results for Tabular Environment

E.1.1. State and State-pair Total Variation (TV) distance

In this section, we show the Total Variation (TV) divergence between the state occupancies of the learner and the expert and the state-pair occupancies between the learner and the expert, i.e., $\text{TV}(d_s^\pi \| d_s^E)$ and $\text{TV}(d_{ss}^\pi \| d_{ss}^E)$. Fig. 11 shows the result of the state occupancy distance between the learner's and the expert policies. Fig. 12 shows the distance between the state-pair occupancies. We observe our method to work better than SMODICE and LobsDICE.

E.1.2. Tabular Experiment with Softmax Expert

To be consistent with LobsDICE (Kim et al., 2022b), we also report experimental results using exactly the same setting of LobsDICE, which uses a highly sub-optimal expert. Fig. 13, Fig. 14, and Fig. 15 show the regret, state occupancy divergence $\text{TV}(d_s^\pi \| d_s^E)$, and state-pair occupancy divergence $\text{TV}(d_{ss}^\pi \| d_{ss}^E)$ of each method in this setting respectively. The result shows that our method does not perform well in minimizing occupancy divergence, as the coefficient of the f-divergence regularizer in our PW-DICE is much smaller or 0, which means that our obtained policy is more deterministic and thus different from the highly stochastic "expert" policy. It is worth noting that our method, with accurate estimation of MDP

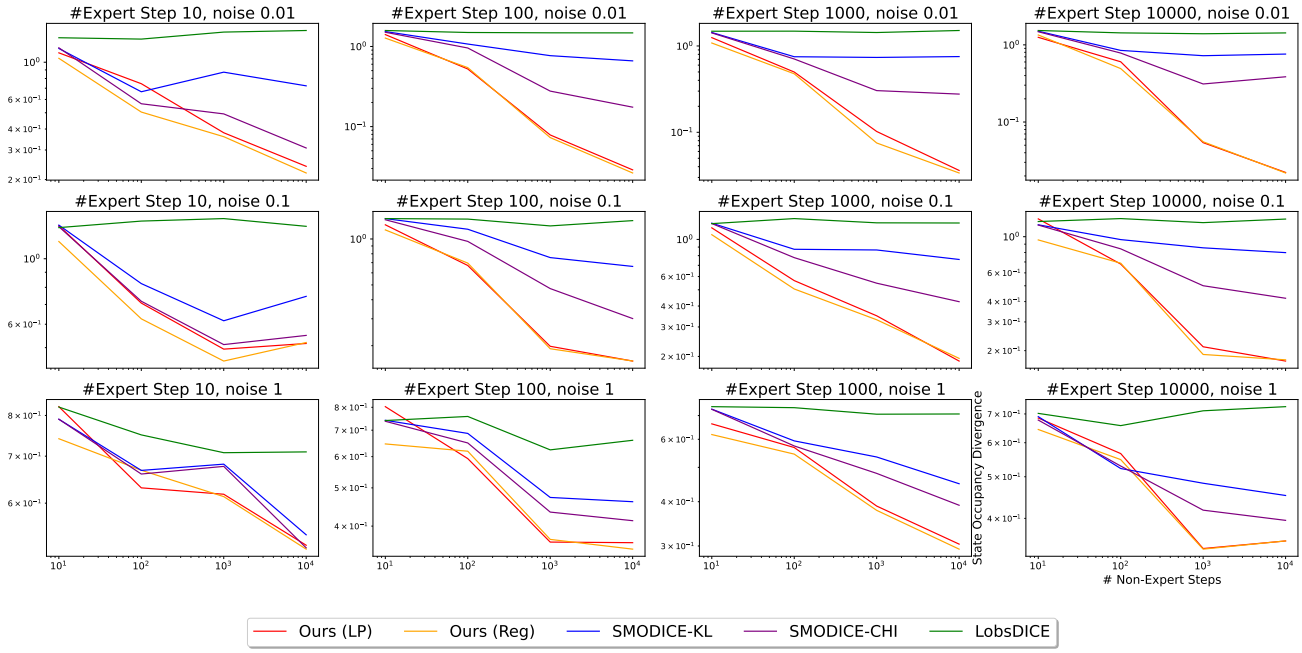


Figure 11. TV distance $\mathrm{TV}(d_s^\pi \| d_s^E)$ of each method on tabular environments. Our method, both with and without regularizer, works comparably well as the baselines for a small task-agnostic dataset, and prevails with larger task-agnostic dataset (more accurate estimated dynamics).

