Offline Reinforcement Learning with Pseudometric Learning

Robert Dadashi [1] **Shideh Rezaeifar** [2] **Nino Vieillard** [1][3] **Léonard Hussenot** [1][4] **Olivier Pietquin** [1] **Matthieu Geist** [1]

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 (closely related to bisimulation metrics) 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 actions. 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 and penalizes non-admissible actions in the evaluation step. Finally, we lead an empirical study on the hand manipulation and locomotion tasks of the D4RL 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 and 2) we exploit this pseudo-

---
[1]Google Research, Brain Team [2]University of Geneva [3]Université de Lorraine, CNRS, Inria, IECL, F-54000 Nancy, France [4]Univ. de Lille, CNRS, Inria Scool, UMR 9189 CRIStAL. Correspondence to: Robert Dadashi <dadashi@google.com>.

metric 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 γ the discount factor for which we assume $\gamma \in [0, 1)$. A stationary deterministic policy π is a mapping from states to actions (the following can easily be extended to stochastic policies). The value function V^π of a policy π is defined as the expected discounted cumulative reward from starting in a particular state and acting according to π: $V^\pi(s) = \mathbb{E}\left(\sum_{i=0}^\infty \gamma^i r(s_i, \pi(s_i)) | s_0 = s)\right)$. The action-value function Q^π is defined as the expected cumulative reward from starting in a particular state, taking an action and then acting according to π: $Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}\left(V^\pi(s')\right)$. The Bellman (1957) operator $\mathcal{B}$ connects an action-value function Q for the state-action pair (s, a) to the action-value function in the subsequent state s': $\mathcal{B}^\pi(Q)(s, a) := r(s, a) + \gamma \mathbb{E}\left(Q(s', \pi(s'))\right)$. Q^π is the (unique) fixed-point of this operator, and the difference between $Q(s, a)$ and its image through the Bellman operator $\|Q - \mathcal{B}^\pi Q\|$ is called a temporal difference error.

An optimal policy π^* maximizes the value function V^{π^*} for all states. In continuous state-action spaces, actor-critic methods (Konda & Tsitsiklis, 2000) are a common paradigm to learn a near-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 π_θ and its associated parametrized action-value function Q_ω. Q_ω 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 π_θ 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 π_θ and Q_ω 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 $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{1:N}$ is 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 respectively.

Pseudometric in MDPs. A core issue in RL is to define a meaningful metric between states or state-action pairs (Le Lan et al., 2021). 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, a pseudometric 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 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:

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

where $\mathcal{W}_1(d)$ is the 1-Wasserstein distance (Villani, 2008) with the distance between states measured according to

---

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

the pseudometric d. Therefore, the bisimulation metric is the limit of the repeated application of the sequence $\left(\mathcal{F}_{\mathcal{S}}{}^n(d_0)\right)_{n \in \mathbb{N}}$ 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 $\mathcal{D}$, 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 monotonically 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 and the critic minimization step (with difference multipliers α_a and α_c). We learn a parametrized policy π_θ and its corresponding action-value function Q_ω. In a schematic way, we sample transitions $(s, a, r, s') \in \mathcal{D}$ and minimize the two following losses:

$$\text{(critic)} \quad \min_{\omega} \| Q_\omega(s, a) - r(s, a)$$
$$- \gamma Q_{\bar{\omega}}(s', \pi_\theta(s')) - \alpha_c b(s', \pi_\theta(s')) \|,$$

$$\text{(actor)} \quad \max_{\theta} Q_\omega(s, \pi_\theta(s)) + \alpha_a b(s, \pi_\theta(s)).$$

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 π 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, plus 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 A.

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

This result indicates the stability of a pseudometric through the operator $\mathcal{F}$ which is not trivial. The stability comes from the deterministic nature of the MDP as well as the fact that the boostrapped estimate is computed for the *same* action at the next states. 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 γ-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 pairs 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:

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

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. *Suppose sufficient coverage of the state-action space: $\exists \epsilon > 0$ 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 ϵ, then 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 (albeit 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 Φ (Bromley et al., 1994) to derive an approximate version of a pseudometric d_Φ. 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 Φ with its parameters. We define the pseudometric d_Φ 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_Φ by minimizing the following loss, which is exactly the temporal difference error $\left(\hat{\mathcal{F}}(d_{\bar{\Phi}}) - d_\Phi\right)^2$:

$$\mathcal{L}_\Phi = \mathop{\mathbb{E}}_{\substack{s_1,a_1,r_1,s'_1 \sim \mathcal{D} \\ s_2,a_2,r_2,s'_2 \sim \mathcal{D}}} \Big(d_\Phi(s_1, a_1; s_2, a_2) - |r_1 - r_2| - \\ \gamma \mathop{\mathbb{E}}_{a' \in \mathcal{U}(\mathcal{A})} d_{\bar{\Phi}}(s'_1, a'; s'_2, a')\Big)^2.$$

