Goal-Conditioned Reinforcement Learning from Sub-Optimal Data on Metric Spaces

Anonymous authors
Paper under double-blind review

Abstract

We address the problem of learning optimal behavior from sub-optimal datasets for goal-conditioned offline reinforcement learning under sparse rewards, invertible actions and deterministic transitions. To do so, we propose the use of metric learning to approximate the optimal value function instead of classic Temporal-Difference solutions that employ the Bellman operator for their value updates. This representation choice allows us to avoid the out-of-distribution issue caused by the *max* operator of the critic update in the offline setting without any conservative or behavioral constraints on the value function. We introduce distance monotonicity, a property for representations to recover optimality and propose an optimization objective that leads to such property. We use the proposed value function to guide the learning of a policy in an actor-critic fashion, a method we name MetricRL. Experimentally, we show that our method estimates optimal behaviors from severely sub-optimal offline datasets without suffering from out-of-distribution estimation errors. We demonstrate that MetricRL consistently outperforms prior state-of-the-art goal-conditioned RL methods in learning optimal policies from sub-optimal offline datasets.

1 Introduction

Effective decision-making is an integral part of intelligent behavior. To achieve this, learning-based control methods have proven to be a viable option in complex scenarios (Andrychowicz et al., 2020; Silver et al., 2017; Mnih et al., 2013; Peters and Schaal, 2008). In particular, reinforcement learning (RL) allows learning near-optimal behavior through trial-and-error (Sutton and Barto, 2018). However, the online reinforcement learning framework generally requires slow, expensive (and potentially dangerous) online interactions with the environment.

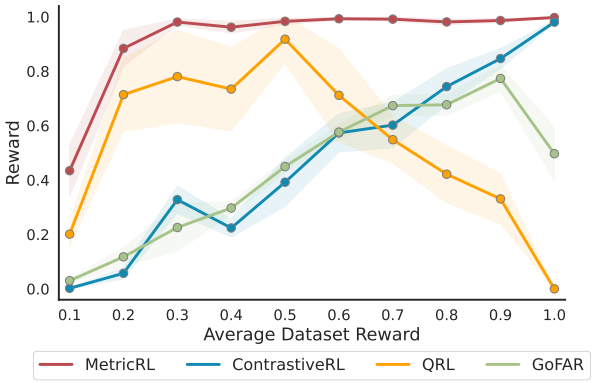


Figure 1: Average reward on `Minigrid DoorKey` (Chevalier-Boisvert et al., 2023) as a function of the expected reward present in the offline dataset. We contribute MetricRL (red line), a novel goal-conditioned offline RL agent able to learn near-optimal behavior from severely sub-optimal datasets.

Offline RL, on the other hand, formalizes the learning of optimal behaviors from a *static* dataset (Levine et al., 2020). This approach offers many advantages over its online counterpart, such as the ability to leverage large-scale datasets to learn complex behavior (Walke et al., 2023; Dasari et al., 2020) without the need of re-collecting data (Shi et al., 2021; Gürtler et al., 2023). Because of the inability to access the environment, in offline RL it is assumed that the dataset already includes suitable information to perform the given task. This data, however, may often be collected by sub-optimal agents. In this work, we address the question of how to learn different (and *better*) behaviors from the one observed in the dataset, regardless of its optimality. We focus on the challenge

of *learning near-optimal behavior from severely sub-optimal datasets*, such as the ones collected by a random policy. We empirically demonstrate in Section 4, and highlight in Figure 1, how under these conditions current offline RL methods (Eysenbach et al., 2022; Wang et al., 2023; Ma et al., 2022; Kostrikov et al., 2021) struggle significantly.

We focus on sparse-reward goal-conditioned offline reinforcement learning which aims at learning optimal behavior to reach multiple goals within the same environment. Motivated by recent work in quasimetric learning (Wang et al., 2023), we explore the use of metric learning to estimate policies from sub-optimal offline data (Balasubramanian and Schwartz, 2002; McInnes et al., 2018). Differently from previous work, by assuming invertible actions, we can naturally exploit symmetry in the learning process of our agent. Endowing the agent with this bias allows for learning an approximation of the optimal value function in severely sub-optimal data conditions without explicitly relying on the Bellman operator, (Bellman, 1954). In turn, this allows us to avoid the out-of-distribution issue caused by the max operator used in dynamic programming solutions, without requiring any additional conservative (e.g., as in CQL (Kumar et al., 2020)) or behavioral (e.g., as in BCQ (Fujimoto et al., 2019), BEAR (Kumar et al., 2019)) regularization of the value function. The core idea of the proposed method is to exploit symmetries of metric spaces to learn an embedding of the state space of the environment such that Euclidean distances are related to the minimum number of actions needed to reach one state from the other. In particular, we introduce the notion of *distance monotonicity* (DM) as a relaxation of isometries of this embedding and show that preserving DM provably allows for the recovery of an optimal policy, *regardless of the quality of the dataset used for training.* We call our method *MetricRL*.

We evaluate MetricRL across a wide range of literature-standard goal-conditioned reinforcement learning tasks. We show how MetricRL outperforms prior goal-conditioned offline reinforcement learning methods in learning near-optimal behavior from severely sub-optimal datasets. Additionally, we show how MetricRL easily scales to high-dimensional observations.

In summary, our contributions are the following:

- **MetricRL**: We propose a novel method that exploits symmetries in latent representation spaces for goal-conditioned offline reinforcement learning.
- **Distance Monotonicity**: We define a new property of these latent representation spaces. We show that, under invertible actions, preserving such property provably leads to policy optimality.
- **Learning from Sub-Optimal Offline Data**: We demonstrate how MetricRL is able to recover optimal behaviors in severely sub-optimal data conditions, outperforming prior state-of-the-art offline RL methods across literature-standard environments.

2 Preliminaries and Assumptions

Goal-Reaching Reinforcement Learning: We consider standard Markov Decision Processes (MDPs) for goal-reaching tasks, $\mathcal{M} = (S, A, T, r, \gamma)$, where S is the state space, A is the action space, $T : S \times A \to S$ is a deterministic transition function (a common assumption in recent offline RL methods (Ma et al., 2022; Park et al., 2023; Wang et al., 2023)), $r : S \times A \to \mathbb{R}$ is a goal-conditioned, sparse reward, i.e., $r(s, a) \neq 0$ iff $T(s, a) = s_g$ (where s_g is the goal-state), and $\gamma \in [0, 1)$ is a discount factor on the future rewards of the agent. We additionally define the goal states to be absorbing and consider the process terminated once these are reached.

The goal of the process is to find an optimal policy $\pi^*(a \mid s, s_g)$ that, given goal state s_g, maximizes the cumulative discounted reward of the agent for any possible starting state. To evaluate the optimality of the policy we resort to the goal-conditioned value function $V^\pi(s, s_g)$, defined as the expected discounted cumulative reward starting in a particular state and acting according to a policy $\pi : V^\pi(s, s_g) = \mathbb{E}_\pi \left[\sum_t \gamma^t r_t | s_0 = s, a = \pi(s, s_g)\right] \quad \forall s \in S$. The value function associated with the optimal policy is referred to as the optimal value function $V^* = V^{\pi^*}$ and, as such, is always greater or equal to any other value functions, i.e. $V^*(s, s_g) \geq V^\pi(s, s_g) \quad \forall s, \pi$.

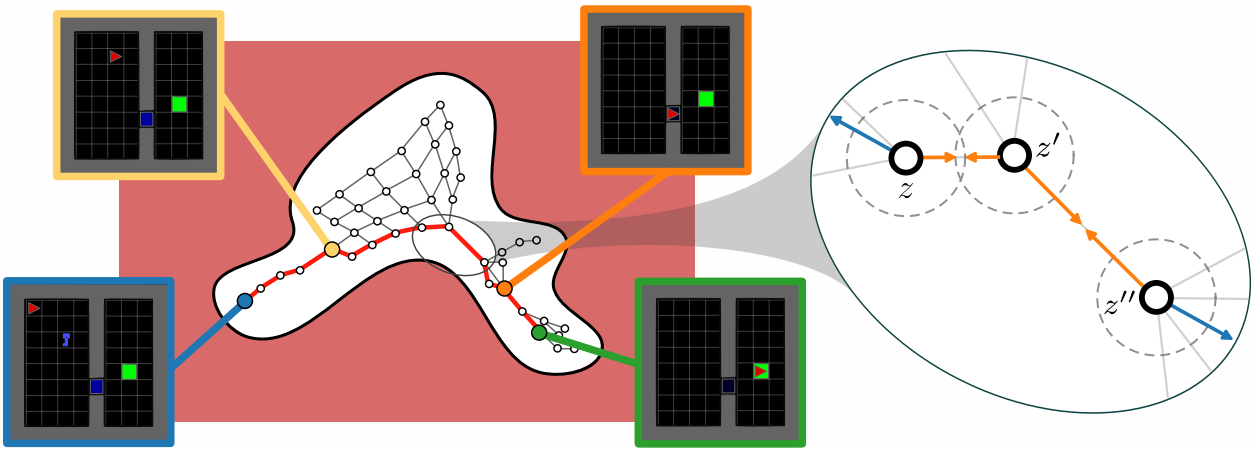


Figure 2: We explore a form of symmetry in representation learning for goal-condition offline reinforcement learning: we learn a metric space in which Euclidean distances between the representation of states (z, z', z'') are related to the value function of the agent. We call our approach **MetricRL**. In the `Minigrid Doorkey` environment, moving greedily to adjacent states translates to the optimal policy (red line) to reach the goal (in green).

