# OFFLINE REINFORCEMENT LEARNING WITH PSEUDOMETRIC LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Offline Reinforcement Learning methods seek to learn a policy from logged transitions of an environment, without any interaction. In the presence of function approximation, and under the assumption of limited coverage of the state-action space of the environment, it is necessary to enforce the policy to visit state-action pairs *close* to the support of logged transitions. In this work, we propose an iterative procedure to learn a pseudometric from logged transitions, and use it to define this notion of closeness. We show its convergence and extend it to the function approximation setting. We then use this pseudometric to define a new lookup based bonus in an actor-critic algorithm: PLOFF. This bonus encourages the actor to stay close, in terms of the defined pseudometric, to the support of logged transitions. Finally, we evaluate the method on hand manipulation and locomotion tasks.

## 1 INTRODUCTION

Reinforcement Learning (RL) has proven its ability to solve complex problems in recent years (Silver et al., 2016; Vinyals et al., 2019). Behind those breakthroughs, the adoption of RL in real-world systems remains challenging (Dulac-Arnold et al., 2019). Learning a policy by trial-and-error, while operating on the system, can be detrimental to the system where it is deployed (*e.g.*, user satisfaction in recommendation systems or material damage in robotics) and is not guaranteed to lead to good performance (sparse rewards problems).

Nevertheless, in the setting where experiences were previously collected in an environment (*logged transitions*), one possible way of learning a policy is to mimic the policy that generated these experiences (Pomerleau, 1991). However, if these experiences come from different sources, with different degrees of desirability, naive imitation might lead to poor results. On the other hand, offline RL (or batch RL) (Lagoudakis & Parr, 2003; Ernst et al., 2005; Riedmiller, 2005; Levine et al., 2020), offers a setting where the policy is learned from the collected experiences. As it requires no interaction with the environment, offline RL is a promising direction for learning policies that can be deployed into systems.

Still, collected experiences typically only cover a subset of the range of possibilities in an environment (not every state is present; for a given state, not every action is taken). With function approximation, this is particularly problematic since actions not executed in the system can be assigned overly optimistic values (especially through bootstrapping), which leads to poor policies. To limit this extrapolation error, offline RL is typically incentivized to learn policies that are plausible in light of the collected experiences. In other words, offline RL methods need to learn policies that maximize their return, while making sure to remain *close* to the support of logged transitions.

This work introduces a new method for computing the closeness to the support by learning a pseudometric from collected experiences. This pseudometric, close to bisimulation metrics (Ferns et al., 2004), computes similarity between state-action pairs based on the expected difference in rewards when following specific sequences of action. We show theoretical properties in the dynamic programming setting for deterministic environments as well as for the sampled setting. We further extend the learning of this pseudometric to the function approximation setting, and propose an architecture to learn it from collected experience. We define a new offline RL actor-critic algorithm: PLOFF (**P**seudometric **L**earning **Off**line RL), which computes a bonus through this learned pseudometric and uses it to filter admissible actions in the greedy step. Finally, we lead an empirical study on the hand manipulation and locomotion tasks of the offline RL benchmark from Fu et al. (2020).

We make the following contributions: 1) we extend neural bisimulation metrics (Castro, 2020) to state-action spaces and to the offline RL setting, 2) we exploit this pseudometric to tackle the out-of-distribution extrapolation error of offline RL by adding a simple lookup bonus to a standard actor-critic algorithm and show that it compares favorably to state-of-the art offline RL methods.

## 2 BACKGROUND

**Reinforcement Learning.** We consider the classic RL setting (Sutton & Barto, 1998), formalized with Markov Decision Processes (MDPs). An MDP is a tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, r, P, \gamma)$, with $\mathcal{S}$ the state space, $\mathcal{A}$ the action space, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ the expected reward function, $P : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{P}(\mathcal{S})$ the transition function which maps state-action pairs to distributions over the set of states $\mathcal{P}(\mathcal{S})$ and $\gamma$ the discount factor for which we assume $\gamma \in [0, 1)$. A stationary deterministic policy $\pi$ is a mapping from states to actions (the following can easily be extended to stochastic policies). The value function $V^\pi$ of a policy $\pi$ is defined as the expected discounted cumulative reward from starting in a particular state and acting according to $\pi$: $V^\pi(s) = \mathbb{E}\big(\sum_{i=0}^{\infty} \gamma^i r(s_i, \pi(s_i)) | s_0 = s\big)$. The action-value function $Q^\pi$ is defined as the expected cumulative reward from starting in a particular state, taking an action and then acting according to $\pi$: $Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}\big(V^\pi(s')\big)$. The Bellman (1957) operator $\mathcal{B}$ connects an action-value function $Q$ for state-action couple $(s, a)$ to the action-value function in the subsequent state: $\mathcal{B}^\pi(Q)(s, a) := r(s, a) + \gamma \mathbb{E}\big(Q(s', \pi(s'))\big)$. $Q^\pi$ is the (unique) fixed-point of this operator, and the difference between $Q(s, a)$ and its image through Bellman $\|Q - \mathcal{B}^\pi Q\|$ is called a temporal difference error.

An optimal policy $\pi$ maximizes the value function $V^\pi$ for all states. In continuous state-action spaces, actor-critic methods (Konda & Tsitsiklis, 2000) are a common paradigm to learn the optimal policy. In this work we only consider deterministic policies; we justify this restriction by the fact that stochastic policies are desirable because of their side effect of exploration (Haarnoja et al., 2018), but in this case we want to learn a policy with near-optimal behavior without interaction with the environment. Therefore, we use the actor-critic framework of Silver et al. (2014). It consists in concurrently learning a parametrized policy $\pi_\theta$ and its associated parametrized action-value function $Q_\omega$. $Q_\omega$ minimizes a temporal difference error $\|Q_\omega(s, a) - r(s, a) - Q_{\bar{\omega}}(s', \pi_\theta(s'))\|$ (with $Q_{\bar{\omega}}$ a target action-value function, tracking $Q_w$), and $\pi_\theta$ maximizes the action-value function $Q_\omega(s, \pi_\theta(s))$.

In the classical RL setting, transitions $(s, a, s', r)$ are sampled through interactions with the environment. In on-policy actor-critic methods (Sutton et al., 1999; Schulman et al., 2015), updates on $\pi_\theta$ and $Q_\omega$ are made as the policy gathers transitions by interacting in the environment. In off-policy actor-critic methods (Lillicrap et al., 2016; Haarnoja et al., 2018; Fujimoto et al., 2019), the transitions gathered by the policy are stored in a replay buffer and sampled using different sampling strategies (Lin, 1992; Schaul et al., 2015). These off-policy methods extend to the offline RL setting quite naturally. The difference is that in the offline RL setting, transitions are not sampled through interactions from the environment, but from a fixed dataset of transitions.

Throughout the paper we refer to $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{1:N}$ as the dataset of $N$ transitions collected in the considered environment. To ease notations we write $s \sim \mathcal{D}, s, a \sim \mathcal{D}, r \sim \mathcal{D}$ to indicate that a transition $(s, a, s', r)$ is sampled at random from this dataset, and that we only consider the associated state $s$, state-action pair $(s, a)$ or reward $r$.

**Pseudometric in MDPs.** A core issue in RL is to define a meaningful metric between states or state-action pairs. Consider for example a maze, two states could be close according to the Euclidean distance, but far away in terms of the minimal distance an agent would have to travel to join one state from the other (due to walls). In this case, a relevant metric is the distance in the graph formed from state transitions induced by the MDP (Mahadevan & Maggioni, 2007). We consider the following relaxed notion of metric[1]:

**Definition 1** (Pseudometric)**.** *Given a set $M$, it is a function $d : M \times M \mapsto \mathbb{R}_+$ such that, $\forall x, y, z \in M$, we have $d(x; x) = 0$, $d(x; y) = d(y; x)$, $d(x; z) \leq d(x; y) + d(y; z)$.*

In the context of Markov Decision Processes, bisimulation relations (Givan et al., 2003) are a form of state abstraction (Li et al., 2006), based on an equivalence relation. They are defined by the following

[1] A metric is a pseudometric for which $d(x, y) = 0 \Rightarrow x = y$.

recurrent definition: two states are bisimilar if they yield the same expected reward and transition to bisimulation equivalence classes with equal probability. This definition is too restrictive to be useful in practice. Ferns et al. (2004) introduce bisimulation metrics which are pseudometrics that soften the concept of bisimulation relations. Bisimulation metrics are defined on the state space $\mathcal{S}$. Denote $\mathbb{M}_{\mathcal{S}}$ the set of pseudometrics on $\mathcal{S}$, the bisimulation metric is the (unique) fixed point of the operator $\mathcal{F}_{\mathcal{S}}$ defined as:

$$\mathcal{F}_{\mathcal{S}}(d)(s;t) := \max_{a \in \mathcal{A}} \Big( |r(s,a) - r(t,a)| + \gamma \mathcal{W}_1(d)(P(s,a), P(t,a)) \Big)$$

where $\mathcal{W}_1(d)$ is the 1-Wasserstein distance (Villani, 2008) with the distance between states measured according to the pseudometric $d$. Therefore, the bisimulation metric is the limit of the repeated application of the sequence $\big({\mathcal{F}_{\mathcal{S}}}^n(d_0)\big)_{n \in \mathbb{N}}$ for any initial $d_0 \in \mathbb{M}_{\mathcal{S}}$.

