Learning Pseudometric-based Action Representations for Offline Reinforcement Learning

Pengjie Gu [1] Mengchen Zhao [*2] Chen Chen [2] Dong Li [2] Jianye Hao [3 2] Bo An [1]

Abstract

Offline reinforcement learning is a promising approach for practical applications since it does not require interactions with real-world environments. However, existing offline RL methods only work well in environments with continuous or small discrete action spaces. In environments with large and discrete action spaces, such as recommender systems and dialogue systems, the performance of existing methods decreases drastically because they suffer from inaccurate value estimation for a large proportion of out-of-distribution (o.o.d.) actions. While recent works have demonstrated that online RL benefits from incorporating semantic information in action representations, unfortunately, they fail to learn reasonable relative distances between action representations, which is key to offline RL to reduce the influence of o.o.d. actions. This paper proposes an action representation learning framework for offline RL based on a pseudometric, which measures both the behavioral relation and the data-distributional relation between actions. We provide theoretical analysis on the continuity of the expected Q-values and the offline policy improvement using the learned action representations. Experimental results show that our methods significantly improve the performance of two typical offline RL methods in environments with large and discrete action spaces.

1. Introduction

Reinforcement learning (RL) approaches have been applied successfully in many decision-making tasks. In a conventional setting, RL agents learn policies through an online fashion, where they collect trial-and-error experiences directly from the environment to improve the current policy. This can be done when the environment can be easily established, and the cost of deploying new policies online is low (e.g., computer games). However, many real-world scenarios allow only offline access to the environment due to cost or safety concerns (Thomas, 2015), meaning that the algorithm cannot access the environment during the training phase. This setting is known as offline RL, or batch RL (Levine et al., 2020). Despite the success that offline RL has achieved in robotic control tasks (Fujimoto et al., 2019b) and some Atari games (Gulcehre et al., 2021), the applicability of offline RL is still limited in many practical scenarios where the action space is extremely large, including recommender systems and dialogue systems.

Existing offline RL methods focus on the setting that the action space is continuous or discrete but with a few actions. To address the issue of overestimating the values of o.o.d. actions, they constrain the learned policy to stay close to the data-generating policies (Fujimoto et al., 2019b; Kumar et al., 2019; Wu et al., 2019; Kumar et al., 2020; Kostrikov et al., 2021; Zhou et al., 2020). However, the performance of these algorithms decreases drastically with the size of action space increasing. There are two main reasons. First, the value function hardly generalizes over the entire action space without proper action representations, especially when the semantics of actions are missing. Note that the success of offline RL in continuous control tasks benefits from the fact that actions can be naturally represented by their physical semantics (e.g., directions, forces), which also demonstrates the importance of action representations. Second, with the size of the action space increasing, the state-action pairs are more sparse to the entire state-action space, resulting in a large proportion of o.o.d. actions. Consequently, the learned policies would be overly restrictive – they are constrained to only select actions within the support of in-distribution actions, without considering a large number of other o.o.d. actions that might contain the optimal action (Zhou et al., 2020; Kumar et al., 2019).

It has been demonstrated that online RL benefits from using action representations to exploit underlying structures

[*] Corresponding author [1]School of Computer Science and Engineering, Nanyang Technological University, Singapore [2]Noah's Ark Lab, Huawei [3]College of Intelligence and Computing, Tianjin University. Correspondence to: Mengchen Zhao <zhaomengchen@huawei.com>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

of large action spaces. However, existing methods perform poorly in offline setting. The reason is that they fail to learn reasonable relative distances between actions, so they exacerbate the wrong value estimation of o.o.d. actions. In this paper, we propose a novel framework to learn pseudoMEtRic-based action representations for offLIne reinfOrcemeNt learning (MERLION) with large action spaces. In MERLION, the policy of an agent is trained with a latent, state-conditional action space, where the action representations are learned under a pseudometric that consists of both behavioral and data-distributional measurements. The behavioral metric aims to explicitly quantify the similarity of two actions in terms of the resulting transitions and rewards. The data-distributional metric aims to quantify the probabilities of different actions being executed by the behavior policies. Given the pseudometric, we propose an architecture of action encoder to learn action representations in a self-supervised way, where ℓ_1 distances between representations correspond to the defined metric.

The metric in MERLION is closely connected to the bisimulation metric, which has been demonstrated effective in learning state representations in an off-policy manner (Zhang et al., 2021). However, whether such a bisimulation-like metric is suitable for learning action representations in the offline setting remains unclear, because state and action are intrinsically different in the context of MDP. In this work, we show how to extend the bisimulation metric to learn action representations in offline setting. We also provide theoretical guarantees of value generalization and policy improvement for offline RL tasks using MERLION. Furthermore, in order to improve the learning efficiency, we propose a relation network architecture to balance in-distribution and o.o.d. actions. Experimental results on two simulated tasks and two real-world applications show that policies trained under the MERLION framework significantly outperform those trained using existing baselines in both offline and online settings. To our knowledge, we are the first to study offline RL with large and discrete action spaces and demonstrate the effectiveness of the bisimulation-like metric in action space, both empirically and theoretically.