Offline Reinforcement Learning: In the offline RL setting, we assume we have access to a dataset D of interactions with the environment described by the MDP and collected by some unknown policy π_β, i.e. $D_{\pi_\beta} = \{(s, a \sim \pi_\beta(a \mid s), r, s' = T(s, a))\}$. A major issue in Offline RL stems from the way dynamic programming methods estimate the optimal value function of the MDP. Most of the current algorithms rely on temporal difference techniques to learn the critic function, e.g. DQN (Mnih et al., 2013), CQL (Kumar et al., 2020), BEAR (Kumar et al., 2019), BCQ (Fujimoto et al., 2019). However, the max operator used to estimate the target value has been shown to result in an overestimation of the expected return, Levine et al. (2020). While this is not an issue in Online RL as the agent has the possibility of exploring overestimated states, it results in catastrophic effects in Offline RL. In this paper, we rely on an alternative technique for learning the critic. As discussed in prior work (Wang et al., 2023; Yang et al., 2020), the dataset D implicitly defines a graph $G = (S, A)$ where the nodes correspond to the states and the edges to the actions of the agent. Here we assume that this graph only has one connected component, i.e., any two states in the dataset are connected through a path on the graph. Furthermore, we assume that there always exists an *inverse* action, i.e., $\exists a' \in A : T(s', a') = s \quad \forall (s, a, s' = T(s, a))$. Note that a' doesn't necessarily need to correspond to the actual opposite action and it can be any viable action. By endowing the graph G with a metric d_S, we can define a corresponding finite metric space (G, d_S). For the tuple to be a valid metric space we need to define the metric $d_S : S \times S \to \mathbb{R}$ to respect the axioms of a metric space (Burago et al., 2001). In particular, we can define the distance between any two states to be the number of edges on the shortest path connecting them (geodesic distance).

3 Method

In this work we focus on learning near-optimal goal-conditioned behavior from sub-optimal offline data. In these conditions, two main problems arise. The first is to learn an optimal behavior without depending on the quality of the distribution of the data used. The second is to avoid the out-of-distribution shift commonly faced in offline reinforcement learning (Levine et al., 2020). We propose to address this using a metric learning approach to estimate the optimal value function. To do so, we start by defining *distance monotonicity* (Section 3.1), a novel property on representations needed to recast the problem of optimal value function estimation into metrics. We propose a loss function to learn maps that respect such property and show how to build an approximation of the value function using distances in this learned representation (Section 3.2). Finally, we define an actor-critic method to learn policies over this approximation and formally prove their optimality (Section 3.3). We call our approach *MetricRL*.

3.1 Distance Monotonicity

Consider a continuous map between the state space of an MDP for goal-reaching tasks and a latent representation space: $\phi : S \to Z \subseteq \mathbb{R}^n$. We can equip this latent vector space with a Euclidean norm to obtain a Euclidean metric space $(Z, \|.\|_2)$. Here we consider distances in the original space, d_S as the minimum number of actions an optimal policy needs to reach one state from the other. We say ϕ is isometric if relative distances in the original state space d_S and the latent metric space d_Z are preserved, i.e., $d_Z(\phi(s), \phi(s')) = d_S(s, s')$, $\forall s, s' \in S$. In fact, if ϕ is isometric then the value function can be defined as simply the norm of the distance between the current state and the goal state A.1, i.e., $V^*(s, s_g) = \gamma^{d_Z(\phi(s),\phi(s'))} r_g$ where r_g represents the reward at the goal and the expectation has been dropped as we assume deterministic transitions.

However, estimating an isometry between these two metric spaces is known to be not always possible (Bourgain, 1985; Matoušek, 2002). To overcome such issue, we consider a relaxation of isometries between metric spaces. Given two metric spaces (S, d_S) and (Z, d_Z) and the corresponding map ϕ between them, we can define the following property of ϕ:

Definition 3.1. We say ϕ is *distance monotonic* (DM) if for all $s_1, s_2, s_3 \in S$, $d_S(s_1, s_3) \leq d_S(s_2, s_3)$ implies $d_Z(z_1, z_3) \leq d_Z(z_2, z_3)$.

A similar property has been described in Ataer-Cansizoglu et al. (2014). There, however, it is imposed on a learned representation by means of explicit constrain in the optimization procedure. Instead, we propose to parameterize the map ϕ_θ and learn it by minimizing the following objective on the dataset D:

$$\mathcal{L}_\theta(D) = \mathbb{E}_D \left[(\|\phi_\theta(s') - \phi_\theta(s)\|_2 - 1)^2 - \lambda \|\phi_\theta(s'') - \phi_\theta(s)\|_2 \right], \tag{1}$$

where (s, s') are any two states connected by an action sampled from D and s'' are other states sampled independently from the dataset at random.

Our loss function balances two requirements on the learned representation: the first term forces the representation to preserve the local distances of states in the graph, encouraging connected states to be separated by a vector of norm one in the latent representation Z. As we show in Figure 2 (right, orange arrows), the representation of connected states (z, z') lies on a circumference of radius one. The second term of the loss maximizes the distance of non-directly connected states, as highlighted in Figure 2 (right, blue arrows). This term of the loss is unbounded if the graph defined by the dataset is not composed of a single connected component. As we discuss in Section 3.4, in cases where the dataset defines multiple connected components (e.g., with image observations) we can always define a synthetic super-node to connect every termination state.

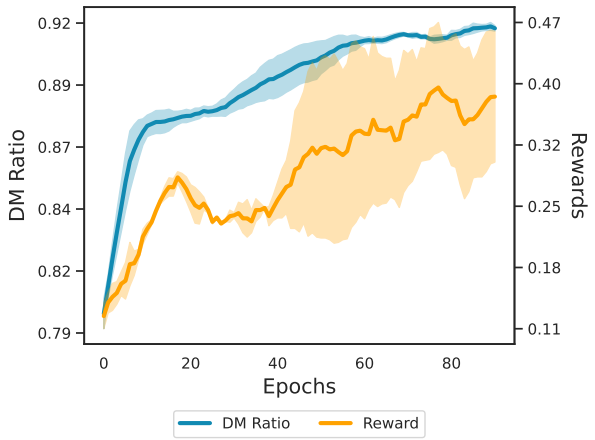


Figure 3: Optimizing Equation 1 increases the ratio of distance monotonic triplets (blue curve) on `Maze2D` (Large). Distance monotonicity is also correlated with an increase in the average return of the agent (orange curve).

The minimization of Equation 1 enforces the learning of a distance monotonic map: in Figure 3 we highlight that the ratio of distance monotonic triplets significantly increases as a function of the training of the map. This can be approximated by discretizing the state space, building an ϵ-graph and the resulting distances in the state space as geodesics on the graph, comparing these distances with distances in the learned representation Z. In Appendix A.2 we provide a more formal intuition of this relationship as well as additional details on the evaluation of the distance monotonicity ratio.

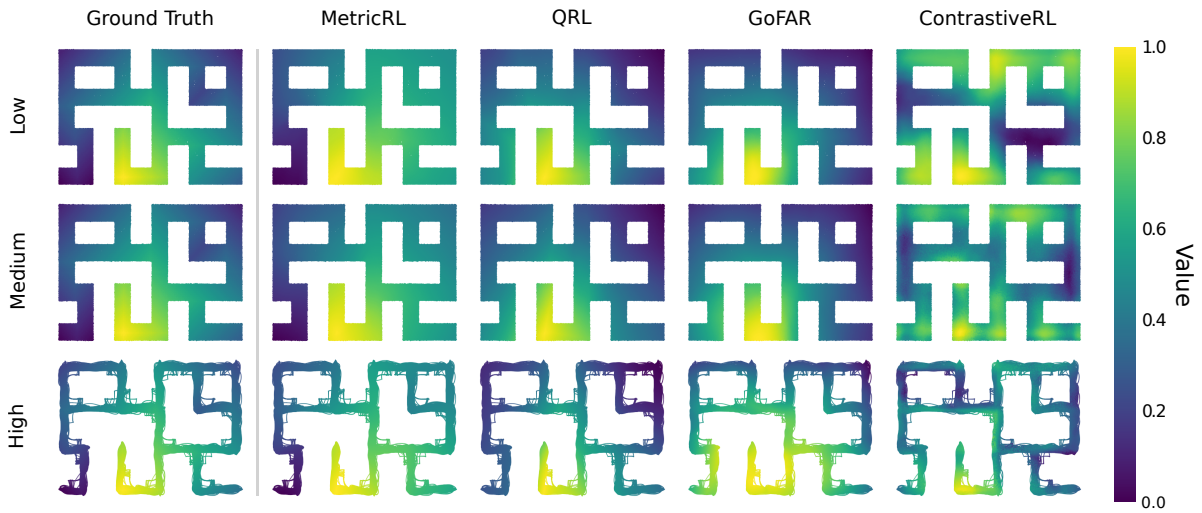


Figure 4: Estimated value function for different methods in `Maze2D` large using datasets collected using policies of different values (rows). Values are normalized.

3.2 Value Function Approximation

Distance monotonic representations (Definition 3.1) allow us to recast the original goal-conditioned RL problem as a distance one: similarly to the case of isometric maps, we can approximate the optimal value function using distances in the learned latent representation:

$$\tilde{V}(s) = \gamma^{d_Z(\phi_\theta(s), \phi_\theta(s_g))} r_g, \tag{2}$$

where s_g is the goal state and r_g is its associated sparse reward (only given at the goal state). This approximation would be identical to the true optimal value function only when the representation is an isometry of the graph.

However, distance monotonicity is enough to retain relative distances between states. Figure 4 shows the estimated value function of different algorithms for a maze-like problem, using offline datasets collected with policies of different values (low, medium and high). In such problems the value function should retain the topology of the maze and estimate the value of each position in the maze based on the distance to the goal within the maze.