Although pseudometrics in MDPs have proven to be effective in some applications (Melo & Lopes, 2010; Kim & Park, 2018; Dadashi et al., 2021), they are usually hand-crafted or learned with ad-hoc strategies.

# 3 METHOD

We present the overall idea of our method in Sec. 3.1: an offline RL algorithm which is incentivized to remain close to the support of collected experiences using a pseudometric-based bonus. In Sec. 3.2 to 3.4, we present how to learn this pseudometric, from a theoretical motivation to a gradient-based method. We give practical considerations to derive the bonus in Sec. 3.5 and provide the resulting algorithm: PLOFF, in Sec. 3.6.

## 3.1 OFFLINE RL WITH LOOKUP BONUS

For the time being, let us assume the existence of a pseudometric $d$ on the state-action space $\mathcal{S} \times \mathcal{A}$. We can infer a distance $d_{\mathcal{D}}$ from a transition to the dataset $\mathcal{D}$:

$$d_{\mathcal{D}}(s,a) = \min_{\hat{s},\hat{a} \in \mathcal{D}} d(s,a;\hat{s},\hat{a}).$$

This distance to the dataset, also referred to as the projection distance, is simply the distance from $(s,a)$ to the nearest element of $\mathcal{D}$. It is central to our work since it defines the notion of closeness to the support of transitions $\mathcal{D}$. From $d_{\mathcal{D}}$, we can infer a bonus $b$ using a monotonously decreasing function $f : \mathbb{R} \mapsto \mathbb{R}$:

$$b(s,a) = f(d_{\mathcal{D}}(s,a)).$$

Note that the concept of a bonus is overloaded in RL; it typically applies to exploration strategies (Schmidhuber, 1991; Thrun, 1992; Bellemare et al., 2016). In our case, the bonus $b$ is opposite to exploration-based bonuses since it will encourage the policy to act similarly to existing transitions of the collected experiences $\mathcal{D}$. In other words, it prevents exploring too far from the dataset.

We adapt the actor-critic framework by adding the bonus to the actor maximization step. We learn a parametrized policy $\pi_\theta$ and its corresponding action-value function $Q_\omega$. In a schematic way, we sample transitions $(s,a,r,s') \in \mathcal{D}$ and minimize the two following losses:

$$\begin{aligned} &\text{(critic)} \quad \min_{\omega} \|Q_\omega(s,a) - r(s,a) - \gamma Q_{\bar{\omega}}(s', \pi_\theta(s'))\|, \\ &\text{(actor)} \quad \max_{\theta} Q_\omega(s, \pi_\theta(s)) + b(s, \pi_\theta(s)). \end{aligned} \tag{1}$$

This modification of the actor-critic framework is common in offline RL (Buckman et al., 2021). The bonus $b$ typically consists in a measure of similarity between an estimated behavior policy that generated $\mathcal{D}$ and the policy $\pi$ we learn (Fujimoto et al., 2018; Wu et al., 2019; Kumar et al., 2019).

## 3.2 PSEUDOMETRIC LEARNING

To define a bonus $b$, we first learn a pseudometric $d$ on the state-action space $\mathcal{S} \times \mathcal{A}$ similarly to bisimulation metrics (Ferns et al., 2004; 2011; 2006), with the difference being that we are interested

in pseudometrics in state-action space $\mathcal{S} \times \mathcal{A}$ rather than state space $\mathcal{S}$. We will show that the pseudometric $d$ we are interested in is the fixed point of an operator $\mathcal{F}$. In the following, we assume that the MDP is deterministic.

Let $\mathbb{M}$ be the set of bounded pseudometrics on $\mathcal{S} \times \mathcal{A}$. We define the operator $\mathcal{F} : \mathbb{M} \mapsto \mathbb{M}$ as follows: for two state-action pairs $(s_1, a_1)$ and $(s_2, a_2)$, that maps to next states $s_1'$ and $s_2'$ we have:

$$\mathcal{F}(d)(s_1, a_1; s_2, a_2) = |r(s_1, a_1) - r(s_2, a_2)| + \gamma \mathbb{E}_{a' \sim \mathcal{U}(\mathcal{A})} d(s_1', a'; s_2', a'),$$

with $\mathcal{U}(\mathcal{A})$ the uniform distribution over actions.

This operator is of particular interest as it takes a distance $d$ over state-action pairs as input, and outputs a distance $\mathcal{F}(d)$ which is the distance between immediate rewards, to which is added the discounted expected distance between the two transitioning states for a random action. Notice that contrary to bisimulation metrics, we do not use a maximum over next actions for multiple reasons: the use of a maximum can be overly pessimistic when computing the similarity (Castro, 2020); in the case of continuous action spaces a maximum is hard to estimate; and finally in the presence of function approximation (Section 3.4) it can lead to instabilities.

We now establish a series of properties of the operator $\mathcal{F}$, all being proven in Appendix B.

**Proposition 3.1.** *Let $d$ be a pseudometric in $\mathbb{M}$, then $\mathcal{F}(d)$ is a pseudometric in $\mathbb{M}$.*

We are now interested in the repeated application of this operator $\left(\mathcal{F}^n(d_0)\right)_{n \in \mathbb{N}}$ starting from the 0-pseudometric $d_0$ (mapping all pairs to 0).

**Proposition 3.2.** *Let $d$ be a pseudometric in $\mathbb{M}$, we note $\|d\|_\infty$ as $\max_{s,s' \in \mathcal{S}} \max_{a,a' \in \mathcal{A}} d(s, a; s', a')$. The operator $\mathcal{F}$ is a $\gamma$-contraction for $\| \cdot \|_\infty$.*

Since the operator $\mathcal{F}$ is a contraction, it follows that the sequence $\left(\mathcal{F}^n(d_0)\right)_{n \in \mathbb{N}}$ converges to the pseudometric of interest $d^*$ (it would for any initial pseudometric, but $d_0$ is of particular empirical interest).

**Proposition 3.3.** *$\mathcal{F}$ has a unique fixed point $d^*$ in $\mathbb{M}$. Suppose $d_0 \in \mathbb{M}$ then $\lim_{n \to \infty} \mathcal{F}^n(d_0) = d^*$.*

The fixed point of $\mathcal{F}$ can be thought of as the similarity between state-actions, measured by the difference in immediate rewards added to the difference in rewards in future states if the sequence of actions is selected uniformly at random. In other words, two state-action couples will be close if 1) they yield the same immediate reward and 2) following a random walk from the resulting transiting states yields a similar return.

### 3.3 Pseudometric learning with sampling

Now, we move to the more realistic setting where the rewards and dynamics of the MDP are not known. Thus, the MDP is available through sampled transitions. We assume in this section that we have a finite state-action space. We define an operator $\hat{\mathcal{F}}$, which is a sampled version of $\mathcal{F}$: suppose we sample a pair of transitions from the environment $(\hat{s}_1, \hat{a}_1, \hat{s}_1', \hat{r}_1), (\hat{s}_2, \hat{a}_2, \hat{s}_2', \hat{r}_2)$, we have:

$$\hat{F}(d)(s_1, a_1; s_2, a_2) = \begin{cases} |\hat{r}_1 - \hat{r}_2| + \gamma \mathbb{E}_{u_1, u_2 \sim \mathcal{U}(\mathcal{A})} \, d(\hat{s}_1', u_1, \hat{s}_2', u_2) \text{if } s_1, a_1, s_2, a_2 = \hat{s}_1, \hat{a}_1, \hat{s}_2, \hat{a}_2, \\ d(s_1, a_1; s_2, a_2) \;\; \text{otherwise.} \end{cases}$$

Similarly to what is observed in the context of bisimulation metrics by Castro (2020), the sampled version $\hat{F}$ has similar convergence properties as $\mathcal{F}$.