We introduce another pair of Siamese networks Ψ (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:

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

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 $\hat{\mathcal{L}}_\Phi$:

$$\hat{\mathcal{L}}_\Phi = \mathop{\mathbb{E}}_{\substack{s_1,a_1,r_1,s'_1 \sim \mathcal{D} \\ s_2,a_2,r_2,s'_2 \sim \mathcal{D}}} \Big(\|\Phi(s_1, a_1) - \Phi(s_2, a_2)\| \\ - |r_1 - r_2| - \gamma|\Psi(s'_1) - \Psi(s'_2)|\Big)^2,$$
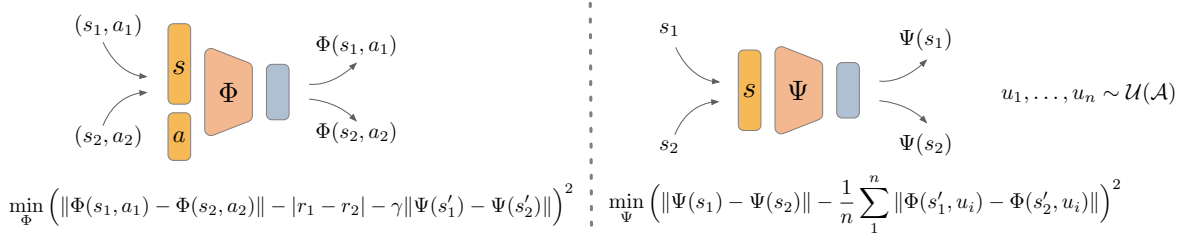
$$\min_{\Phi} \Big(\|\Phi(s_1, a_1) - \Phi(s_2, a_2)\| - |r_1 - r_2| - \gamma \|\Psi(s_1') - \Psi(s_2')\| \Big)^2$$

$$\min_{\Psi} \Big(\|\Psi(s_1) - \Psi(s_2)\| - \frac{1}{n} \sum_1^n \|\Phi(s_1', u_i) - \Phi(s_2', u_i)\| \Big)^2$$

Figure 1. Architecture details of pseudometric learning. Two pairs of Siamese networks Φ and Ψ are concurrently optimized. Left: the pseudometric d_Φ on $\mathcal{S} \times \mathcal{A}$. Right: The pseudometric d_Ψ on $\mathcal{S}$ tracking the boostrapped estimate of $\mathcal{F}(d_\Phi)$.

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

A visual representation of the two pairs of Siamese networks as well as their losses is provided in Figure 1. 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_Φ 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_Φ 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 the exact derivation of the bonus computationally expensive.

Therefore we pre-compute the k-nearest neighbors of each state $s \in \mathcal{D}$ according to the Euclidean distance d_Ψ induced by Ψ; $d_\Psi(s_1, s_2) = \|\Psi(s_1) - \Psi(s_2)\|$. 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 Ψ). In our experiments, we use a kd-tree algorithm (Friedman et al., 1977) from scikit-learn (Pedregosa et al., 2011). With multi-processing (~ 50 CPUs), pre-computing 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.

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

Algorithm 2 Actor-Critic Training.

1: Initialize action-value network Q_ω, target network $Q_{\bar{\omega}}$, Q_ω and policy π_θ.
2: **for** step $i = 0$ to K **do**
3:      Train Q_ω: $\min_\omega \big(Q_\omega(s, a) - r - Q_{\bar{\omega}}(s', \pi_\theta(s')) - \alpha_c \bar{b}(s', \pi(s')) \big)^2$
4:      Train π_θ: $\max_\theta Q_\omega(s, \pi_\theta(s)) + \alpha_a \bar{b}(s, \pi(s))$
5:      Update target network $Q_{\bar{\omega}} := Q_\omega$

Figure 2. Visualization of the environments considered. From left to right: Door, Hammer, Pen, Relocate, HalfCheetah, Hopper, Walker2d.

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 B.