Figure 4 highlights that a distance monotonic representation (MetricRL) is the only value function able to correctly estimate the low value of the bottom left corner of the maze when the goal is on the other branch of the maze, regardless of the quality of the policy used to collect the offline dataset. Equations 1 and 2 allow us to abstract the estimate of the value function from the distribution of the policy that collected the dataset. Note that, for more complex topologies (e.g., loop on the top-right corner), distance monotonicity is not equivalent to isometries. However, as we show in the next subsection, our approximation still allows us to estimate a provably optimal policy.

3.3 MetricRL

Enforcing distance monotonicity in the latent representation allows us to learn an *approximation* of the true value function, not fully recover it. However, when distance monotonicity is preserved a greedy policy built on this value function is equivalent to the optimal policy given some assumptions. Define the greedy policy according to the value function V as $\pi_g^V(s) = \arg\max_a V(T(s,a))$, note that in general the argmax results in a set of possible actions. We have the following:

Theorem 3.2. *If the MDP is deterministic, sparse, and goal-conditioned, then*

$$\pi_g^{V^*}(s) \subseteq \pi_g^{\tilde{V}}(s) \quad \forall s \in S$$

holds if ϕ is distance monotonic.

A proof can be found in Appendix A.3.

Remark 3.3. Intuitively, Theorem 3.2 states that every optimal action is also optimal in a DM representation but not necessarily the reverse. This is due to the non-uniqueness of the argmax operator. Consider the case of a representation ϕ that maps every state to the vertex of a simplex. Such a representation is DM and for every state the argmax results in every viable action as the distance between any two pairs of states is equal. It must be noted, however, that in a continuous high-dimensional Euclidean space, the probability of a state being equidistant to other (suboptimal) states is effectively zero. This is particularly true when the representation results from the optimization of the objective described in 1 as the negative term of the loss would not be minimized.

To learn such policy we take a three-stage approach, which we call *MetricRL*: (i) we learn a distance monotonic map, using the loss function of Equation 1; (ii) we define the value function using Equation 2; (iii) we estimate a policy using the following loss function:

$$\mathcal{L}_\pi = \mathbb{E}_{s,a\sim D}\left[-(\tilde{V}(s') - \tilde{V}(s))\log(\pi(a \mid s))\right]. \tag{3}$$

This optimization procedure for the policy is akin to the family of weighted maximum likelihood Nair et al. (2020) or weighted imitation learning Wang et al. (2018). A key difference is that the advantage is not exponentiated. Empirically this didn't have particular effects on the learning of the policy and allowed the removal of the additional hyper-parameter controlling the temperature of the exponential thus simplifying the training procedure. Using a value function from a distance monotonic representation ensures the term $\tilde{V}(s') - \tilde{V}(s)$ to be positive only for actions that bring the agent closer to the goal. This approach for the policy update is particularly suitable for offline RL: actions are sampled from the dataset which guarantees us to consider only in-distribution actions. Moreover, using the proposed distance monotonic representation instead of Temporal Difference methods for the critic solves the classic offline RL issue of out-of-distribution transitions (Levine et al., 2020).

3.4 Practical Implementation

Stabilizing the loss In practice, we take the logarithm of the contrastive term (second term) in Equation 1. We have found this formulation to stabilize the training, in particular for environments with longer episode horizons:

$$\mathcal{L}_\theta(D) = \mathbb{E}_D\left[(\|\phi_\theta(s') - \phi_\theta(s)\|_2 - 1)^2 - \lambda \log \|\phi_\theta(s'') - \phi_\theta(s)\|_2\right]. \tag{4}$$

Learning with images When learning with images, often the goal information is present in the image, e.g., the position of the green square in the grid experiment in Figure 5, which can break the connectivity assumption: two images with different goal positions are not connected by any path. This can be a problem in practice as the second term in Equation 1 can grow indefinitely when comparing representations of images with different goals. However, in the case of finite MDPs, we can easily recover it by introducing an additional *super*-state that connects every termination state together. We present the implementation details of such a super-state in Appendix A.4. The introduction of this super-state and the consequent connection of the environments with different goals leads to a modification of the learned representation. Figure 5 shows this learned representation. On the left is the distribution of the states for one single goal, while on the right is the distribution of all the goals connected by the super-state (red star). The solution to this representation is a radial distribution of the states connected in the middle by the super-state. All the states with the same goal (orange dots in the image) compose one ray of the overall distribution.

4 Results

We evaluate MetricRL against standard baselines in offline reinforcement learning across multiple environments. In all experiments we consider three types of offline datasets: a *low* dataset, often collected using a random

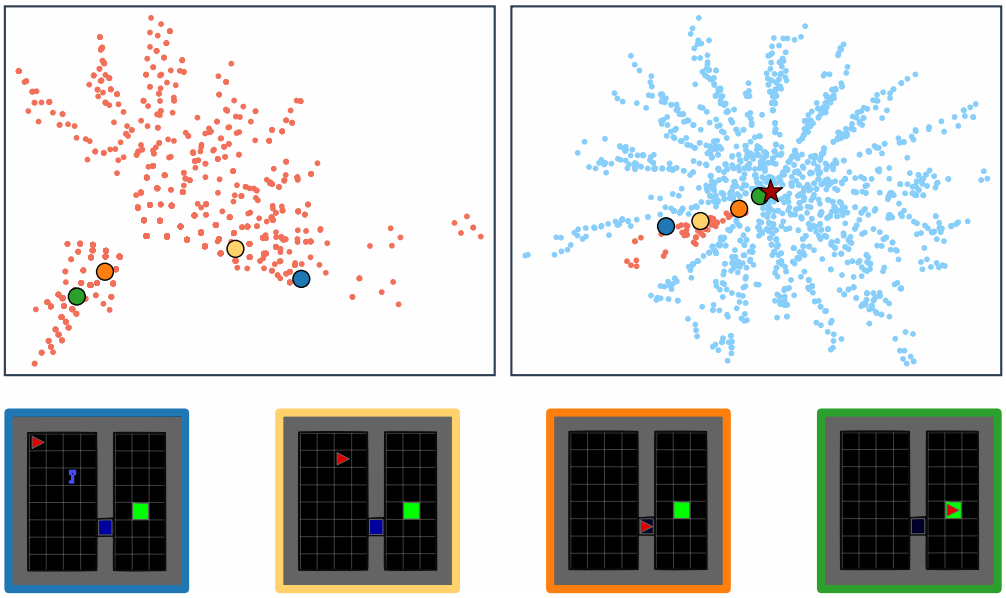


Figure 5: Visualization of the two-dimensional latent space of MetricRL in the `DoorKey` environment when considering state features (left) and image (right) observations. We observe that the addition of a super-state (red star on the right figure) for image observations results in a significant change in the structure of the embedded graph as each set of states with a different (and visible) goal gets separated (orange dots). Nonetheless, in both cases, the optimal policy still follows a geodesic in the graph: from the starting state (blue) the agent needs to pick up the key (yellow) to open the locked door (orange) and move to the goal state (green).

Algorithm 1 MetricRL.

Require: Initialize θ, ψ
Require: Offline dataset B, hyperparameters λ, η

repeat
  Sample batch $(s, a, s', s_g)_{\times B} \sim D$
  $s_r = \texttt{shuffle}(s)$ ▷ shuffle states in the batch
  $\mathcal{L}_\theta = (\|\phi_\theta(s) - \phi_\theta(s')\|_2 - 1)^2 - \lambda \log(\|\phi_\theta(s) - \phi_\theta(s_r)\|_2)$
  Update $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_\theta$
  $V = \|\phi_\theta(s) - \phi_\theta(s_g)\|_2$
  $V' = \|\phi_\theta(s') - \phi_\theta(s_g)\|_2$
  $\mathcal{L}_\pi = -(V' - V) \log(\pi_\psi(a|s))$
  Update $\psi \leftarrow \psi - \eta \nabla_\psi \mathcal{L}_\pi$
until convergence

policy or an untrained agent; a *medium* dataset, collected using the policy of an online RL agent during training or adding stochasticity to a fully trained agent, and a *high* dataset, collected using the policy of a fully-trained online RL agent.

For baselines, we consider:

- **CQL** (Kumar et al., 2020) introduces a conservative term in the estimation of the Q value. This term penalizes the highest values of the estimated Q function;
- **BCQ** (Fujimoto et al., 2019) perturbs the policy learned with a VAE with a DDPG term;
- **BEAR** (Kumar et al., 2019) constrains the learned policy to the behavioral one estimated with BC;

- **PLAS** (Zhou et al., 2021) trains the policy within the latent space of a conditional VAE trained on the Offline dataset;
- **IQL** (Kostrikov et al., 2021) avoids sampling out-of-distribution actions using a SARSA like critic update with quantile regression;
- **ContrastiveRL** (Eysenbach et al., 2022) approximates the value function of the policy that collected the dataset using contrastive learning. To adapt it to offline RL, the objective is coupled with a behavioral cloning term.
- **QRL** (Wang et al., 2023) estimates a quasimetric to approximate the value function using a contrastive learning formulation. The model is paired with a learned policy regularized with a behavioral cloning term to avoid out-of-distribution state-actions pairs in offline RL conditions.
- **GoFAR** (Ma et al., 2022) estimates an offline goal-conditioned RL policy by recasting it as a state occupancy matching problem.
- **HIQL** (Park et al., 2024) Reformulates IQL to an action-free hierarchical model.

For each model, we perform standard hyperparameter tuning or use the author's suggested hyperparameters (if available). The complete list of training hyperparameters is available in Appendix A.5, CQL, BCQ, BEAR, PLAS, IQL are implemented using Seno and Imai (2022). For the remaining models we use the author's provided code.