**Proposition 3.4.** *Under common assumptions of state-action coverage, the repeated application of $\hat{\mathcal{F}}$ converges to the fixed point $d^*$ of $\mathcal{F}$.*

If the environment is stochastic, the repeated application of $\hat{\mathcal{F}}$ does not converge to the fixed point of the operator $\mathcal{F}$. In fact, it is not even stable in the space of pseudometrics (the expectation and absolute

value are not commutative). This results appears as a limitation of our work, since it only applies to deterministic environments. We leave to future work whether we can define a different pseudometric (fixed point of another operator) that would have convergence guarantees in the sampling case. As we only evaluate our approach on deterministic environments (Section 4), another interesting direction is whether our approach empirically extends to stochastic environments (even if it is less principled).

### 3.4 PSEUDOMETRIC LEARNING WITH APPROXIMATION

Building upon the insights of the previous sections, we derive an approximate version of the iterative scheme, to estimate a near-optimal pseudometric $d^*$. Pairs of transitions are assumed to be sampled from a fixed dataset $\mathcal{D}$. We use Siamese neural networks $\Phi$ (Bromley et al., 1994) to derive an approximate version of a pseudometric $d_\Phi$. Using Siamese networks to learn a pseudometric (Castro, 2020) is natural since it respects the actual definition of a pseudometric by design (Definition 1). To ease notations, we conflate the definition of the deep network $\Phi$ with its parameters. We define the pseudometric $d_\Phi$ as:

$$d_\Phi(s_1, a_1; s_2, a_2) = \|\Phi(s_1, a_1) - \Phi(s_2, a_2)\|,$$

where $\|\cdot\|$ is the Euclidean distance.

From the fixed-point iteration scheme defined in Section 3.3, we want to define a loss to retrieve the fixed point $d^*$. Similarly to fitted value-iteration methods (Bertsekas & Tsitsiklis, 1996; Munos & Szepesvari, 2008), which is the basis for the DQN algorithn (Mnih et al., 2015), we consider the parameters of the image of the operator $\hat{\mathcal{F}}$ to be fixed, and note it $\hat{\mathcal{F}}(d_{\bar{\Phi}})$. We thus learn $d_\Phi$ by minimizing the following loss, which is exactly the temporal difference error $\big(\hat{\mathcal{F}}(d_{\bar{\Phi}}) - d_\Phi\big)^2$:

$$\mathcal{L}_\Phi = \mathbb{E}_{\substack{s_1,a_1,r_1,s_1' \sim \mathcal{D} \\ s_2,a_2,r_2,s_2' \sim \mathcal{D}}} \Big( d_\Phi(s_1, a_1; s_2, a_2) - |r_1 - r_2| - \gamma \mathbb{E}_{a' \in \mathcal{U}(\mathcal{A})} d_{\bar{\Phi}}(s_1', a'; s_2', a') \Big)^2.$$

We introduce another pair of Siamese networks $\Psi$ (again we conflate the definition of the network with its parameters) to track the bootstrapped estimate $\mathbb{E}_{a' \in \mathcal{U}(\mathcal{A})} d_{\bar{\Phi}}(s_1, a', s_2, a')$, that we learn minimizing the following loss:

$$\mathcal{L}_\Psi = \mathbb{E}_{s_1 \sim \mathcal{D}, s_2 \sim \mathcal{D}} \Big( \|\Psi(s_1) - \Psi(s_2)\| - \mathbb{E}_{a' \in \mathcal{U}(\mathcal{A})} d_{\bar{\Phi}}(s_1, a'; s_2, a') \Big)^2.$$

We justify this design choice in Section 3.5, where we show that this makes the derivation of the bonus tractable. Therefore, we optimize the two following losses $\hat{\mathcal{L}}_\Psi$ and $\hat{\mathcal{L}}_\Phi$:

$$\hat{\mathcal{L}}_\Phi = \mathbb{E}_{\substack{s_1,a_1,r_1,s_1' \sim \mathcal{D} \\ s_2,a_2,r_2,s_2' \sim \mathcal{D}}} \Big( \|\Phi(s_1, a_1) - \Phi(s_2, a_2)\| - |r_1 - r_2| - \gamma |\Psi(s_1') - \Psi(s_2')| \Big)^2,$$

$$\hat{\mathcal{L}}_\Psi = \mathbb{E}_{s_1 \sim \mathcal{D}, s_2 \sim \mathcal{D}} \Big( \|\Psi(s_1) - \Psi(s_2)\| - \frac{1}{n} \sum_{\substack{u_1,\ldots,u_n \\ \sim \mathcal{U}(\mathcal{A})}} \|\Phi(s_1, u_1) - \Phi(s_2, u_2)\| \Big)^2.$$

A visual representation of the two pairs of Siamese networks as well as their losses is provided in the Appendix C. Remark that the proposed architecture seems to present similar caveats as naive offline RL approaches (since we estimate quantities that might not be present in the dataset of collected experiences). However, here, the divergence of the quantity at hand (the pseudometric learned) is unlikely since the goal is to minimize a positive quantity. This makes the problem of learning $d_\Phi$ inherently more stable than learning an optimal policy. A limitation of this method is that it relies on the reward function $r$ to build similarities between state-action pairs, therefore in very sparse reward environments or with very limited coverage of the state-action space, the quality of the pseudometric learned might conflate state-action pairs together, and hence be less adapted to learn a meaningful measure of similarity.

### 3.5 TRACTABLE BONUS

Once the pseudometric $d_\Phi$ is learned, we can define a lookup bonus introduced in section 3.1. Given a monotonously decreasing function $f$, we have: $b(s, a) = f\big(\min_{(\hat{s},\hat{a}) \in \mathcal{D}} d_\Phi(\hat{s}, \hat{a}; s, a)\big)$. This bonus

has a complexity that is linear in the size of $\mathcal{D}$ and in the dimension of the representation $\Phi(s, a)$. As we are considering datasets with large numbers of transitions ($\sim 10^6$), this makes each actor step in Equation (1) computationally expensive.

Therefore we pre-compute the $k$-nearest neighbors of each state $s \in \mathcal{D}$ according to the Euclidean distance $d_\Psi$ induced by $\Psi$; $d_\Psi(s_1, s_2) = \|\Psi(s_1) - \Psi(s_2)\|$.

We note: $\mathcal{H}(s) = \Big\{(\hat{s}, \hat{a}) \in \mathcal{D} |\ \hat{s} \text{ is a } k\text{-nearest neighbor of } s \text{ for } d_\Psi\Big\}$.

We infer the approximate distance bonus $\bar{b}(s, a) = f\big(\min_{\hat{s},\hat{a} \in \mathcal{H}(s)} d_\Phi(s, a; \hat{s}, \hat{a})\big)$.

Pre-computing the $k$-nearest neighbors is expensive (the brute force complexity is quadratic in the size of the dataset, and linear in the dimension of the representation $\Psi$). In our experiments, we use a kd-tree algorithm (Friedman et al., 1977) from scikit-learn (Pedregosa et al., 2011). With multiprocessing ($\sim 50$ CPUs), precomputing nearest neighbors did not take more than a couple hours even for the largest dataset ($2.10^6$ transitions). If the size of the dataset were to be larger, we can naturally scale our method with approximate nearest neighbor methods.

### 3.6 Algorithm

We now compile the results from this section and present the pseudocode of our method in Algorithms 1 and 2. We refer to the combination of both as PLOff (**P**seudometric **L**earning **Off**line RL).

**Algorithm 1** Bonus learning.

Initialize $\Phi$, $\Psi$ networks.
**for** step $i = 1$ to $N$ **do**
  Train $\Phi$: $\min_\Phi \hat{\mathcal{L}}_\Phi$
  Train $\Psi$: $\min_\Psi \hat{\mathcal{L}}_\Psi$
Initialize $k$-nearest neighbors array $H$.
**for** step $j = 1$ to $|\mathcal{D}|$ **do**
  Compute k-nearest neighbors of $\Psi(s_j)$.
  Add k-nearest neighbors to the array $H$.