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 2. 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 scaling them in $(0, 1)$ by $r := (r - \min_{\mathcal{D}} r)/(\max_{\mathcal{D}} r - \min_{\mathcal{D}} r)$ and learn a policy on this scaled reward.

4.2. Pseudometric learning

We concurrently learn the deep networks Φ and Ψ 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 Ψ) and are passed to a 2-layer network of layers sizes (1024, 32), with a relu

activation on top of the first layer. 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.

To present a qualitative intuition on the nature of the pseudometric learned, we first present the learned pseudometric on a gridworld environment with walls presented in Figure 3. There is a single reward state and we impose a time limit of 50 steps. We gather all transitions visited by the Q-learning algorithm trained for 500 episodes, with ϵ-exploration ($\epsilon = 0.1$) and a discount factor $\gamma = 0.99$. States and actions are both represented using one-hot encoding. We represent the learned distances (in the sense of d_ψ) from the central state and from the goal state to the rest of the states. Interestingly, the distance learned takes into account the geometry of the environment, hence showing that it is task relevant. In the left part of the gridworld (far from reward), states tend to be conflated together which highlights the limitation of our work in sparse environments.

For the D4RL environments, we show in Figure 4 the decreasing learning curves for $\hat{\mathcal{L}}_\Phi$ and $\hat{\mathcal{L}}_\Psi$. In Figure 5, we show that the distribution of the learned distance d_Φ 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 visualizations of the state similarities learned by Ψ in Appendix C.

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 π_θ and an action-value function Q_ω. The critic loss and actor loss are modified to incorporate the pseudometric bonus, as described in Algorithm 2.
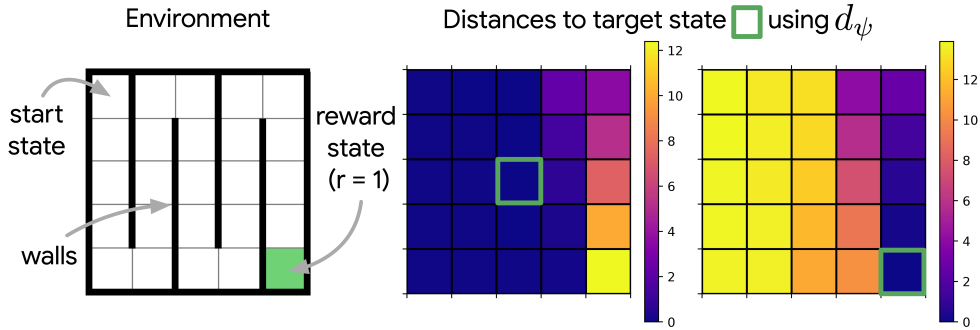
Figure 3. Visualization of the learned pseudometric (center and right) in a gridworld (left). The distance from each state to the "central" state is represented in the center figure, and to the reward state in the right figure.
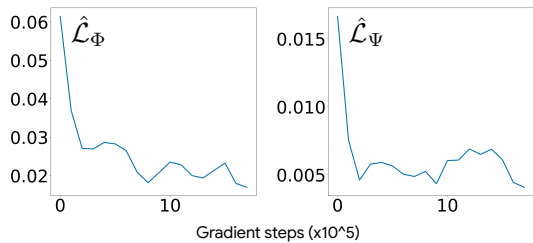

Figure 4. Learning curve of Φ and Ψ, for the Walker2d environment together with the "medium-replay" dataset from Fu et al. (2020). We show the values (averaged over batch) of $\hat{\mathcal{L}}_\Phi$ (left) and $\hat{\mathcal{L}}_\Psi$ (right) throughout the learning procedure.
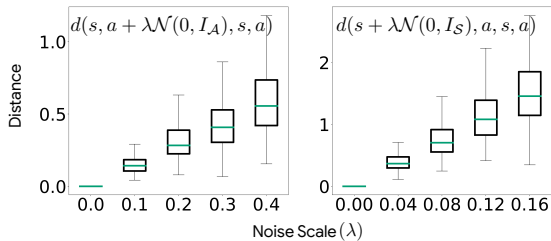

Figure 5. Influence of noise on the distance. We show on the left the learned distance of a state-action pair (s, a) to a perturbated version of itself $(s, a + \lambda\mathcal{N}(0, I_\mathcal{A})$. We show on the right the learned distance of a state-action pair (s, a) to a perturbated version of itself $(s + \lambda\mathcal{N}(0, I_\mathcal{S}), a)$.