For environments, we consider:

- `Maze2D` (Fu et al., 2020): a navigation task within a two-dimensional maze, with continuous actions and Newtonian physics. We consider three sizes of the maze: *u-maze*, medium and large. For each size, we use a uniform random policy to collect the low dataset, a policy with Ornstein-Uhlenbeck noise (Uhlenbeck and Ornstein, 1930) to collect the medium dataset, and we use the Minari dataset provided in D4RL for the high dataset (Fu et al., 2020);
- `Reach` (Plappert et al., 2018): a manipulation task with a 7-DoF robot with continuous actions to reach a randomly-selected goal position in the workspace. To collect the datasets we employ a PPO agent trained on dense rewards along three different stages of training. For the low dataset, we use the policy of the randomly-initialized agent, for the medium dataset we use a PPO agent achieving half of the optimal expected reward, and for the high dataset we use the policy of the fully-converged agent;
- `Hypermaze`: a novel navigation task on a grid-like n-dimensional maze with discrete actions. To collect the offline datasets we employ a DQN agent trained online performing actions using an ϵ-greedy policy: for the low dataset we use purely uniformly random actions, for the medium dataset we sample random actions half of the time and optimal actions the other half, and for the high dataset we use the policy of the fully-trained agent;
- `Minigrid` (Chevalier-Boisvert et al., 2023): a navigation task on a grid-like 2D room with discrete actions. We restrict the action space of the agent to navigation actions and remove rotations. To collect the offline datasets we employ the same strategy as above, that is ϵ-greedy on a fully trained DQN agent. We consider 2 tasks, `Minigrid Empty` consists of an open grid with external walls as the only obstacles. `Minigrid DoorKey` is a three-step task where the agent must first pick-up a key, then open a door, then reach a goal position;

We present our main results in Figure 6[1] For a complete list of hyperparameters employed in the data collection procedure, please refer to Appendix A.5.

[1]We present in Appendix A.8 additional results. The conclusions remain the same for those additional environments.

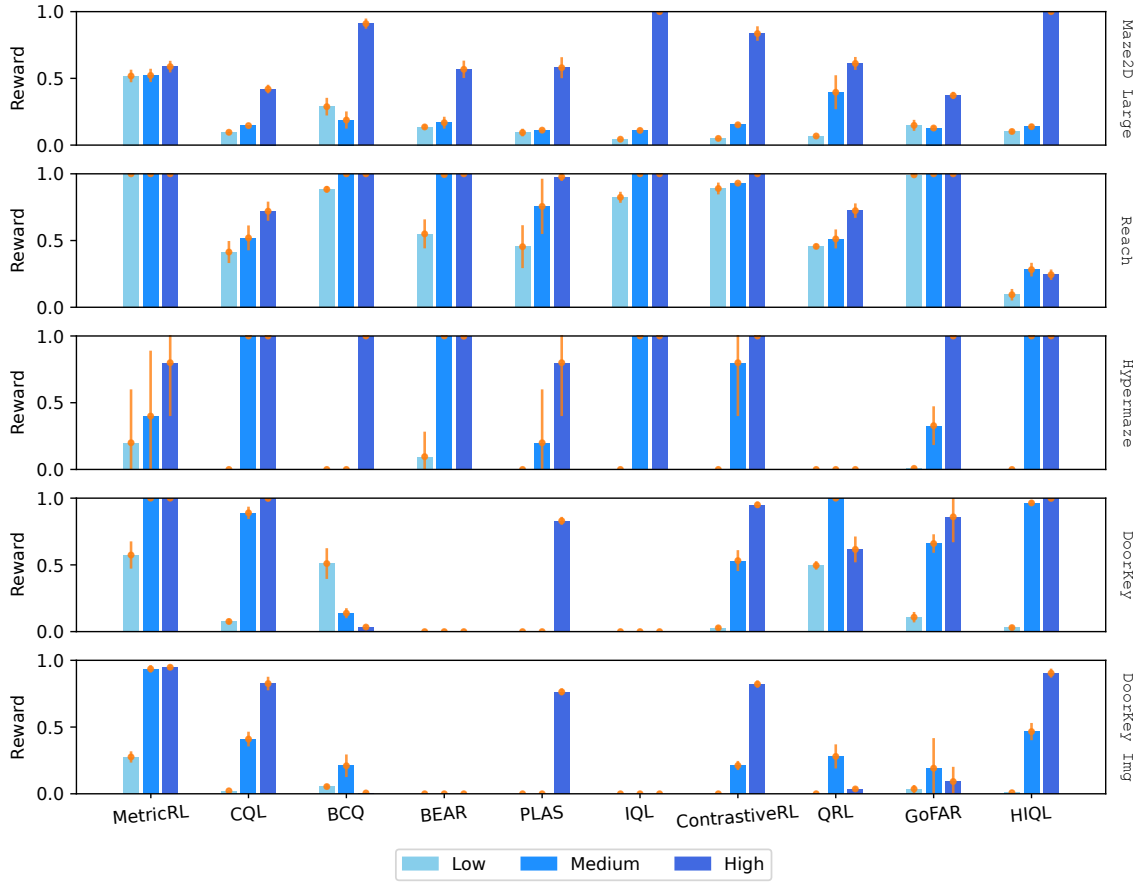


Figure 6: Average reward returns on offline RL tasks with different types of datasets. All results are averaged over 5 randomly-selected seeds. Higher is better. We present extra results in Appendix A.8. MetricRL is the only model able to consistently learn near-optimal behavior from sub-optimal datasets (low and medium), outperforming the baselines, while performing on par with optimal datasets (high).

MetricRL outperforms other methods in learning from sub-optimal datasets: The results highlight that MetricRL is the only model able to maintain a consistent level of performance, regardless of the type of dataset used for training. Additionally, MetricRL outperforms the other baseline methods when learning a policy from datasets collected using sub-optimal policies (*low* and *medium* datasets). In particular, for low datasets, MetricRL consistently outperforms all other baselines.

MetricRL provides more stable training across different data distributions in offline datasets: We conduct an additional experiment on the `Minigrid DoorKey` environment and consider a finer discretization of the distribution of rewards. We collect multiple datasets using an increasing value of ϵ for an ϵ-greedy optimal DQN agent. As shown in Figure 1, MetricRL (red line) requires datasets with a smaller average reward than the baselines to achieve optimal performance. Additionally, the results show that MetricRL does not suffer from out-of-distribution data when the dataset has a very narrow distribution (almost all or all optimal trajectories).

MetricRL outperforms QRL in tasks with large action spaces: MetricRL is able to estimate good policies in both discrete action spaces and continuous ones. In particular, it manages to converge in very high-dimensional action spaces like the `Hypermaze` where there are 81 possible actions with low datasets.

MetricRL can learn from sub-optimal datasets of images: To evaluate the performance of MetricRL when provided with high-dimensional observations (images) we reuse the MiniGrid `Empty` and `DoorKey` environments and introduce a *super*-state following the discussion in Section 3.4. For every model, we

introduce a CNN architecture that maps the images to a lower-dimensional representation. This highlights that MetricRL maintains its performance, and remains the only model able to learn optimal behavior from sub-optimal datasets. We present an additional result in the Appendix with Figure 14.

In Appendix A.6 we explore the sample efficiency of MetricRL in scenarios with large state spaces. The results show that our method significantly outperforms temporal-difference (TD) methods (e.g., DQN (Mnih et al., 2015)): to solve larger state space problems, we require a linear increase in the number of training iterations against the exponential increase of TD methods. In Appendix A.7, we show how MetricRL can be used in multi-goal tasks without modifications, considering changing multiple goals and discount factors at execution time.

4.1 Discussion

MetricRL recasts the computation of the value function as a problem of measuring distances in an appropriate learned metric space. To do so, it requires two additional assumptions on the MDPs it is applied to: the existence of inverse actions and the connectivity of the dataset used to learn the value function. As stated in Section 3.4, the connectivity assumption can be solved using "super-states" to join the states into a unique connected component. The existence of inverse actions, on the other hand, defines a trade-off. It limits the applicability of the proposed method to MDPs where the assumption is respected. On the other hand, it greatly simplifies the learning process by imposing a relevant bias on the representation. MetricRL, in fact, requires significantly fewer data points as it can interpolate missing transitions when the inverse transition is present in the data. This allows to estimate effective policies even in severely sub-optimal data conditions as shown in Section 4.

5 Related Work

Offline RL: The challenge of learning a policy from a static sub-optimal dataset of transitions and rewards has been extensively studied (Levine et al., 2020). The problem of exploration is not considered as it is assumed that the dataset given contains all the relevant information to estimate an optimal policy. Methods can be roughly divided into four main categories. The first one is constraining the learned policy to not deviate much from the policy that collected the dataset: Kumar et al. (2019) restrict policies to have the same support as the behavior policy rather than the policy itself; Zhou et al. (2021) implicitly constrain the policy by learning it using latent representations of actions with Gaussian prior (VAE), the constraint given by the KL divergence term; Siegel et al. (2020) learns the behavioral policy explicitly. The second category introduces a penalty in the reward function based on the uncertainty of the transitions or the reward function. This penalty has been defined as the uncertainty of the learned Q function, (Kidambi et al., 2020; Yu et al., 2020), or a measure of pessimism (regularization of the highest Q) (Kumar et al., 2020). A third family of methods instead uses model-based RL and explicitly computes a model of the environment that can be used in different forms to regularize the learned policy (Matsushima et al., 2020; Yu et al., 2021; Rigter et al., 2022; Fujimoto et al., 2019). The last category includes *in-sample* algorithms which restrict the learning of the policy only on data within the provided dataset and reweighted by an estimate of the advantage function. This family of methods has been referred to as weighted supervised learning (Wang et al., 2018) or maximum likelihood (Nair et al., 2020). It has been extended with different forms of regularization including expectile regression for the critic (Kostrikov et al., 2021), penalizing out-of-distribution actions in the critic (Chebotar et al., 2021), trust regions (Mao et al., 2023), goal relabeling (Yang et al., 2022) and Generative Adversarial Networks (Wang et al., 2024). The policy estimate of our proposed method falls in this last category, as can be seen in Equation 3. The use of Temporal Difference learning for the critic, however, subjects these methods to the problem of distribution shift in Offline RL. In this work, we propose a method that can estimate a high-return policy independently from the distribution of the data used even when the quality of the data decreases substantially.