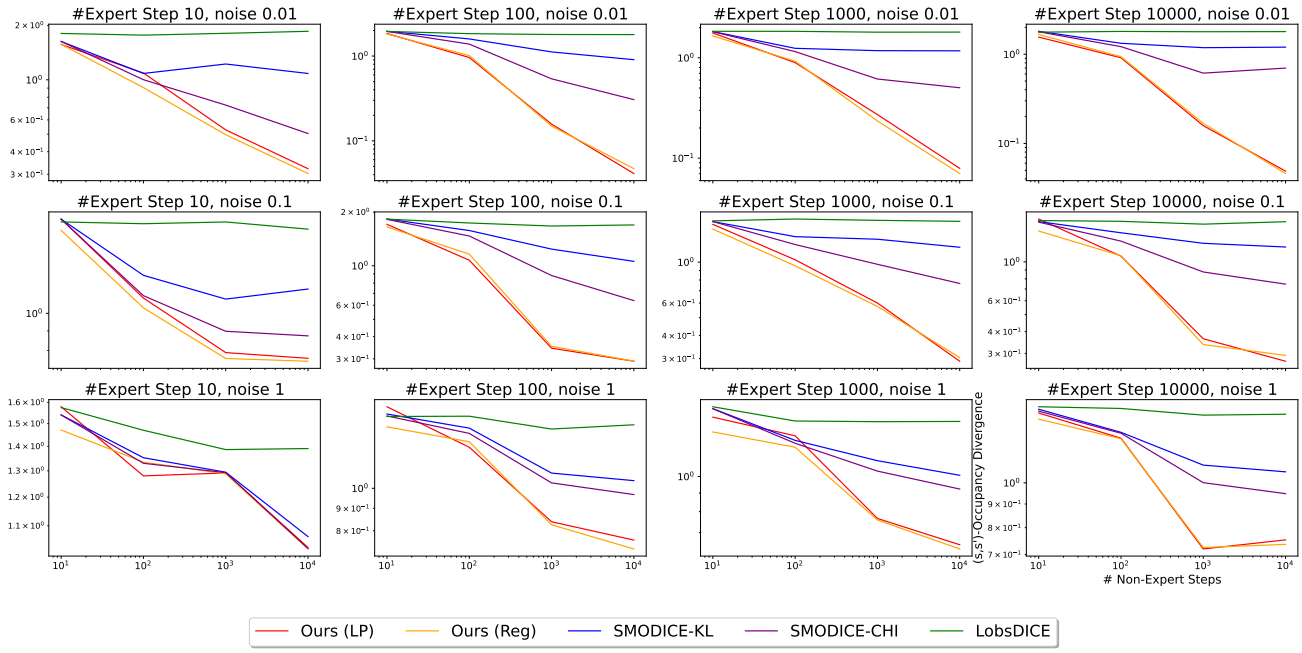


Figure 12. State-pair occupancy TV distance between the learner and expert ($\mathrm{TV}(d_{ss}^\pi \| d_{ss}^E)$) on tabular environments. Similar to TV distance between state-occupancies and regret, our method works the best, especially with larger task-agnostic dataset.