**Algorithm 2** Actor-Critic Training.

Initialize action-value network $Q_\omega$, target network $Q_{\bar{\omega}}$, $Q_\omega$ and policy $\pi_\theta$.
**for** step $i = 0$ to $K$ **do**
  Train $Q_\omega$: $\min_\omega \big(Q_\omega(s, a) - r - Q_{\bar{\omega}}(s', \pi_\theta(s'))\big)^2$
  Train $\pi_\theta$: $\max_\theta Q_\omega(s, \pi_\theta(s)) + f(\min_{\hat{s},\hat{a} \in H[s]} \|\Phi(\hat{s}, \hat{a}) - \Phi(s, \pi_\theta(s))\|)$
  Update target network $Q_{\bar{\omega}} := Q_\omega$

## 4 Experiments

In this section we conduct an experimental study for the proposed approach. We evaluate it on a series of hand manipulation tasks (Rajeswaran et al., 2018), as well as MuJoCo locomotion tasks (Todorov et al., 2012; Brockman et al., 2016) with multiple data collection strategies from Fu et al. (2020). We first show the details of the learning procedure of the pseudometric, before showing its performance against several baselines from Fu et al. (2020). All implementation details can be found in Appendix C.

### 4.1 Evaluation environments

We evaluate PLOff on four hand manipulation tasks (Rajeswaran et al., 2018): nailing a hammer, opening a door, manipulating a pen and relocating a ball. We also evaluate PLOff on MuJoCo locomotion tasks (Brockman et al., 2016) where the goal is to maximize the distance traveled: Walker2d, HalfCheetah and Hopper. We provide visualization of the environments in Figure 1. For each environment we consider multiple datasets $\mathcal{D}$ from the D4RL benchmark (Fu et al., 2020). On

hand manipulation tasks, these datasets are the following, "human": transitions collected by a human operator, "cloned": transitions collected by a policy trained with behavioral cloning interacting in the environment + the initial demonstrations, "expert": transitions collected by a fine-tuned RL policy interacting in the environment. On locomotion tasks, the datasets are the following, "random": transitions collected by a random policy, "medium-replay" the first 1M transitions collected by a SAC agent (Haarnoja et al., 2018) trained from scratch on the environment, "medium" transitions collected by a policy with suboptimal performance, "medium-expert": transitions collected by a near optimal policy + transitions collected by a suboptimal policy. To have comparable range of rewards between environments, we scale offline rewards by dividing them by $\max_{\mathcal{D}} r - \min_{\mathcal{D}} r$ and learn a policy on this scaled reward (which leaves the optimal policies unchanged).

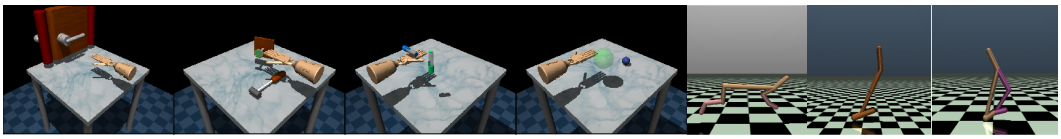

Figure 1: Visualization of the environments considered. From left to right: Door, Hammer, Pen, Relocate, HalfCheetah, Hopper, Walker2d.

### 4.2 Pseudometric learning

We concurrently learn the deep networks $\Phi$ and $\Psi$ by minimizing the losses $\hat{\mathcal{L}}_\Phi$ and $\hat{\mathcal{L}}_\Psi$. State-action pairs are concatenated (and states only in the case of $\Psi$) and are passed to a 2-layer network of hidden layers of size (1024, 32). Note that the concatenation step could be preceded by two disjoint layers to which the state and action are passed (thus making it more handy for visual-based obseravations). We sample 256 actions to derive the bootstrapped estimate (loss $\hat{\mathcal{L}}_\Psi$). We optimize $\hat{\mathcal{L}}_\Phi$ and $\hat{\mathcal{L}}_\Psi$ using the Adam optimizer (Kingma & Ba, 2015) with batches of state-action pairs and states of size 256.

We show in Figure 2 the decreasing learning curves for $\hat{\mathcal{L}}_\Phi$ and $\hat{\mathcal{L}}_\Psi$. In Figure 3, we show that the distribution of the learned distance $d_\Phi$ between state-action couples and perturbated versions of themselves (with Gaussian noise either on the state or the action). We show that the distance respects the intuition that the greater the perturbation is, the larger the distance becomes. Finally we provide visualization of the state similarities learned by $\Psi$ in Appendix E.

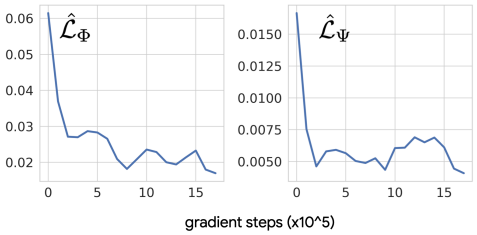


Figure 2: Learning curve of $\Phi$ and $\Psi$, for the Walker2d environment together with the "medium-replay" dataset from Fu et al. (2020). We show the values (averaged over batch) of $\hat{\mathcal{L}}_\Phi$ (left) and $\hat{\mathcal{L}}_\Psi$ (right) throughout the learning procedure.

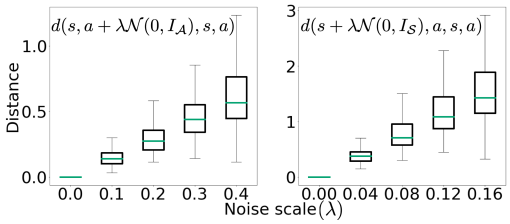


Figure 3: Influence of noise on the distance. We show on the left the learned distance of a state-action pair $(s, a)$ to a perturbated version of itself $(s, a + \lambda\mathcal{N}(0, I_\mathcal{A})$. We show on the right the learned distance of a state-action pair $(s, a)$ to a perturbated version of itself $(s + \lambda\mathcal{N}(0, I_\mathcal{S}), a)$.

### 4.3 Agent training with pseudometric bonus

In this section, we empirically evaluate the performance of an agent trained with a bonus based on the pseudometric. In our experiments, we use TD3 (Fujimoto et al., 2019). It is an off-policy actor-critic algorithm (Konda & Tsitsiklis, 2000) which builds on top of DDPG (Lillicrap et al., 2016).

We use an implementation of TD3 where we load the dataset of experiences $\mathcal{D}$ in its replay buffer. We concurrently learn a policy $\pi_\theta$ and an action-value function $Q_\omega$. The critic step is the same as the one described in Fujimoto et al. (2019). However we modify the actor loss to incorporate the pseudometric bonus, as described in Algorithm 2: $\max_\theta \mathbb{E}_{s\sim\mathcal{D}}\Big(Q_\omega(s,\pi_\theta(s)) + \bar{b}(s,\pi_\theta(s))\Big)$.

We found that the bonus $\bar{b}(s,a) := \alpha \max(0, Q_{\bar{\omega}}(s,a)) \exp(-\beta d_{\mathcal{D}}(s,a))$ leads to strong empirical results. This bonus form is quite natural, since it uses the action-value function to scale the value of the bonus. We limit the influence of the bonus on positive action-values (in the case of negative action-values, our bonus would incentivize to maximize the distance to the dataset $d_{\mathcal{D}}$). Another natural way to define the bonus would be to shift the reward function to make it positive (but problematic in environments where early termination is desirable).

We ran a hyperparameter search on $\alpha \in \{0.1, 0.5, 1.\}$ and $\beta \in \{0.1, 0.25, 0.5\}$ and the number of gradient steps $N \in \{100000, 500000\}$. We select the best hyperparameters for each family of environments (locomotion and hand manipulation), and re-run for 10 seeds. For each seed we evaluate the resulting policy on 10 episodes. We report aggregated performance in Table 2 and the complete breakdown in Table 2. We compare the method with numerous baselines: AWR (Peng et al., 2019), Behavioral Cloning (Pomerleau, 1991), BEAR (Kumar et al., 2019), BRAC (Wu et al., 2019), BCQ (Fujimoto et al., 2018) and CQL (Kumar et al., 2020).

| Algorithm | BC | BEAR | BRAC-p | BRAC-v | AWR | BCQ | CQL | PLOFF |
|---|---|---|---|---|---|---|---|---|
| Locomotion | 25.0 | 39.8 | 29.8 | 31.4 | 26.8 | 40.4 | **52.3** | 43.6 |
| Hand manipulation | 36.7 | 38.3 | 0.4 | -0.4 | 32.2 | 39.6 | 40.3 | **41.3** |

Table 1: Average performance of PLOFF on hand-manipulation tasks and locomotion tasks. Complete breakdown of performance is given in Table 2. We report the results of the baselines using performance results reported by Fu et al. (2020). Performance is averaged over the manipulation tasks (Hammer, Door, Relocate, Pen) and locomotion (Walker2d, Hopper, HalfCheetah), for 10 evaluation episodes for 10 seeds, following recommendations from Henderson et al. (2018).