Contrastive Learning: Representation learning techniques have been used to aid the RL problem. Several works have used contrastive learning models to speed up or improve the generalization of a classic RL algorithm, (Laskin et al., 2020; Oord et al., 2018; Anand et al., 2019; Stooke et al., 2021). Other applications include reward function estimation from demonstrations (Ma et al., 2022; Sermanet et al., 2018) or generating

intermediate goals for curriculum learning (Venkattaramanujam et al., 2019). In Eysenbach et al. (2022); Hatch et al. (2023) contrastive learning is used to estimate the discounted state occupancy measure which is equivalent to the Q function in some particular cases. This method however works only when the reward function can be expressed as a goal reaching density and doesn't estimate the optimal Q value but rather the Q value of a current policy, thus still requiring the concurrent learning of a policy in a classic RL fashion. Zhu et al. (2022) proposes the use of contrastive learning to build a representation of states where distances are correlated with reachability in terms of actions. After the representation is learned, they propose to explicitly build the graph of the offline dataset and apply value iteration to get an estimate of the value function. The policy can be obtained by applying Dijkstra on the learned graph. More similar to this work Wang et al. (2023); Yang et al. (2020) estimate the optimal value function rather than the policy one. In Yang et al. (2020) the authors use contrastive learning to find a representation where connections between states in action terms can be estimated in terms of Euclidean distances. The value function can then be recovered as the sum of the shortest path distances between the goal and the current state using depth-first search classical algorithms. Wang et al. (2023) learns a map between pairs of states and a quasimetric representing the estimated optimal value function of a goal-conditioned MDP. This is done by setting the distance between two consecutive states to be equal to the reward between them and the distance between random pairs of states to be maximized. The optimal value function can then be defined as the distance between each state and the goal state. While being more general, quasimetrics cannot capture the appropriate bias induced by the inverse actions assumption. We show empirically that our proposed method is more effective in estimating a policy when data is severely sub-optimal. In this paper, we describe a property of representations needed to ensure the optimality of the critic function and propose a simple method to recover such a representation for a particular class of MDPs. Moreover, differently from the works above, we study the effectiveness of this methodology in handling offline datasets collected by policies that are not necessarily optimal.

Other metrics: Other measures of state similarity from a control perspective have been explored before. Bisimulation metric defines a measure of similarity between states in terms of future transitions and rewards, (Ferns and Precup, 2014; Castro, 2020). These methods are theoretically grounded but particularly difficult to make them work in practice. This is especially true in the case of continuous spaces. Older work has explored different forms of value function approximation. By parametrically approximating the map between each state and its value, (Ormoneit and Sen, 2002) approximates a notion of similarity (in a value sense) between states with an appropriate kernel and rewrite the Bellman operator as a function of this kernel. This still needs to solve the optimization problem with value iteration techniques. Proto-value functions, (Parr et al., 2008), instead express the transition and reward functions as linear matrices, the value function problem has exact solutions and can be estimated with an appropriate kernel method at the cost of expressivity.

6 Conclusions

In this paper, we proposed MetricRL, a novel approximation method for the optimal value function of sparse, deterministic, goal-conditioned MDPs. MetricRL relies on learning a *distance monotonic* representation of the state space, allowing it to define a value function that is correlated with the distance of each state to the goal. We have proved that, when the representation is indeed *distance monotonic*, a greedy policy on this approximated value function is optimal for the class of MDPs stated above. Experimentally, we have shown that MetricRL outperforms prior offline RL methods in learning near-optimal behavior from severely sub-optimal datasets. For future work, we plan on generalizing the notion of *distance monotonicity* to quasimetrics (Durugkar et al., 2021; Durugkar, 2023), extend our method to stochastic MDPs and adapt it to online reinforcement learning problems.

References

[1] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

[2] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[3] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[5] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

[6] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[7] 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.

[8] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[9] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. In *Conference on Robot Learning*, pages 885–897. PMLR, 2020.

[10] Tianyu Shi, Dong Chen, Kaian Chen, and Zhaojian Li. Offline reinforcement learning for autonomous driving with safety and exploration enhancement. *arXiv preprint arXiv:2110.07067*, 2021.

[11] Nico Gürtler, Sebastian Blaes, Pavel Kolev, Felix Widmaier, Manuel Wüthrich, Stefan Bauer, Bernhard Schölkopf, and Georg Martius. Benchmarking offline reinforcement learning on real-robot hardware. *arXiv preprint arXiv:2307.15690*, 2023.

[12] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 35603–35620, 2022.

[13] Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. *arXiv preprint arXiv:2304.01203*, 2023.

[14] Yecheng Jason Ma, Jason Yan, Dinesh Jayaraman, and Osbert Bastani. How far i'll go: Offline goal-conditioned reinforcement learning vis f-advantage regression. In *Workshop on Learning from Diverse, Offline Data*, 2022.

[15] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[16] Mukund Balasubramanian and Eric L Schwartz. The isomap algorithm and topological stability. *Science*, 295(5552):7–7, 2002.

[17] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[18] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

[19] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[20] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

[21] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

[22] Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[23] Ge Yang, Amy Zhang, Ari Morcos, Joelle Pineau, Pieter Abbeel, and Roberto Calandra. Plan2vec: Unsupervised representation learning by latent plans. In *Learning for Dynamics and Control*, pages 935–946. PMLR, 2020.

[24] Dmitri Burago, Yuri Burago, Sergei Ivanov, et al. *A course in metric geometry*, volume 33. American Mathematical Society Providence, 2001.

[25] Jean Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52:46–52, 1985.

[26] Jiří Matoušek. Embedding finite metric spaces into normed spaces. In *Lectures on Discrete Geometry*, pages 355–400. Springer, 2002.

[27] Esra Ataer-Cansizoglu, Murat Akcakaya, Umut Orhan, and Deniz Erdogmus. Manifold learning by preserving distance orders. *Pattern recognition letters*, 38:120–131, 2014.

[28] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[29] Qing Wang, Jiechao Xiong, Lei Han, Han Liu, Tong Zhang, et al. Exponentially weighted imitation learning for batched historical data. *Advances in Neural Information Processing Systems*, 31, 2018.

[30] Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. In *Conference on Robot Learning*, pages 1719–1735. PMLR, 2021.

[31] Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36, 2024.

[32] Takuma Seno and Michita Imai. d3rlpy: An offline deep reinforcement learning library. *Journal of Machine Learning Research*, 23(315):1–20, 2022. URL `http://jmlr.org/papers/v23/22-0017.html`.

[33] 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.

[34] George E Uhlenbeck and Leonard S Ornstein. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

[35] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[37] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[38] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

[39] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

[40] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *arXiv preprint arXiv:2006.03647*, 2020.

[41] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

[42] Marc Rigter, Bruno Lacerda, and Nick Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *Advances in neural information processing systems*, 35:16082–16097, 2022.

[43] Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jacob Varley, Alex Irpan, Benjamin Eysenbach, Ryan C Julian, Chelsea Finn, et al. Actionable models: Unsupervised offline reinforcement learning of robotic skills. In *International Conference on Machine Learning*, pages 1518–1528. PMLR, 2021.

[44] Yixiu Mao, Hongchang Zhang, Chen Chen, Yi Xu, and Xiangyang Ji. Supported trust region optimization for offline reinforcement learning. In *International Conference on Machine Learning*, pages 23829–23851. PMLR, 2023.

[45] Rui Yang, Yiming Lu, Wenzhe Li, Hao Sun, Meng Fang, Yali Du, Xiu Li, Lei Han, and Chongjie Zhang. Rethinking goal-conditioned supervised learning and its connection to offline rl. *International Conference on Learning Representations*, 2022.

[46] Mianchu Wang, Rui Yang, Xi Chen, Hao Sun, Meng Fang, and Giovanni Montana. Goplan: Goal-conditioned offline reinforcement learning by planning with learned models. *Transactions on Machine Learning Research*, 2024.

[47] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[48] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[49] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in atari. *Advances in neural information processing systems*, 32, 2019.

[50] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pages 9870–9879. PMLR, 2021.

[51] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

[52] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[53] Srinivas Venkattaramanujam, Eric Crawford, Thang Doan, and Doina Precup. Self-supervised learning of distance functions for goal-conditioned reinforcement learning. *arXiv preprint arXiv:1907.02998*, 2019.

[54] Kyle Beltran Hatch, Benjamin Eysenbach, Rafael Rafailov, Tianhe Yu, Ruslan Salakhutdinov, Sergey Levine, and Chelsea Finn. Contrastive example-based control. In *Learning for Dynamics and Control Conference*, pages 155–169. PMLR, 2023.

[55] Deyao Zhu, Li Erran Li, and Mohamed Elhoseiny. Value memory graph: A graph-structured world model for offline reinforcement learning. *arXiv preprint arXiv:2206.04384*, 2022.

[56] Norman Ferns and Doina Precup. Bisimulation metrics are optimal value functions. In *UAI*, pages 210–219, 2014.

[57] Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10069–10076, 2020.

[58] Dirk Ormoneit and Śaunak Sen. Kernel-based reinforcement learning. *Machine learning*, 49:161–178, 2002.

[59] Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 752–759, 2008.