We found that the bonus defined as $\bar{b}(s, a) := Q_{\bar{\omega}}(s, a)\exp(-\beta d_\mathcal{D}(s, a))$ 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 (and therefore enables hyperparameters to be more robust).

We ran a hyperparameter search on $\alpha_a, \alpha_c \in \{1, 5, 10\}$ and $\beta \in \{0.1, 0.25, 0.5\}$. 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 performance in Table 1. 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).

Table 1 shows the performance of PLOFF on the D4RL benchmark. On average, 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 1).

4.4. Ablations

We perform two ablations of the proposed method and report results in Table 1. The first one consists in using 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 and hyperparameter search as the proposed method.

The results in Table 1 show that without a bonus, the performance of the policy learned is mediocre. On the other hand, PLOFF-L2 reaches performance slightly below PLOFF. This should not come as a surprise since the Euclidean distance has been shown to be an effective measure of similarity in some continuous control tasks (Dadashi et al., 2021). Note however that in more complex tasks (typically vision-based environments), the Euclidean distance is a poor measure of similarity which makes the learning of a pseudometric necessary.

Algorithm	BC	BEAR	BRAC-v	AWR	BCQ	CQL	PLOFF	PLOFF-L2	TD3-OFF
cheetah-rand	2.1	25.1	31.2	2.5	2.2	**35.4**	1.7 ± 0.1	2.2 ± 0.0	28.0 ± 2.6
walker-rand	1.6	**7.3**	1.9	1.5	4.9	7.0	0.7 ± 1.1	4.7 ± 7.3	1.1 ± 1.2
hopper-rand	9.8	11.4	**12.2**	10.2	10.6	10.8	11.5 ± 0.1	11.6 ± 0.2	0.9 ± 0.4
cheetah-med	36.1	41.7	**46.3**	37.4	40.7	44.4	38.2 ± 0.4	38.0 ± 0.4	0.3 ± 4.4
walker-med	6.6	59.1	**81.1**	17.4	53.1	79.2	73.2 ± 7.4	72.2 ± 3.4	0.0 ± 0.3
hopper-med	29.0	52.1	31.1	35.9	54.5	58.0	**87.0 ± 18.2**	76.7 ± 10.3	0.9 ± 0.5
cheetah-med-rep	38.4	38.6	**47.7**	40.3	38.2	46.2	38.5 ± 1.6	39.0 ± 1.3	35.9 ± 3.6
walker-med-rep	11.3	19.2	0.9	15.5	15.0	**26.7**	<u>23.2 ± 9.3</u>	14.6 ± 6.9	8.0 ± 4.3
hopper-med-rep	11.8	33.7	0.6	28.4	33.1	**48.6**	33.5 ± 10.1	<u>48.2 ± 11.5</u>	9.9 ± 9.4
cheetah-med-exp	35.8	53.4	41.9	52.7	**64.7**	62.4	<u>58.0 ± 8.2</u>	52.1 ± 8.8	3.4 ± 1.9
walker-med-exp	6.4	40.1	81.6	53.8	57.5	**111.0**	98.8 ± 8.2	98.5 ± 8.5	0.4 ± 1.5
hopper-med-exp	111.9	96.3	0.8	27.1	110.9	98.7	**112.1 ± 0.3**	107.6 ± 10.0	5.5 ± 5.4
Mean perf	25.0	39.8	31.4	26.8	40.4	**52.3**	<u>48.0 ± 5.4</u>	<u>47.0 ± 5.7</u>	7.3 ± 3.0
pen-human	34.4	-1.0	0.6	12.3	68.9	37.5	<u>69.6 ± 22.0</u>	**69.9 ± 14.2**	0.0 ± 4.0
hammer-human	1.5	0.3	0.2	1.2	0.5	**4.4**	2.5 ± 1.6	1.7 ± 1.2	0.2 ± 0.0
door-human	0.5	-0.3	-0.3	0.4	-0.0	**9.9**	-0.2 ± 0.2	0.6 ± 1.6	-0.3 ± 0.0
relocate-human	0.0	-0.3	-0.3	-0.0	-0.1	**0.2**	0.0 ± 0.0	0.0 ± 0.1	-0.3 ± 0.0
pen-cloned	**56.9**	26.5	-2.5	28.0	44.0	39.2	35.4 ± 7.7	31.7 ± 10.1	-3.7 ± 0.5
hammer-cloned	0.8	0.3	0.3	0.4	0.4	**2.1**	0.4 ± 0.0	0.4 ± 0.1	0.2 ± 0.0
door-cloned	-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	**-0.1**	-0.3	-0.3	-0.2	-0.3	-0.1	-0.2 ± 0.0	-0.2 ± 0.0	-0.2 ± 0.0
pen-expert	85.1	105.9	-3.0	111.0	**114.9**	107.0	104.8 ± 22.8	111.8 ± 12.0	-2.8 ± 1.2
hammer-expert	**125.6**	127.3	0.3	39.0	107.2	86.7	<u>116.6 ± 9.4</u>	120.0 ± 5.8	0.2 ± 0.0
door-expert	34.9	103.4	-0.3	102.9	99.0	101.5	**104.2 ± 2.0**	<u>104.0 ± 1.4</u>	-0.1 ± 0.1
relocate-expert	101.3	98.6	-0.4	91.5	41.6	95.0	**107.0 ± 2.4**	77.3 ± 1.4	-0.3 ± 0.0
Mean perf	36.7	38.3	-0.4	32.2	39.6	40.3	**45.0 ± 5.7**	<u>43.2 ± 4.3</u>	-0.6 ± 0.5