Table 1 shows the performance of PLOFF on the D4RL benchmark. It tops other methods on hand manipulation tasks, and tops all methods but CQL on locomotion tasks. However, even if PLOFF performs well across the board, the approach does not solve the common failure cases shared by all methods (random datasets as well as datasets with human operated transitions, see Table 2). We leave to future work whether PLOFF, combined with other methods (in particular CQL which operates on the critic side rather than the actor side) could lead to a solution to these failure cases.

## 5 Concluding Remarks

We introduced a new paradigm to compute a pseudometric-based bonus for offline RL. We learn policies consistent with the behavior policy that generated the collected transitions, and hence reduce action extrapolation error.

We showed how to derive a pseudometric from logged transitions, extending existing work from Castro (2020) from the online to the offline setting and from pseudometrics on state space to pseudometrics on state-action space. We showed that the pseudometric we desire to learn is the fixed point of an operator, and we provide a neural architecture as well as a loss to learn it.

Conceptually, our bonus introduces a larger computational cost against other approaches that reduce extrapolation errors. We argue that this is actually a desirable direction of research for offline RL. In the presence of a fixed dataset of transitions, and since we cannot add new transitions into memory, we should insist on the other side of the well-known memory-computation trade-off.

We demonstrated in our experimental study that our method performs comparably to existing state-of-the-art methods (tops other methods on hand manipulation task, second to top on locomotion tasks).

## REFERENCES


Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. *International Conference on Learning Representations (ICLR)*, 2021.

Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fund. math*, 1922.

Marc G. Bellemare, S. Srinivasan, Georg Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Neural Information Processing Systems (NeurIPS)*, 2016.

R. Bellman. A markovian decision process. *Indiana University Mathematics Journal*, 1957.

D. Bertsekas and J. Tsitsiklis. Neuro-dynamic programming. 1996.

Dimitri P Bertsekas and John N Tsitsiklis. Some aspects of parallel and distributed iterative algorithms-a survey. *Automatica*, 1991.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

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

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. *Neural Information Processing Systems (NeurIPS)*, 1994.

Jacob Buckman, Carles Gelada, and Marc G Bellemare. The importance of pessimism in fixed-dataset policy optimization. *International Conference on Learning Representations (ICLR)*, 2021.

Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *AAAI Conference on Artificial Intelligence*, 2020.

Gheorghe Comanici, Prakash Panangaden, and Doina Precup. On-the-fly algorithms for bisimulation metrics. In *International Conference on Quantitative Evaluation of Systems*, 2012.

Robert Dadashi, Léonard Hussenot, M. Geist, and O. Pietquin. Primal wasserstein imitation learning. *International Conference on Learning Representations (ICLR)*, 2021.

Thomas Dean and Robert Givan. Model minimization in markov decision processes. In *AAAI Conference on Artificial Intelligence*, 1997.

Gabriel Dulac-Arnold, Daniel J. Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *CoRR*, abs/1904.12901, 2019.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 2005.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Uncertainty in Artificial Intelligence (UAI)*, 2004.

Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 2011.

Norman Ferns and Doina Precup. Bisimulation metrics are optimal value functions. In *Uncertainty in Artificial Intelligence (UAI)*, 2014.

Norman Ferns, Pablo Samuel Castro, Doina Precup, and Prakash Panangaden. Methods for computing state similarity in markov decision processes. *Uncertainty in Artificial Intelligence (UAI)*, 2006.

J. Friedman, J. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 1977.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

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

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning (ICML)*, 2019.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 2003.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018.

Peter Henderson, R. Islam, Philip Bachman, Joelle Pineau, Doina Precup, and D. Meger. Deep reinforcement learning that matters. *AAAI Conference on Artificial Intelligence*, 2018.

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

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Neural Information Processing Systems (NeurIPS)*, 2020.

Kee-Eung Kim and H. Park. Imitation learning via kernel mean embedding. In *AAAI Conference on Artificial Intelligence*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Diederik P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In *Neural Information Processing Systems (NeurIPS)*, 2000.

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

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

Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 2003.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*.

Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning (ICML)*, 2019.

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

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2006.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

L. J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 1992.

S. Mahadevan and M. Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 2007.

Francisco S. Melo and M. Lopes. Learning from demonstration using mdp induced metrics. In *European Conference on Machine Learning (ECML)*, 2010.

V. Mnih, K. Kavukcuoglu, D. Silver, Andrei A. Rusu, J. Veness, Marc G. Bellemare, A. Graves, Martin A. Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, S. Petersen, C. Beattie, A. Sadik, Ioannis Antonoglou, H. King, D. Kumaran, Daan Wierstra, S. Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

R. Munos and Csaba Szepesvari. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 2008.

Kimia Nadjahi, Romain Laroche, and Rémi Tachet des Combes. Safe policy improvement with soft baseline bootstrapping. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019.

Fabian Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, Mathieu Blondel, Gilles Louppe, P. Prettenhofer, R. Weiss, Ron J. Weiss, J. VanderPlas, Alexandre Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 2011.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)*, 2011.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 1991.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Robotics: Science and Systems (RSS)*, 2018.

Balaraman Ravindran and Andrew G Barto. Relativized options: Choosing the right transformation. In *International Conference on Machine Learning (ICML)*, 2003.

Balaraman Ravindran and Andrew G Barto. Approximate homomorphisms: A framework for non-exact minimization in markov decision processes. 2004.

Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, 2005.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *International Conference on Learning Representations (ICLR)*, 2015.

J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. 1991.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015.

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

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, 2014.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

Thiago D Simão, Romain Laroche, and Rémi Tachet des Combes. Safe policy improvement with an estimated baseline policy. *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2020.

R. Sutton and A. Barto. Introduction to reinforcement learning. 1998.

R. Sutton, David A. McAllester, Satinder Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Neural Information Processing Systems (NeurIPS)*, 1999.

Jonathan Taylor, Doina Precup, and Prakash Panagaden. Bounding performance loss in approximate mdp homomorphisms. *Neural Information Processing Systems (NeurIPS)*, 2008.

S. Thrun. Efficient exploration in reinforcement learning. 1992.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. *International Conference on Intelligent Robots and Systems*, 2012.

Elise van der Pol, Thomas Kipf, Frans A Oliehoek, and Max Welling. Plannable approximations to mdp homomorphisms: Equivariance under actions. *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2020.

C. Villani. Optimal transport: Old and new. 2008.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Çaglar Gülçehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nature*, 2019.

Alicia P Wolfe and Andrew G Barto. Decision tree methods for finding reusable mdp homomorphisms. In *The National Conference on Artificial Intelligence*, 2006.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Neural Information Processing Systems (NeurIPS)*, 2020.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *International Conference on Learning Representations (ICLR)*, 2021.

# A RELATED WORK

**Offline Reinforcement Learning.** Offline RL (Lagoudakis & Parr, 2003; Ernst et al., 2005; Riedmiller, 2005; Pietquin et al., 2011; Lange et al.; Levine et al., 2020) methods suffer from overestimation of state-action pairs that are not in the support of logged transitions. A number of methods have been explored to mitigate this phenomenon, by constraining the learned policy to be close in terms of a probabilistic distance to the behavioral policy (Jaques et al., 2019; Wu et al., 2019; Kumar et al., 2019; Siegel et al., 2020; Peng et al., 2019; Fujimoto et al., 2018), or a *pessimistic* (Buckman et al., 2021) estimate of either the Q-value or the bootstrapped Q-value (Kumar et al., 2020; Laroche et al., 2019; Simão et al., 2020; Nadjahi et al., 2019). Fujimoto et al. (2018); Kumar et al. (2019); Wu et al. (2019) estimate the behavior policy using a conditional VAE (Kingma & Welling, 2014), and constrain the policy learned offline with a distance to the behavior policy using the Kullback-Leibler divergence, the Wasserstein distance and the Maximum-Mean Discrepancy respectively.

In this work we trade the problem of estimating the behavioral policy (which is specially problematic if it is multimodal) with the computational cost of a lookup; and instead of penalizing the policy with a distribution-based distance, we learn a task relevant pseudometric instead. As our bonus is based on learned structural properties of the MDP we can also draw a connection to model-based off-policy RL (Kidambi et al., 2020; Yu et al., 2020; Argenson & Dulac-Arnold, 2021).