[60] Ishan Durugkar, Mauricio Tec, Scott Niekum, and Peter Stone. Adversarial intrinsic motivation for reinforcement learning. *Advances in Neural Information Processing Systems*, 34:8622–8636, 2021.

[61] Ishan Durugkar. *Estimation and control of visitation distributions for reinforcement learning.* PhD thesis, 2023.

[62] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping : Reinforcement learning withless data and less real. 1993. URL `https://api.semanticscholar.org/CorpusID:264654473`.

A Appendix

A.1 Sparse Goal-Conditioned Value Functions in Isometric Spaces

Recall the definition of the optimal value function described above as the maximum over the possible policies of the expectation of the discounted cumulative reward: $V^*(s, s_g) = \max_\pi \mathbb{E}_\pi \left[\sum_t \gamma^t r_t | s_0 = s, a = \pi(s, s_g)\right]$.

In the particular case of a deterministic MDP with a sparse goal-conditioned reward function, the value function simplifies as there is only one state for which the reward is non-zero (the goal state). Moreover, in the optimal value function, the estimated discounted cumulative reward becomes equivalent to the reward at the goal discounted by $\gamma^{d_G(s,s_g)}$, where $d_G(s, s_g)$ is the geodesic. In this setting, the optimal value function estimation can be reduced to a shortest path estimation problem.

Assuming ϕ is an isometry, $d_G(s, s_g) = d_Z(\phi(s), \phi(s'))$ resulting in: $V^*(s, s_g) = \gamma^{d_Z(\phi(s),\phi(s'))} r_g$.

A.2 Distance Monotonicity Measure

We provide a more formal intuition on the increase in distance monotonicity of a representation that minimizes Equation 1. As such, consider the following modification of the objective:

$$\min_\theta \mathbb{E}_D \left[-\left\|\phi_\theta(s'') - \phi_\theta(s)\right\|_2\right] \tag{5}$$

$$\text{subject to: } \left\|\phi_\theta(s') - \phi_\theta(s)\right\|_2 = 1. \tag{6}$$

This can be seen as the second term of the loss in Equation 1 with the first term as an explicit constraint. As defined before, the distance between any two points, s_i, s_j, is the length of the geodesic on the graph defined as the offline dataset. This is equivalent to the sum of the intermediate steps within the geodesic path. Using the constraint in Equation A.2 we can use triangular inequality to bound the distance between points in the learned representation:

$$d_Z(z_i, z_j) = \left\|\phi_\theta(s_j) - \phi_\theta(s_i)\right\|_2 \in \left[0, d_S(s_i, s_j)\right]. \tag{7}$$

The representation's ϕ_θ distance monotonicity can be measured as the ratio of triplets that respect the definition 3.1. That is, if $d_S(s_1, s_3) \leq d_S(s_2, s_3)$ then $d_Z(z_1, z_3) \leq d_Z(z_2, z_3)$, where $z = \phi(s)$. The distance monotonicity of ϕ increases for each triplet for which this becomes true. Graphically this can be seen in Figure 7. As the distance $d_Z(z_1, z_3)$ is bounded, the distance monotonicity of ϕ increases when $d_Z(z_2, z_3) \in [d_Z(z_1, z_3), d_S(s_2, s_3)]$ (or equivalently z_2 goes outside the blue circumference in the figure). As such, distance monotonicity increases by stretching the distance between any two states. This is equivalent to the objective described in Equation A.2.

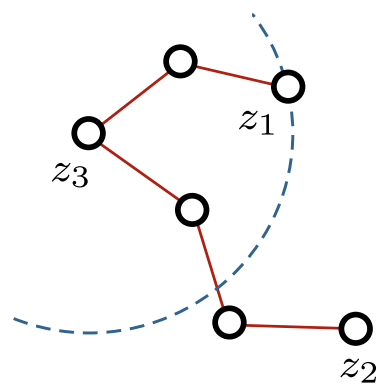


Figure 7: The optimization in A.2 increases the distance between the representations. As long as z_2 is outside of the circumference, the triplet $s_{\{1,2,3\}}$ is distance monotonic.

A.2.1 Experiment on Distance Monotonicity Measure

We tested quantitatively the effects of minimizing the loss defined in Equation 1 and the increase in distance monotonicity of the learned representation. To do so we used the environment defined in `Maze2D` large with the high dataset. Figure 3 (blue curve) shows the increase in distance monotonicity of a representation throughout the learning process. We computed the measure as follows:

- Before starting the training we compute a discretization of the positions of the maze and an adjacency matrix based on whether two positions are connected or not. This effectively defines a graph of the states of the maze.

- During every epoch of training we sample 1000 triplets of these nodes (s_1, s_2, s_3) at random and compute the distances on the graph between the two pairs using breath-first search, i.e., $d_G(s_1, s_3)$ and $d_G(s_2, s_3)$.
- Concurrently we map these three points in Z using the representation that's being learned and compute the distances using a Euclidean norm, i.e., $d_Z(z_1, z_3)$ and $d_Z(z_2, z_3)$.
- We then compute the distance monotonicity (DM) ratio as the number of triplets among these 1000 for which if $d_G(s_1, s_3) \leq d_G(s_2, s_3)$ then $d_Z(z_1, z_3) \leq d_Z(z_2, z_3)$ or if $d_G(s_2, s_3) \leq d_G(s_1, s_3)$ then $d_Z(z_2, z_3) \leq d_Z(z_1, z_3)$.
- The plot is also paired with the average reward a MetricRL agent achieves during training (orange curve).

More results on this are provided in Figure 15.

A.3 Proof of Theorem 3.2

Proof. Here we prove that, assuming a deterministic, sparse, goal-conditioned MDP, every optimal solution defined as the greedy policy on the optimal value function is contained in the greedy policy on a value function defined as in Equation 2.

To simplify the notation we will refer to $\phi(s)$ as z.

We start by rewriting the statement of the Theorem in the argmax form:

$$\pi_g^{V^*}(s) \subseteq \pi_g^{\tilde{V}}(s) \tag{8}$$

$$\implies \arg\max_a \left[V^*(s' = T(s,a))\right] \subseteq \arg\max_a \left[\tilde{V}(s' = T(s,a))\right], \tag{9}$$

here we make use of the fact that the MDP is deterministic and thus the tuple (s, a) injectively corresponds to a unique next state s' through the transition function T.

We can rewrite the left-hand side using Bellman and the right-hand side using the definition of the value function of Equation 2:

$$\arg\max_a \left[\max_{a'} \left(r(s', a') + \gamma V^*(s'')\right)\right] \subseteq \arg\max_a \left[\gamma^{d_Z(z', z_g)} r_g\right], \tag{10}$$

where again we denote $s'' = T(s', a')$. Assuming a deterministic sparse goal-conditioned MDP, the maximum of the left-hand side in Equation 10 is equal to the discount factor raised to the power of the minimal number of actions needed to reach the goal, $d_S(s', s_g)$, times the reward at the goal, r_g:

$$\max_{a'} \left(r(s', a') + \gamma V^*(s'')\right) = \gamma^{d_S(s', s_g)} r_g. \tag{11}$$

We can now rewrite Equation 10 as:

$$\arg\max_a \left[\gamma^{d_S(s', s_g)} r_g\right] \subseteq \arg\max_a \left[\gamma^{d_Z(z', z_g)} r_g\right], \tag{12}$$

and change the argmax to an argmin by dropping the discount factor ($\gamma < 1$) and the reward value:

$$\arg\min_a \left[d_S(s', s_g)\right] \subseteq \arg\min_a \left[d_Z(z', z_g)\right]. \tag{13}$$

We can now make use of the distance monotonicity assumption of ϕ. The set of actions a that minimizes the left-hand side of the above equation implies that for every other state $\hat{s}$ that can be reached from s we have:

$$d_S(s', s_g) \leq d_S(\hat{s}, s_g), \tag{14}$$

if ϕ is distance monotonic, this implies:

$$d_Z(z', z_g) \leq d_Z(\hat{z}, z_g), \tag{15}$$

for their corresponding latent representation and thus the actions that minimize $d_S(s', s_g)$ are contained in the actions that minimize $d_Z(z', z_g)$ resulting in:

$$a_{V^*} \subseteq a_{\tilde{V}} \quad \forall s \in S, \tag{16}$$

where we denoted $a_{\tilde{V}}$ as the set of actions that minimize $d_Z(z', z_g)$ and a_{V^*} as the set of actions that minimize $d_S(s', s_g)$. This, in turn, implies that the optimal policy is contained in $\pi_g^{\tilde{V}}(s)$ thus concluding the proof. □

A.4 Incorporating Super-States in the Dataset

Here we provide a short description on how to add super-states in the dataset to connect it. The steps can be summarized as follows:

- **Define super-states:** These can be defined synthetically by creating a state that is not present in the dataset. In the experiments, we always defined it as a vector of all zeros of the same dimensionality of the state space.
- **Add transitions:** The offline dataset can then be augmented with synthetic transitions. For each trajectory in the dataset, we can append a new transition from the terminating state to the meta state. The action connecting these states is not relevant as it will not be used in the representation. In the experiments, we set the action value to a random (but valid) value.

A.5 Experimental Details

For each experiment, we provide the results for 5 runs with different seeds. Each model is trained for 100 epochs consisting of 500 batches of 256 data points each. Every model is trained using the Adam optimizer with a learning rate of 10^-3. All experiments have been conducted using an NVIDIA RTX 3080 GPU accelerator.

Both the policy and the value function are parameterized using a simple Multi-Layer Perception architecture consisting of 3 layers with 64 neurons each and a ReLU activation function. In the case of *MetricRL*, the policy outputs the mean of a Gaussian distribution with fixed variance when the actions are continuous and the logits of a Categorical distribution when the actions are discrete. When the observations are images we use a CNN architecture consisting of 4 layers with 64 filters each of size 3 by 3 to preprocess the images.