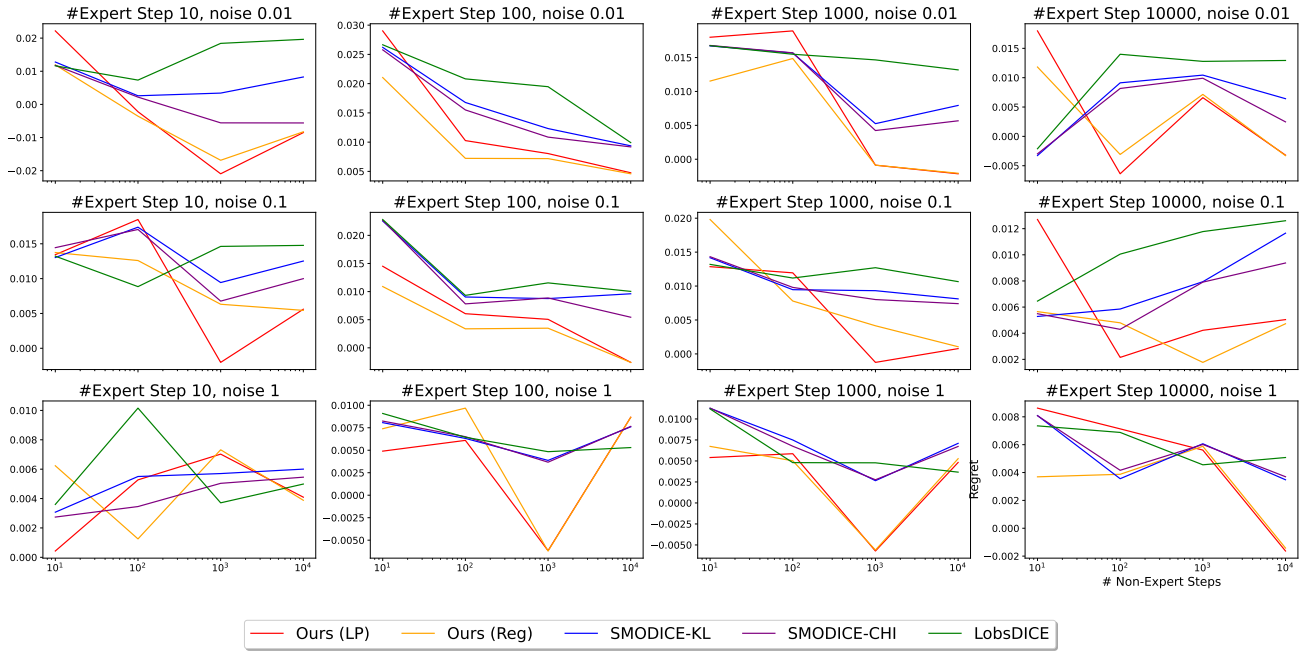


Figure 13. Regret of each method for tabular experiments with softmax expert. Our method with regularizer generally achieves the lower regret. Also, our method is the only one that achieves negative regret (i.e., better than the highly suboptimal "expert").

dynamics (i.e., large size of the task-agnostic/non-expert dataset), is the only method that achieves negative regret, i.e., our method is even better than the "expert" policy. Also, our method with regularizer generally achieves lower regret.

E.2. PW-DICE with χ^2-divergence on MuJoCo Environment

In the main paper, we mainly considered PW-DICE with KL-divergence. However, as Corollary C.2 suggests, the D_f regularizer in PW-DICE can also be χ^2-divergence. Suppose we use half χ^2-divergence as SMODICE (Ma et al., 2022) does, i.e., $f(x) = \frac{1}{2}(x-1)^2$, $f_*(x) = \frac{1}{2}(x+1)^2$, and $f'(x) = x + 1$. With such a divergence, the final optimization objective of PW-DICE reads as follows:

$$\begin{aligned}&\min_{\lambda} \frac{\epsilon_1}{2}\mathbb{E}_{s_i\sim I, s_j\sim E}\left(\frac{\lambda_{i+|S|} + \lambda_{j+2|S|} - c(s_i, s_j)}{\epsilon_1} + 1\right)^2 \\ &+\frac{\epsilon_2}{2}\mathbb{E}_{(s_i,a_j,s_k)\sim I}\left(\frac{-\gamma\lambda_k + \lambda_i - \lambda_{i+|S|}}{\epsilon_2} + 1\right)^2 - \left[(1-\gamma)\mathbb{E}_{s\sim p_0}\lambda_{:|S|} + \mathbb{E}_{s\sim E}\lambda_{2|S|:3|S|}\right],\end{aligned} \tag{32}$$

and the policy loss is

$$E_{(s,a)\sim I}\max\left(0, \frac{-\gamma\mathbb{E}_{s_k\sim p(\cdot|s_i,a_j)}\lambda_k + \lambda_i - \lambda_{i+|S|}}{\epsilon_2}\right). \tag{33}$$

However, similar to SMODICE, we found that the χ^2-divergence regularizer does not work well under MuJoCo environments, as the weight ratio between good and bad actions in the task-agnostic dataset is only proportional (instead of exponential) to $-\gamma\lambda_k + \lambda_i - \lambda_{i+|S|}$, and thus is not discriminative enough. As a result, the retrieved policy is highly stochastic. Fig. 16 shows the result of χ^2-divergence, which is much worse than the KL-divergence result.