**State-action similarity in MDPs.** Bisimulation relations are a form of state abstraction (Li et al., 2006), introduced in the context of MDPs by Givan et al. (2003), where states with the same rewards and transitions are aggregated. As this definition is strict, a number of works define approximate versions of it. For example, Dean & Givan (1997) use a bound rather than an exact equivalence. Similarly, Ferns et al. (2004) introduce bisimulation metrics (Ferns et al., 2011; 2006), which use the reward signal to decide the proximity between two states. This makes bisimulation metrics close to the difference between optimal values between two states (Ferns & Precup, 2014). Learning a bisimulation metric online (Castro, 2020; Comanici et al., 2012) has been shown to be beneficial to learn controllable representations that eliminates unnecessary details of the state (Zhang et al., 2021) or can be used as an auxiliary loss which leads to improvement performance on the Atari benchmark (Gelada et al., 2019). Castro (2020) introduces the Siamese network architecture, as well as a loss, to derive the bisimulation metric online. Our work differs as it learns a pseudometric on state-action pairs rather than states, and is based on offline transitions.

Ravindran & Barto (2003) introduces MDP homomorphism as another form of abstraction which is state-action dependent rather than state dependent. Again as the partitioning induced by a homomorphism is too restrictive to be useful in practice, a number of work has looked into relaxed versions (Ravindran & Barto, 2004; Wolfe & Barto, 2006; van der Pol et al., 2020; Taylor et al., 2008). Our work is closer to bisimulation metrics since it introduces the similarity between states-actions as a difference between reward accumulated when following the same sequence of actions (except for the first one).

## B PROOFS

**Proposition 3.1.** *Let $d$ be a pseudometric in $\mathbb{M}$, then $\mathcal{F}(d)$ is a pseudometric in $\mathbb{M}$.*

*Proof.* Let $d$ be a pseudometric in $\mathbb{M}$, we show that $\mathcal{F}(d)$ respects all properties in Definition 1 and therefore is a pseudometric. Let $(s_1, a_1), (s_2, a_2), (s_3, a_3) \in \mathcal{S} \times \mathcal{A}$ and their associated rewards $r_1, r_2, r_3$ and next states $s'_1, s'_2, s'_3$:

- the pseudo-distance of a couple to itself is null:

$$\mathcal{F}(d)(s_1, a_1; s_1, a_1) = \underbrace{|r_1 - r_1|}_{=0} + \gamma \mathbb{E}_{u \in \mathcal{U}(\mathcal{A})} \underbrace{d(s'_1, u; s'_1, u)}_{=0 \text{ since } d \text{ is a pseudometric}} = 0;$$

- symmetry:

$$\mathcal{F}(d)(s_1, a_1; s_2, a_2) = \underbrace{|r_1 - r_2|}_{=|r_2 - r_1|} + \gamma \mathbb{E}_{u \in \mathcal{U}(\mathcal{A})} \underbrace{d(s'_1, u; s'_2, u)}_{=d(s'_2, u; s'_1, u) \text{ since } d \text{ is a pseudometric}} = \mathcal{F}(d)(s_2, a_2; s_1, a_1);$$

- triangular inequality:

$$\begin{aligned}\mathcal{F}(d)(s_1, a_1; s_3, a_3) &= |r_1 - r_3| + \gamma \mathbb{E}_{u \in \mathcal{U}(\mathcal{A})} d(s'_1, u; s'_3, u) \\ &\leq |r_1 - r_2| + |r_2 - r_3| + \gamma \mathbb{E}_{u \in \mathcal{U}(\mathcal{A})} d(s'_1, u; s'_2, u) + d(s'_2, u; s'_3, u) \\ &\leq \mathcal{F}(d)(s_1, a_1; s_2, a_2) + \mathcal{F}(d)(s_2, a_2; s_3, a_3).\end{aligned}$$

□

**Proposition 3.2.** *Let $d$ be a pseudometric in $\mathbb{M}$, we note $\|d\|_\infty$ as $\max_{s,s' \in \mathcal{S}} \max_{a,a' \in \mathcal{A}} d(s, a; s', a')$. The operator $\mathcal{F}$ is a $\gamma$-contraction for $\| \cdot \|_\infty$.*

*Proof.* Let $d_1, d_2 \in \mathbb{M}$, let $(s_1, a_1), (s_2, a_2) \in \mathcal{S} \times \mathcal{A}$ and their associated rewards $r_1, r_2$ and next states $s'_1, s'_2$, we have:

$$\begin{aligned}&\mathcal{F}(d_1)(s_1, a_1; s_2, a_2) - \mathcal{F}(d_2)(s_1, a_1; s_2, a_2) \\ &\qquad = |r_1 - r_2| - |r_1 - r_2| + \gamma \mathbb{E}_{u \in \mathcal{U}(\mathcal{A})} d_1(s'_1, u; s'_2, u) - \gamma \mathbb{E}_{u \in \mathcal{U}(\mathcal{A})} d_2(s'_1, u; s'_2, u) \\ &\qquad = \gamma \mathbb{E}_{u \in \mathcal{U}(\mathcal{A})} d_1(s'_1, u; s'_2, u) - d_2(s'_1, u; s'_2, u).\end{aligned}$$

Therefore, we have:

$$\begin{aligned}|\mathcal{F}(d_1)(s_1, a_1; s_2, a_2) - \mathcal{F}(d_2)(s_1, a_1; s_2, a_2)| &\leq \gamma \mathbb{E}_{u \in \mathcal{U}(\mathcal{A})} |d_1(s'_1, u; s'_2, u) - d_2(s'_1, u; s'_2, u)| \\ &\leq \gamma \max_{u \in \mathcal{A}} |d_1(s'_1, u; s'_2, u) - d_2(s'_1, u; s'_2, u)| \\ &\leq \gamma \max_{s,s' \in \mathcal{S}} \max_{u,u' \in \mathcal{A}} |d_1(s, u; s', u') - d_2(s, u; s', u')| \\ &\leq \gamma \|d_1 - d_2\|_\infty.\end{aligned}$$

We thus have that $\|\mathcal{F}(d_1) - \mathcal{F}(d_2)\|_\infty \leq \gamma \|d_1 - d_2\|_\infty$, therefore $\mathcal{F}$ is a $\gamma$-contraction for $\| \cdot \|_\infty$. □

**Proposition 3.3.** *$\mathcal{F}$ has a unique fixed point $d^*$ in $\mathbb{M}$. Suppose $d_0 \in \mathbb{M}$ then $\lim_{n \to \infty} \mathcal{F}^n(d_0) = d^*$.*

*Proof.* This is a direct application of the Banach theorem (Banach, 1922). $\mathcal{F}$ is a $\gamma$-contracting operator with $\gamma \in [0, 1)$, in the metric space $((\mathcal{S} \times \mathcal{A}) \times (\mathcal{S} \times \mathcal{A}), \| \cdot \|_\infty)$, therefore using the Banach theorem we have that $\mathcal{F}$ has a unique fixed point $d^*$ and $\forall d_0 \in \mathbb{M}, \lim_{n \to \infty} \mathcal{F}^n(d_0) = d^*$. □

**Proposition 3.4.** *Under common assumptions of state-action coverage, the repeated application of $\hat{\mathcal{F}}$ converges to the fixed point $d^*$ of $\mathcal{F}$.*

*Proof.* The state-action coverage assumption is the following: $\exists \epsilon > 0$ such that for any pairs of state-action pairs $(s, a), (\hat{s}, \hat{a}) \in (\mathcal{S} \times \mathcal{A}) \times (\mathcal{S} \times \mathcal{A})$, $(s, a), (\hat{s}, \hat{a})$ is sampled with at least probability $\epsilon$.

The repeated application of $\hat{\mathcal{F}}$ is an asynchronous fixed point iteration scheme. The convergence to $d^*$ (almost surely) is a direct application of Proposition 3 from Bertsekas & Tsitsiklis (1991). Note that the state-action coverage assumption enables to apply this result since all pairs of state-action are visited an infinite number of times (almost surely).

□

## C IMPLEMENTATION

In this section, we provide a detailed description of the experimental study. We are working on making the code open-source.