Table 1. Evaluation of PLOFF. We report the results of the baselines 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. Results are bolded if they are best on average, underlined if within a standard deviation of the best average performance.

5. 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).

6. 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

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

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

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

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

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

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

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. Signature verification using a" siamese" time delay neural network. *Neural Information Processing Systems (NeurIPS)*, 1994.

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

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

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

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

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

Dulac-Arnold, G., Mankowitz, D. J., and Hester, T. Challenges of real-world reinforcement learning. *CoRR*, abs/1904.12901, 2019.

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

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

Ferns, N., Panangaden, P., and Precup, D. Metrics for finite markov decision processes. In *Uncertainty in Artificial Intelligence (UAI)*, 2004.

Ferns, N., Castro, P. S., Precup, D., and Panangaden, P. Methods for computing state similarity in markov decision processes. *Uncertainty in Artificial Intelligence (UAI)*, 2006.

Ferns, N., Panangaden, P., and Precup, D. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 2011.

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

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

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

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

Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning (ICML)*, 2019.

Givan, R., Dean, T., and Greig, M. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 2003.

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

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. *AAAI Conference on Artificial Intelligence*, 2018.

Jaques, N., Ghandeharioun, A., Shen, J. H., Ferguson, C., Lapedriza, A., Jones, N., Gu, S., and Picard, R. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. *Neural Information Processing Systems (NeurIPS)*, 2020.

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

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

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

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

Kumar, A., Fu, J., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Neural Information Processing Systems (NeurIPS)*, 2019.

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

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

Lange, S., Gabel, T., and Riedmiller, M. Batch reinforcement learning. In *Reinforcement learning*.

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

Le Lan, C., Bellemare, M. G., and Castro, P. S. Metrics and continuity in reinforcement learning. *AAAI Conference on Artificial Intelligence*, 2021.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

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

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

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

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

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

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

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

Nadjahi, K., Laroche, R., and des Combes, R. T. Safe policy improvement with soft baseline bootstrapping. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019.

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

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Pietquin, O., Geist, M., Chandramohan, S., and Frezza-Buet, H. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)*, 2011.

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

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Robotics: Science and Systems (RSS)*, 2018.

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

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

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

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *International Conference on Learning Representations (ICLR)*, 2015.

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

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015.

Siegel, N. Y., Springenberg, J. T., Berkenkamp, F., Abdolmaleki, A., Neunert, M., Lampe, T., Hafner, R., and Riedmiller, M. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *Internation Conference on Learning Representations*, 2020.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, 2014.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

Simão, T. D., Laroche, R., and Combes, R. T. d. Safe policy improvement with an estimated baseline policy. *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2020.

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

Sutton, R., McAllester, D. A., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with

function approximation. In *Neural Information Processing Systems (NeurIPS)*, 1999.

Taylor, J., Precup, D., and Panagaden, P. Bounding performance loss in approximate mdp homomorphisms. *Neural Information Processing Systems (NeurIPS)*, 2008.

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

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

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

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

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gülçehre, Ç., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T. P., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nature*, 2019.

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

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

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

Zhang, A., McAllister, R., Calandra, R., Gal, Y., and Levine, S. Learning invariant representations for reinforcement learning without reconstruction. *International Conference on Learning Representations (ICLR)*, 2021.