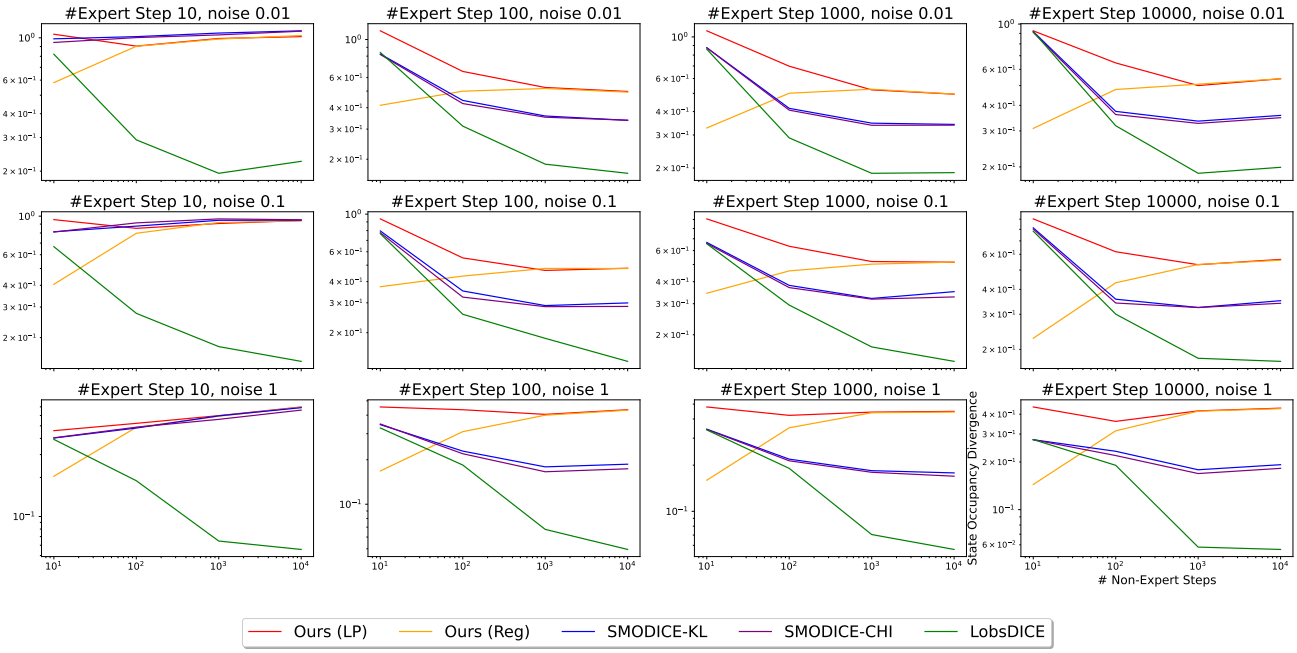


Figure 14. State occupancy TV distance $\mathrm{TV}(d_s^\pi \| d_s^E)$ of each method for tabular experiments with softmax expert. Our method does not work well because the expert policy is highly stochastic.

E.3. Learning from Expert with Mismatched Dynamics

In order to show that our method is robust with respect to different dynamics, we adopt the mismatched dynamics setting from SMODICE (Ma et al., 2022), where the agent needs to learn from the same task-agnostic dataset as that in the main results of Sec. 4.2, but with the expert dataset generated by an expert with very different dynamics; for example, one of the legs of the expert is amputated in the ant environment, and the torso of the expert is much shorter in the halfcheetah environment. We use exactly the same setting as SMODICE; Fig. 17 shows the result, which illustrates that our method is generally more robust to embodiment differences than SMODICE, LobsDICE, and ORIL.

F. Notations Table

Tab. 2 lists the symbols that appear in the paper.

G. Computational Resources

All experiments are carried out with a single NVIDIA RTX 2080Ti GPU on an Ubuntu 18.04 server with 72 Intel Xeon Gold 6254 CPUs @ 3.10GHz. Given these resources, our method needs about $5 - 5.5$ hours to finish training in the MuJoCo environments (during which the training of the distance metric, including $R(s)$ and contrastive learning, takes $20 - 40$ minutes), while ORIL, SMODICE, and LobsDICE require about $2.5 - 3$ hours. As the actor network is identical across all methods, the inference speed is similar and is not a bottleneck.