A.5.1 Model Hyperparameters

MetricRL: The representation of the metric space has always dimension 128. The regularization term λ in Equation 1 is 1 and the variance of the policy when actions are continuous is 1.

CQL: $\gamma = 0.95$, we use a conservative weight of 5.0.

BCQ: $\gamma = 0.95$, action flexibility: 0.5, we sample 10 actions per step and use 2 critics.

BEAR: $\gamma = 0.95$, we use an adaptive α with an initial value of 0.001 and a threshold of 0.05, we use 2 critics modules and sample 10 actions per step.

PLAS: $\gamma = 0.95$, we use 2 critics modules.

IQL: $\gamma = 0.95$, we use 2 critics modules and an expectile value of 0.9.

ContrastiveRL: $\gamma = 0.95$, an offline regularization of 0.05 and a fixed variance for the policy of 1.

QRL: $\epsilon = 0.25$, initial $\lambda = 0.01$, offset softplus 500 and $\beta = 0.01$ and an offline regularization of 0.05.

GoFAR: $\gamma = 0.98$ and a discriminator gradient penalty of 0.01. For the f-divergence we use the χ^2-divergence.

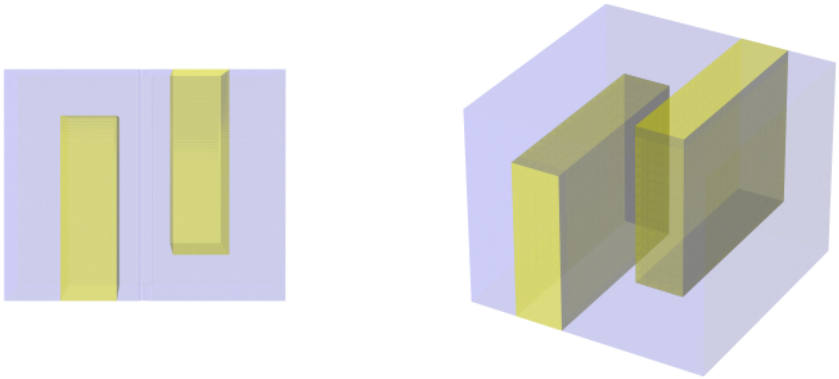


Figure 8: Wall positions (yellow) in the HyperMazes in 2D and 3D, in blue are free cells.

A.5.2 Evaluation Scenarios

`Maze2D`: The goal of the environment is to navigate an agent (actuated ball) to a target position inside a maze of varying complexity. The state space is 4-dimensional consisting of position and velocity in the plane, the action space is the torque applied to the ball in the two directions. The reward function is defined to be 1 when the agent reaches a position within 5 cm from the goal state and zero otherwise. Transitions take into consideration the momentum of the ball, frictions in the environment and collisions with the walls. For the low dataset we collect a dataset of 1000 trajectories of the agent performing uniform random actions. For the medium dataset we collect actions according to an Ornstein-Uhlenbeck process with parameters: $\theta = 0.1$ and $\sigma = 0.2$. For the high dataset we rely on the Minari dataset provided by Fu et al. [33].

`Reach`: The task consists of moving the end-effector of a simulated 7-DoF arm to a desired position in 3D. The state space consists of positions and velocities in 3D of the end effector and gripper of the robot and the goal refers to the desired position of the end-effector and zero velocity. Actions are translations of the end-effector. We first train a PPO agent online to solve the task and collect datasets at different stages of the training to define different qualities.

`Hypermaze`: Defines a generalization of the classic Grid Maze navigation task. The environment consists of a hypercube of n dimensions of m cells per dimension where every cell can either be empty or wall. The agent occupies one cell at a time if it is empty and can translate to adjacent cells if they are not walls. The positions of the wall are initialized in an S-like shape similar to Figure 8 when the hypermaze is defined in either 2 or 3 dimensions. The goal of the environment is to reach a goal placed on the other side of the maze.

For the results in Figure 6 we fix the maze to be 4 dimensional with 20 cells per dimension. We collect the datasets by first training a DQN agent online to solve the task and then collect 3 datasets using an ϵ-greedy policy with ϵ respectively of values 0.9, 0.5 and 0.1.

For the sample complexity analysis our method is coupled with a learned transition function to recover the Q estimate. We vary the dimensions (from 2 to 5) and the number of cells (from 10 to 50). Here the state space is defined as the position of the agent in the maze discretized into cells plus whether there are obstacles or not in the adjacent cells. To train the agents, random states and actions are sampled from the environment. A reward of 1 is given only if the agent steps into the goal state at the end of the maze.

`Minigrid`: We experiment with two variations of the minigrid environment. The first is the `Empty` environment where an agent translates freely within a grid-like environment. The state is described by the 2D position of the agent and the actions are the 4 possible translation directions. The reward is 1 once a randomly selected cell is reached and 0 otherwise. The `DoorKey` environment introduces bottlenecks in the MDP. A wall is

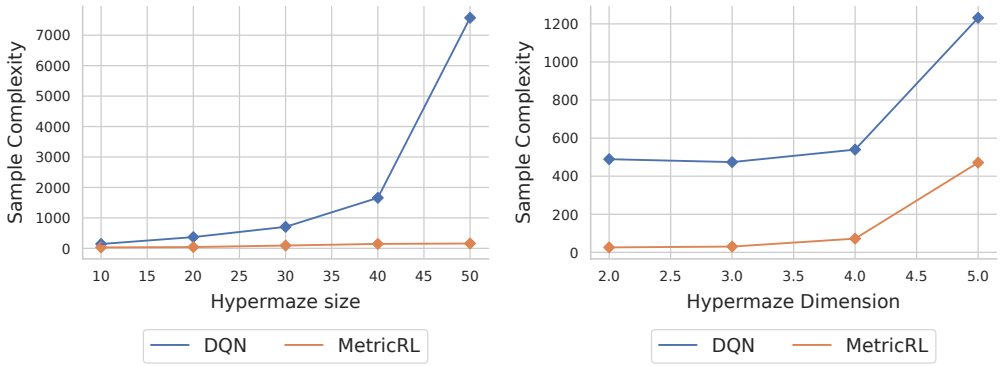


Figure 9: Number of updates required to reliably solve the `Hypermaze` environment with varying number of cells (left plot) and dimensions (right plot).

introduced in the center of the grid separating it into 2 rooms with a closed door in the middle. In the first room, a key is placed in a random position. The goal is to pick up the key, open the door and reach a goal cell in the other room. The state space is defined as the position in the grid of the agent plus the position of the key plus a binary value describing whether the door is open or not. Actions are translations in the grid plus a pick-up action that has an effect only when the agent is adjacent to the key plus an open door action that has an effect only when the agent has the key and is adjacent to the door. As before, The datasets are collected by training a DQN agent online to convergence and collecting the datasets using ϵ-greedy strategy with the three different values for ϵ. For the high-dimensional observations case we use images rendered by the environment as the states. These are 80 by 80 pixels with 3 color channels.

A.6 Sample-Efficiency

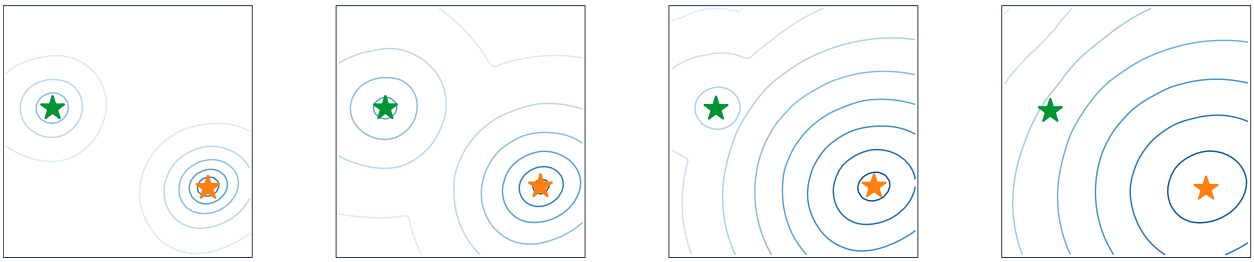
A main advantage of MetricRL stems from the nature of the loss function. Temporal difference methods (e.g., DQN) are known for their inefficiency when the time horizon grows considerably [62]. On the other hand, MetricRL mitigates this issue by using neural networks to learn a representation of a metric space.

To validate our hypothesis, we compare MetricRL against DQN [36] in the `Hypermaze` environment, considering a variable number of dimensions and cells. This allows us to control for both the dimensionality of the action space and the size of the state space of the underlying MDP.

In Figure 9 we present the number of iterations required for the two methods to solve the maze at least 25 times consecutively during training as a function of the size of the maze (state space). The results show that for MetricRL the number of iterations required to learn to perform the maze grows linearly with the size of the maze. However, for DQN the number of iterations rises exponentially.

A.7 Multi-goal Tasks

Another advantage of the proposed representation used in Equation 2 is that it does not depend explicitly on the goal. As long as goals are valid states within the training distribution, they can be arbitrarily used to estimate the value function. The explicit use of the discount factor allows for reshaping the value function to increase or decrease long-term reward delay. Moreover, with a slight modification, we can easily consider multiple competing goals. In this setting, the agent has to consider both the possible reward it can get in each goal state as well as its relative distance to each goal. As such, the discount factor of the MDP influences the optimal policy of the agent. For example when the agent has to navigate in a maze toward a door but there are multiple doors. In this case, the agent has to learn both how to navigate the maze as well as make the decision of which door to choose based on their reward and the length of the path.