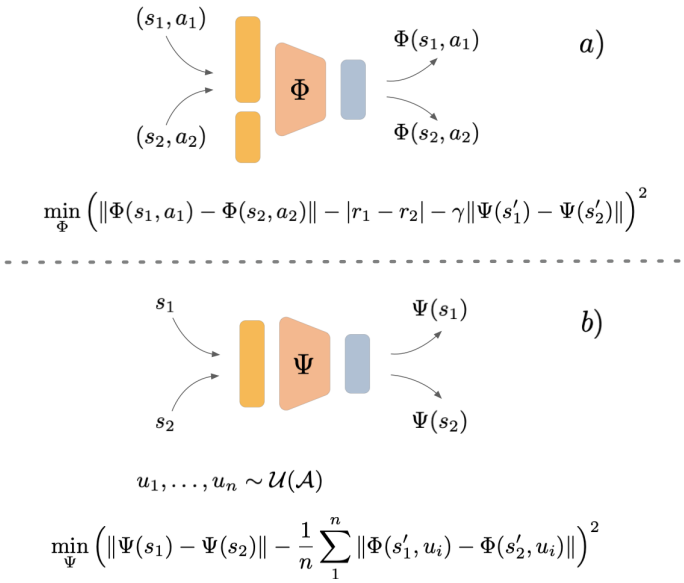


Figure 4: Architecture details of pseudometric learning. Two pairs of Siamese networks $\Phi$ and $\Psi$ are concurrently optimized.

**Offline datasets preprocessing.** We use datasets from Fu et al. (2020). We scale the rewards by dividing them by $\max_{r \sim \mathcal{D}} r - \min_{r \sim \mathcal{D}} r$ for both pseudometric learning and policy learning. This enables to have comparable range of rewards between environments.

**Pseudometric learning.** The Siamese networks $\Phi$ et $\Psi$ have the same architecture: a two-layer MLP of size $(1024, 32)$ with relu activation on top of the first layer. Note that $\Phi$ takes the concatenation of state and action as input, whereas $\Psi$ only takes the state as an input. We use a discount factor $\gamma = 0.9$ (which is different than the discount factor from the agent) in the experiments (we noticed some instabilities on human datasets, which contains less transitions than the rest of the datasets, if the discount factor is larger).

We minimize the losses $\hat{\mathcal{L}}_\Phi$ and $\hat{\mathcal{L}}_\Psi$ using the Adam optimizer with a learning rate $10^{-3}$. The batch size used to compute the losses is $256$. The bootstrapped estimate in $\hat{\mathcal{L}}_\Phi$ is estimated with $256$ actions sampled uniformly. We train the two networks by iteratively taking a gradient step on each loss for $2.10^4$ gradient steps.

Once the $\Psi$ network is trained, we derive the $k$-nearest neighbors of each state in $\mathcal{D}$ for the distance induced by $\Psi$, with $k = 50$. The nearest neighbors are computed using the scikit-learn implementation of the kd-tree algorithm, taking advantage of multiprocessing (with 50 CPUs).

**Agent training.** We re-implemented the TD3 agent from Fujimoto et al. (2019) in JAX (Bradbury et al., 2018). We use the default hyperparameters (and did not perform HP search) for the exception of the target noise in the bootstrapped estimate that we removed (in the hopper environment, we noticed that target noise in the bootstrapped estimate resulted in overestimation and Q-value divergence).

For the critic, we used a three-layer network with size $(256, 256, 1)$ with relu activation on top of the first two layers. For the policy, we used a three-layer network with size $(256, 256, |\mathcal{A}|)$ where $|\mathcal{A}|$ is the dimension of the action space, with relu activation on top of the first two layers and tanh activation on top of the last layer. We used the Adam optimizer for both the actor and the critic and used a learning rate of $3.10^{-4}$ (consistently with Fujimoto et al. (2019)) and trained them using batch of transitions of size 256 sampled uniformly in $\mathcal{D}$.

We led experiments with the following bonuses $\bar{b}$ (and focused on the first one as it led to better empirical performance):

- $\bar{b}(s, a) = \alpha \max(0, Q_{\bar{\omega}}(s, a)) \exp(-\beta d_{\mathcal{D}}(s, a))$
- $\bar{b}(s, a) = \alpha Q_{\bar{\omega}}(s, a) \exp(-\beta d_{\mathcal{D}}(s, a))$
- $\bar{b}(s, a) = \alpha \exp(-\beta d_{\mathcal{D}}(s, a))$
- $\bar{b}(s, a) = \alpha(1 - \exp(\beta d_{\mathcal{D}}(s, a)))$

We led a hyperparameter search for both the locomotion environments and the hand manipulation environments on $\alpha, \beta, N$ where $\alpha, \beta$ are related to the bonus and $N$ is the number of gradient steps. We selected $\alpha$ in $\{0.1, 0.5, 1.\}$, $\beta \in \{0.1, 0.25, 0.5\}$ and the number of gradient steps $N \in \{100000, 500000\}$ as the better combination on the average normalized performance on the tasks (averaged over 3 seeds). We re-ran the best combination of hyperparameters for 10 seeds and report results averaged over the 10 seeds and 10 evaluation episodes per seed. We found that the best combination for hand manipulation tasks was $\alpha = 1, \beta = 0.25, N = 100000$ and for locomotion tasks was $\alpha = 0.1, \beta = 0.5, N = 500000$.

# D BASELINES

In this section we provide the results for simplified versions of our method.

The first one is the performance of TD3 without any bonus that we note TD3-OFF. The second one is similar to PLOFF although with a bonus based on the Euclidean distance between the concatenation of the state and the action rather than a learned distance, we refer to it as PLOFF-L2. For PLOFF-L2, we used the same experimental evaluation protocol as the one described in the previous section. We led the same hyperparameter search and we found that for both set of tasks, the best combination was $\alpha = 0.1, \beta = 0.5, N = 100000$.

The results in Table 2 show that without a bonus, the performance of policy learned is mediocre. Also notice that PLOFF-L2 reaches strong results on hand manipulations tasks as it significantly tops all other methods, and reaches relatively good performance on locomotion tasks as it is third to CQL and PLOFF. This should not come as a surprise since the L2 distance has been shown to be effective in some continuous control tasks (*e.g.* (Dadashi et al., 2021)). Note however that in more complex tasks (typically vision-based environments), the Euclidean distance is a poor measure of similarity.