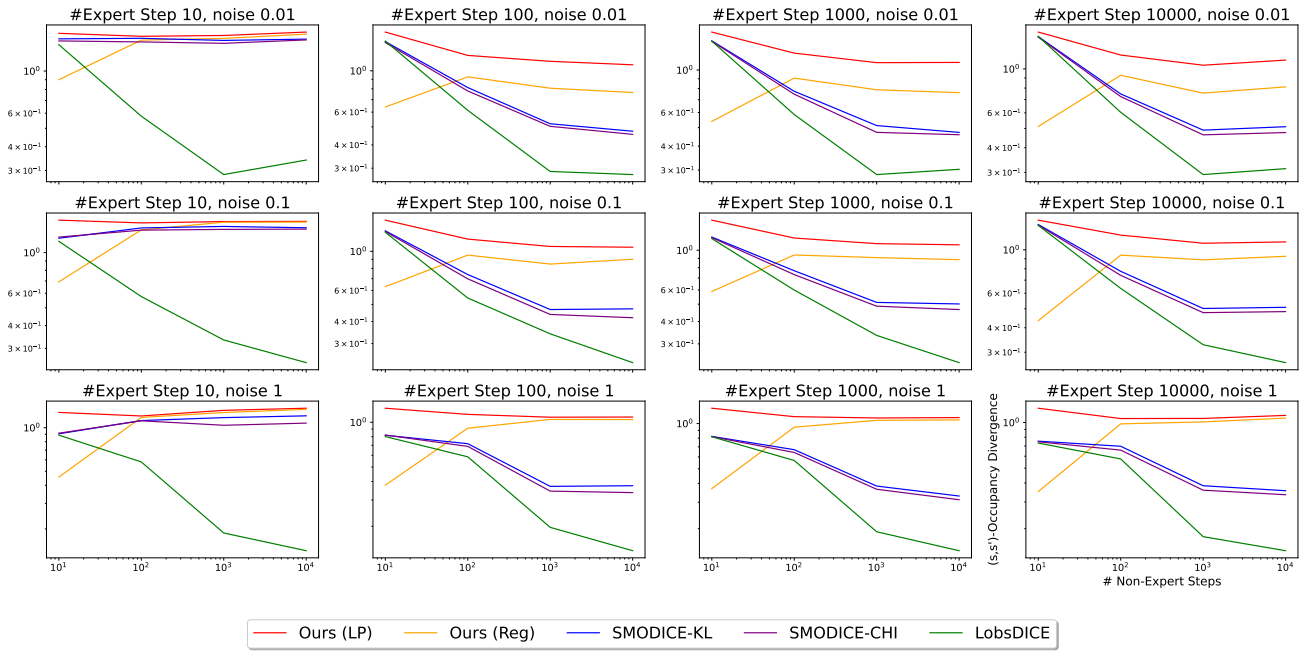


Figure 15. State-pair occupancy TV distance $\mathrm{TV}(d^{\pi}_{ss} \| d^{E}_{ss})$ of each method for tabular experiments with softmax expert. Our method does not work well because the expert policy is highly stochastic.

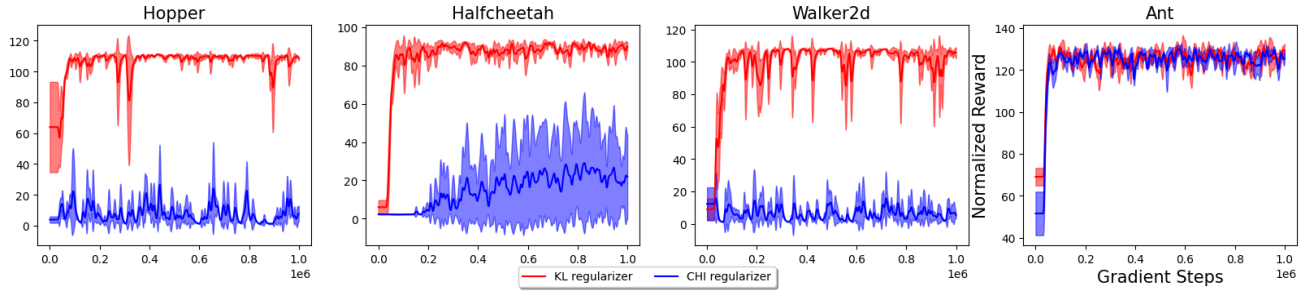


Figure 16. Performance comparison between χ^2-divergence (blue) and KL-divergence (red) in PW-DICE. χ^2-divergence does not work as well as KL-divergence.