(a) $\gamma = 0.9$ (b) $\gamma = 0.95$ (c) $\gamma = 0.99$ (d) $\gamma = 0.999$

Figure 10: Visualization of the gradient of the value function learned by MetricRL in an environment containing two fixed goals with different rewards (r_1, r_2), as a function of the discount factor: the green star has $r_1 = 0.7$ and the orange star has $r_2 = 1.0$.

With multiple rewards, the value function approximation can be reformulated by considering the maximum over the value function of all the possible goals (s_i, r_i):

$$\tilde{V}(s) = \max_i \{\gamma^{d_Z(\phi(s),\phi(s_i))} r_i\}. \tag{17}$$

A.8 Additional Results

In this section, we provide additional results on the experiments described.

`Maze2D`: We additionally provide the results for the u-maze and the medium maze described in Fu et al. [33], (Figures 11 and 12). Results confirm the findings described in the paper. MetricRL consistently outperforms the baselines in the case of low and medium datasets.

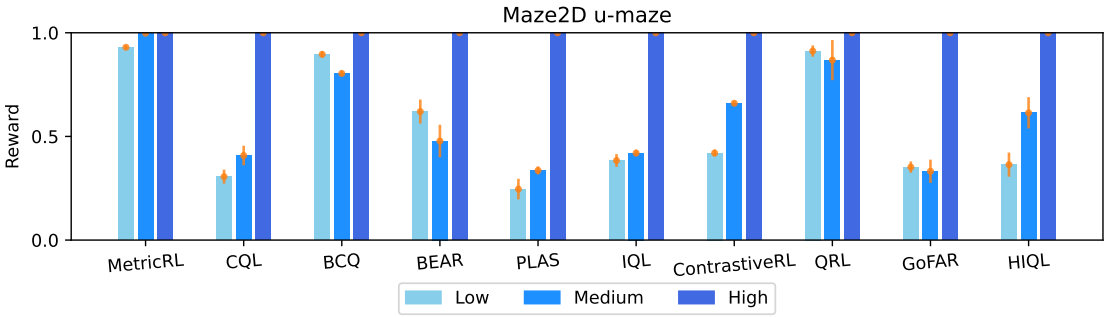


Figure 11: Average reward returns on `Maze2D` u-maze with different types of datasets. All results are averaged over 5 randomly-selected seeds. Higher is better.

`Minigrid`: We provide results of the Empty environment with states and images as input, (Figures 13 and 14).

Below (Figure 15) are additional results on the increase in distance monotonicity measure paired with an increase in reward for different mazes and datasets of `Maze2D`.

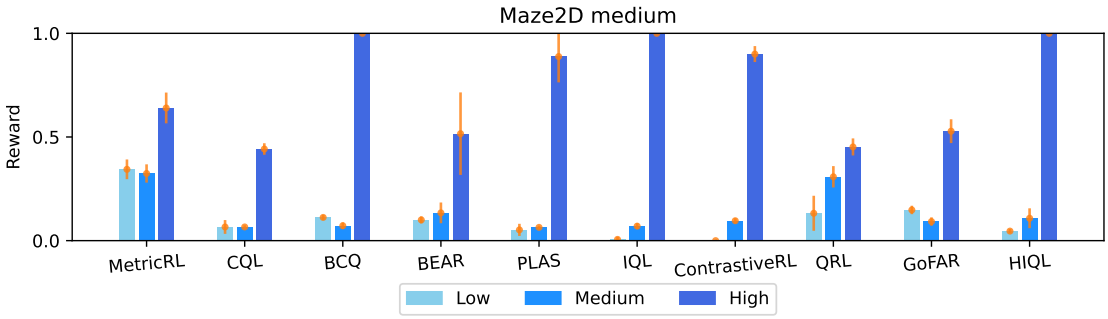


Figure 12: Average reward returns on `Maze2D` medium maze with different types of datasets. All results are averaged over 5 randomly-selected seeds. Higher is better.

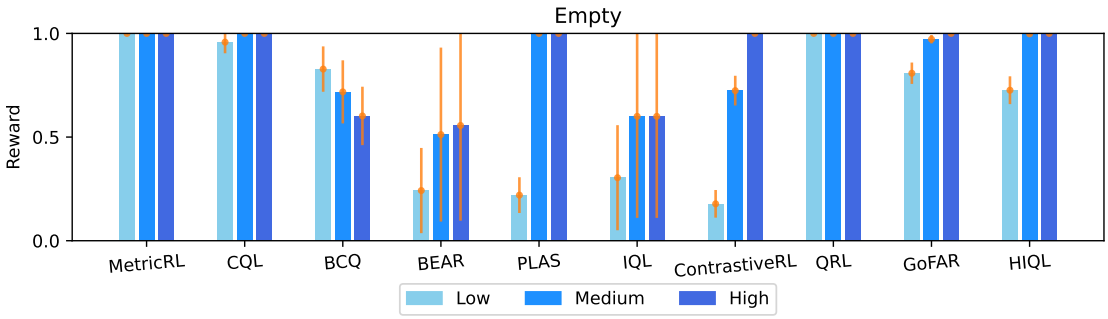


Figure 13: Average reward returns on `Minigrid Empty` with different types of datasets. All results are averaged over 5 randomly-selected seeds. Higher is better.

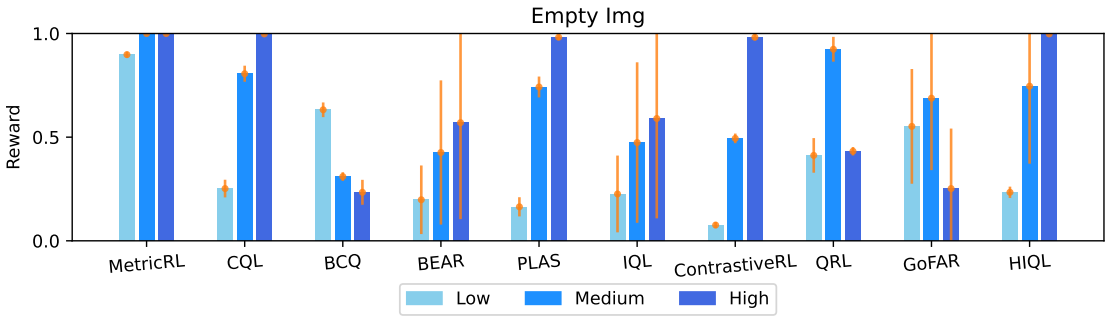


Figure 14: Average reward returns on `Minigrid Empty` with images with different types of datasets. All results are averaged over 5 randomly-selected seeds. Higher is better.

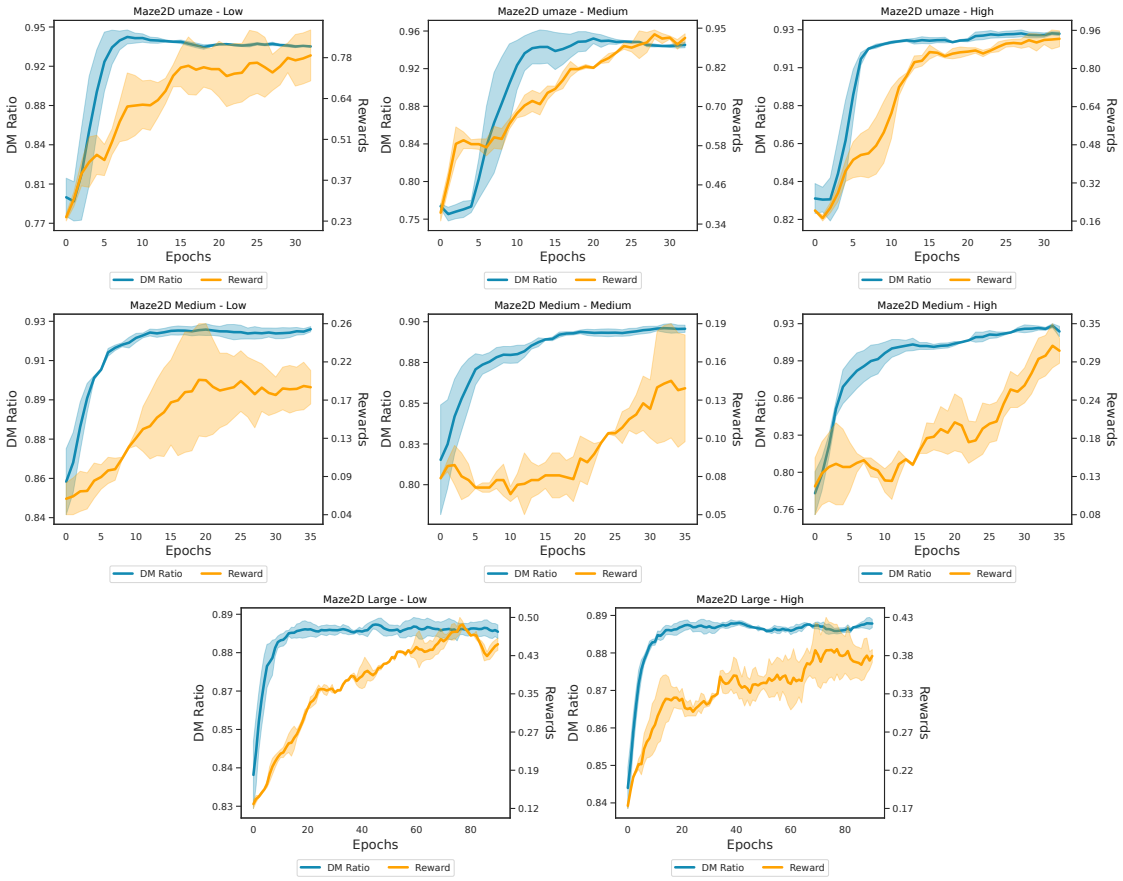


Figure 15: Distance monotonicity ratio compared with average reward on `Maze2D` environments with different mazes and datasets.