| Algorithm | BC | BEAR | BRAC-p | BRAC-v | AWR | BCQ | CQL | PLOFF | PLOFF-L2 | TD3-OFF |
|---|---|---|---|---|---|---|---|---|---|---|
| halfcheetah-random-v0 | 2.1 | 25.1 | 24.1 | 31.2 | 2.5 | 2.2 | **35.4** | 15.7 ± 0.8 | 12.2 ± 0.8 | 28.0 ± 2.6 |
| walker2d-random-v0 | 1.6 | **7.3** | -0.2 | 1.9 | 1.5 | 4.9 | 7.0 | 4.1 ± 6.5 | 4.6 ± 6.4 | 1.1 ± 1.2 |
| hopper-random-v0 | 9.8 | 11.4 | 11.0 | **12.2** | 10.2 | 10.6 | 10.8 | 11.5 ± 0.2 | 10.8 ± 0.1 | 0.9 ± 0.4 |
| halfcheetah-medium-v0 | 36.1 | 41.7 | 43.8 | **46.3** | 37.4 | 40.7 | 44.4 | 39.1 ± 0.7 | 38.2 ± 0.4 | 0.3 ± 4.4 |
| walker2d-medium-v0 | 6.6 | 59.1 | 77.5 | **81.1** | 17.4 | 53.1 | 79.2 | 73.1 ± 7.2 | 61.7 ± 9.9 | 0.0 ± 0.3 |
| hopper-medium-v0 | 29.0 | 52.1 | 32.7 | 31.1 | 35.9 | 54.5 | **58.0** | 38.8 ± 15.0 | 47.8 ± 23.1 | 0.9 ± 0.5 |
| halfcheetah-medium-replay-v0 | 38.4 | 38.6 | 45.4 | **47.7** | 40.3 | 38.2 | 46.2 | 40.8 ± 0.6 | 39.9 ± 0.9 | 35.9 ± 3.6 |
| walker2d-medium-replay-v0 | 11.3 | 19.2 | -0.3 | 0.9 | 15.5 | 15.0 | **26.7** | 10.2 ± 5.3 | 12.4 ± 4.3 | 8.0 ± 4.3 |
| hopper-medium-replay-v0 | 11.8 | 33.7 | 0.6 | 0.6 | 28.4 | 33.1 | **48.6** | 28.3 ± 1.5 | 28.8 ± 1.9 | 9.9 ± 9.4 |
| halfcheetah-medium-expert-v0 | 35.8 | 53.4 | 44.2 | 41.9 | 52.7 | 64.7 | 62.4 | **67.5 ± 10.2** | 51.9 ± 13.3 | 3.4 ± 1.9 |
| walker2d-medium-expert-v0 | 6.4 | 40.1 | 76.9 | 81.6 | 53.8 | 57.5 | **111.0** | 85.5 ± 17.7 | 74.3 ± 7.4 | 0.4 ± 1.5 |
| hopper-medium-expert-v0 | **111.9** | 96.3 | 1.9 | 0.8 | 27.1 | 110.9 | 98.7 | 108.7 ± 11.3 | 107.3 ± 11.2 | 5.5 ± 5.4 |
| Mean Performance | 25.0 | 39.8 | 29.8 | 31.4 | 26.8 | 40.4 | **52.3** | 43.6 ± 6.4 | 40.8 ± 6.6 | 7.3 ± 3.0 |
| pen-human-v0 | 34.4 | -1.0 | 8.1 | 0.6 | 12.3 | **68.9** | 37.5 | 49.7 ± 23.4 | 56.1 ± 14.7 | -0.0 ± 4.0 |
| hammer-human-v0 | 1.5 | 0.3 | 0.3 | 0.2 | 1.2 | 0.5 | **4.4** | 0.9 ± 0.5 | 2.0 ± 2.8 | 0.2 ± 0.0 |
| door-human-v0 | 0.5 | -0.3 | -0.3 | -0.3 | 0.4 | -0.0 | **9.9** | -0.0 ± 0.0 | -0.0 ± 0.0 | -0.3 ± 0.0 |
| relocate-human-v0 | 0.0 | -0.3 | -0.3 | -0.3 | -0.0 | -0.1 | **0.2** | -0.0 ± 0.0 | -0.0 ± 0.0 | -0.3 ± 0.0 |
| pen-cloned-v0 | **56.9** | 26.5 | 1.6 | -2.5 | 28.0 | 44.0 | 39.2 | 41.5 ± 12.6 | 34.9 ± 21.3 | -3.7 ± 0.5 |
| hammer-cloned-v0 | 0.8 | 0.3 | 0.3 | 0.3 | 0.4 | 0.4 | **2.1** | 0.3 ± 0.0 | 0.3 ± 0.0 | 0.2 ± 0.0 |
| door-cloned-v0 | -0.1 | -0.1 | -0.1 | -0.1 | 0.0 | 0.0 | **0.4** | 0.0 ± 0.0 | 0.0 ± 0.0 | -0.2 ± 0.1 |
| relocate-cloned-v0 | **-0.1** | -0.3 | -0.3 | -0.3 | -0.2 | -0.3 | -0.1 | -0.2 ± 0.0 | -0.2 ± 0.0 | -0.2 ± 0.0 |
| pen-expert-v0 | 85.1 | 105.9 | -3.5 | -3.0 | 111.0 | 114.9 | 107.0 | 114.1 ± 13.6 | **120.1 ± 20.3** | -2.8 ± 1.2 |
| hammer-expert-v0 | 125.6 | 127.3 | 0.3 | 0.3 | 39.0 | 107.2 | 86.7 | 104.9 ± 15.4 | **126.4 ± 0.7** | 0.2 ± 0.0 |
| door-expert-v0 | 34.9 | 103.4 | -0.3 | -0.3 | 102.9 | 99.0 | 101.5 | 81.8 ± 25.3 | **103.9 ± 0.5** | -0.1 ± 0.1 |
| relocate-expert-v0 | 101.3 | 98.6 | -0.3 | -0.4 | 91.5 | 41.6 | 95.0 | **102.3 ± 5.0** | 101.1 ± 5.4 | -0.3 ± 0.0 |
| Mean performance | 36.7 | 38.3 | 0.4 | -0.4 | 32.2 | 39.6 | 40.3 | 41.3 ± 8.0 | **45.4 ± 5.5** | -0.6 ± 0.5 |

Table 2: Evaluation of PLOFF, PLOFF-L2, TD3-OFF. We report the results of the baseline using performance results reported by Fu et al. (2020), which do not incorporate standard deviation of performances, since the numbers are based on 3 seeds. In our case we use 10 seeds, following recommendations from Henderson et al. (2018), and evaluate on 10 episodes for each seed before reporting average and standard deviations of performance.

## E METRIC VISUALIZATION

In this section, we show the state similarity learned by PLOFF ($\Psi$ network) and visualize it for MuJoCo locomotion environments (we could not provide such visualizations on Adroit tasks since states cannot be retrieved from observations, which is the condition necessary to render images from observations).

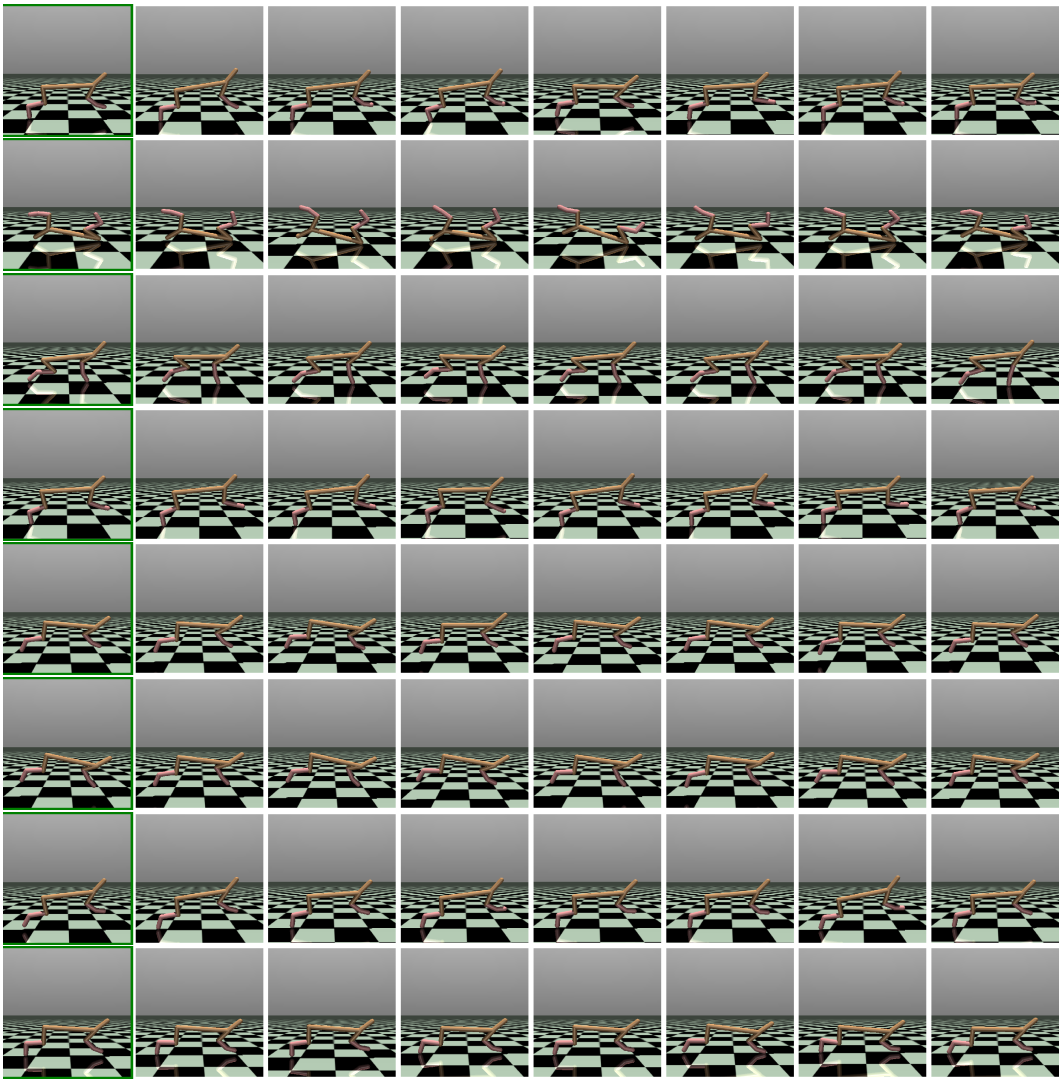

Figure 5: State similarity learned by PLOFF for HalfCheetah on the medium-replay dataset. For each row, the leftmost image is the state for which we compute nearest neighbors in the dataset $\mathcal{D}$ for the metric induced by $\Psi$ (ranked by decreasing level of similarity).

Figure 6: State similarity learned by PLOFF for Hopper on the medium-replay dataset. For each row, the leftmost image is the state for which we compute nearest neighbors in the dataset $\mathcal{D}$ for the metric induced by $\Psi$ (ranked by decreasing level of similarity).

Figure 7: State similarity learned by PLOFF for Walker2d on the medium-replay dataset. For each row, the leftmost image is the state for which we compute nearest neighbors in the dataset $\mathcal{D}$ for the metric induced by $\Psi$ (ranked by decreasing level of similarity).