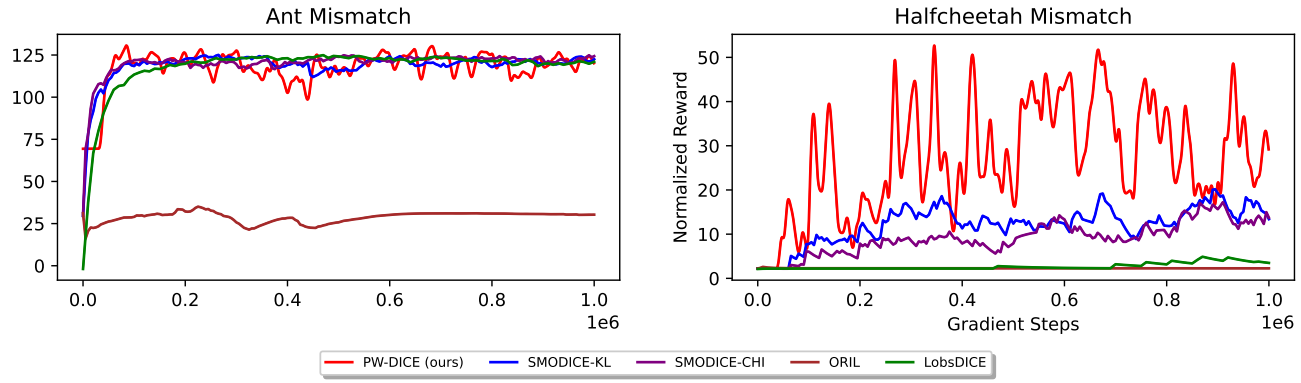


Figure 17. Performance comparison on expert with mismatched dynamics; it is shown that our method is generally more robust than SMODICE, LobsDICE, and ORIL on the two environments.

Name	Meaning	Note
S	State space	$\|S\|$ is the size of state space
A	Action space	$\|A\|$ is the size of state space
γ	Discount factor	$\gamma \in (0, 1)$
r	Reward function	$r(s, a)$ for single state-action pair
T	Transition function	
p	Transition (single entry)	$p(s'\|s, a) \in \Delta(S)$
p_0	Initial distribution	$p_0 \in \Delta(S)$
s	State	$s \in S$
a	Action	$a \in A$
$\bar{s}$	Past state	
$\bar{a}$	Past action	
τ	Trajectories	State-only or state-action; depend on context
E	Expert dataset	state-only expert trajectories
I	Task-agnostic dataset	state-action trajectories of unknown optimality
π	Learner policy	
π^E	Expert policy abstracted from E	
π^I	Task-agnostic policy abstracted from I	
d^π_{sa}	State-action occupancy of π	
d^π_s	State occupancy of π	1) $\forall s \in \mathcal{S}, \sum_a d^\pi_{sa}(s, a) = d^\pi_s(s)$. This equation also applies similarly between d^E_{sa} and d^E_s, as well as d^I_{sa} and d^I_s. 2) $d^\pi_s(s) = (1-\gamma)\sum_{i=0}^{\infty} \gamma^i \Pr(s_i = s)$, where s_i is the i-th state in a trajectory. This holds similarly for $d^I(s)$ and $d^E(s)$. 3) $d^\pi_{sa}(s, a) = d^\pi_s(s)\pi(a\|s)$. This holds similarly for d^E_{sa}, π_E and d^I_{sa}, π_I.
d^π_{ss}	State-pair occupancy of π	
d^E_s	State occupancy of π^E	
d^E_{ss}	State-pair occupancy of π^E	
d^I_{sa}	State-action occupancy of π^I	
d^I_s	State occupancy of π^I	
λ	Dual variable	
D_f	f-divergence	
$f_*(\cdot)$	Fenchel conjugate of f	
c	Matching cost for Wasserstein distance	
c'	Matching cost for Wasserstein distance	With extended domain
Π	Wasserstein matching variable	$\sum_{s\in S} \Pi(s, s') = d^E_s(s')$, $\sum_{s'\in S} \Pi(s, s') = d^\pi_s(s)$
A	Equality constraint matrix	
x	unified self-variable	concatenation of flattened Π and d^π_{sa} (row first)
b	Equality constraint vector	$Ax = b$
U	Distribution as regularizer	product of d^I_s and d^E_s
$\mathcal{W}$	Wasserstein distance	
$h(\cdot)$	state discriminator	
W	Weight matrix of contrastive learning	$W \in \mathbb{R}^{32\times 32}$
$g(\cdot)$	embedding to be learned	
L	score matrix	$L \in \mathbb{R}^{4096\times 4096}$
n	number of dimensions for state	
M	number of dimensions for embedding	$M = 32$
β	coefficient for learned embedding in distance metric	

Table 2. Complete list of notations used in the paper. The first part is for offline LfO settings, the second part lists notations specific to PW-DICE, and the third part is for notations used in contrastive learning (Appendix D.2).