2. Related Works

Offline RL Offline RL aims to learn policies from logged transition data without any active data collection (Levine et al., 2020). Recently, many works have been proposed in this area (Kumar et al., 2019; Wu et al., 2019; Kumar et al., 2020; Kostrikov et al., 2021; Fujimoto et al., 2019b). They empirically and theoretically reveal that the critical issue in offline RL is the overestimation error induced by o.o.d. actions. As a result, they propose a variety of behavioral regularizations in RL training that compel the learned policy to stay close to the offline data. These regularizations consist of incorporating some divergence regularization into the critic (Kumar et al., 2020; Nachum et al., 2019), policy divergence penalties (Kumar et al., 2019; Wu et al., 2019), and appropriate network initializations (Matsushima et al., 2021). These works show effective results in simulated tasks, like Mujoco (Todorov et al., 2012) and Atari games (Gulcehre et al., 2021). However, we empirically show that they fail in the real-world applications with large discrete actions, likely due to the difficulty of generalization over large action sets or the excessive restriction induced by a large proportion of o.o.d. actions.

Action Representations in RL Existing works on action representations focus on the online setting. For example, Lee et al. (2018) uses predefined action embeddings to represent discrete actions and utilize continuous policy gradients for policy optimization. Chandak et al. (2019) avoid predefined embeddings by linking action representations to state transitions. Tennenholtz & Mannor (2019) regard action trajectories as natural languages and thus learn action representations from trajectories of expert demonstrations. Wang et al. (2021) learn action representations that focus on accurate reconstruction of rewards and next observations. Though these works show good results in online RL tasks, we empirically show that their performance is extremely unstable when combined with offline RL algorithms, due to the lack of explicitly defined metric.

Some other works use action representations as pre-training behavioral priors or primitives (Zhou et al., 2020; Ajay et al., 2021; Singh et al., 2021). They train policies on latent action spaces which are learned from past successful trials and thus avoid a serious distributional shift in offline RL. However, these works heavily rely on the existence of optimal behaviors to learn useful priors. By contrast, our method learns from natural results of actions, e.g., transitions and rewards, therefore works well with more general datasets.

Metrics in RL A crucial principle to generalization in reinforcement learning is to assign similar predictions to similar states. A standard implementation is to use the similarity in an adaptive fashion and group states into clusters while preserving some desired properties. The fundamental assumption behind it is the existence of a metric characterizing the real-valued distance between states (Le Lan et al., 2021). A related concept is bisimulation metrics that measure how "behaviorally similar" states are (Ferns et al., 2004). Recently, Zhang et al. (2021) extend this concept by proposing a gradient-based method for learning a representation space with the properties of bisimulation metrics. Unfortunately, a recent work empirically shows that the bisimulation metric of states does not work well in the offline setting (Yang & Nachum, 2021). In offline RL, Dadashi et al. (2021) propose a metric to measure the closeness of state-action pairs to the support of logged transitions, but does so directly, without learning a representation. These works mainly focus on metrics on states or *joint* state-action pairs, while our work focuses on measuring the relations between actions *conditioned* on the given state.

3. Preliminaries

We start by introducing notation in this work. We consider the underlying environment as a Markov decision process (MDP) with a discrete action space, represented by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. Here, $\mathcal{S}$ is the state space. $\mathcal{A}$ is a finite set of actions, called the action set, and $|\mathcal{A}|$ denotes the size of the action set. $\mathcal{P}$ and $\mathcal{R}$ are the transition function and the reward function, respectively, indicating that when the agent takes the action $a \in \mathcal{A}$ under the state $s \in \mathcal{S}$, the probability of transitioning from state s to state $s' \in \mathcal{S}$ is $\mathcal{P}(s'|s, a)$, and there is a environmental reward $\mathcal{R}(s, a) \in \mathbb{R}$. The goal of the agent is to learn a good policy $a \sim \pi(s)$ that maximizes the expected cumulative discounted rewards: $\mathbb{E}_{\mathcal{P}}[\sum_{t=1}^{\infty}[\gamma^t \mathcal{R}(s_t, \pi(s_t)]]$. Our work focuses on the offline setting, in which the agent cannot collect new experience data and learns policy from a static dataset $\mathcal{D} = \{(s, a, s', r)\}$ generated by some other policy. We call the policy that generates $\mathcal{D}$ the *behavioral* policy and denote it as $\pi_\beta(a|s)$.

4. Offline RL with MERLION

In this section, we elaborate on MERLION, our proposed method for leveraging pseudometric-based action representations to address offline RL tasks with large discrete action spaces. We begin by describing the paradigm of how to train and execute policies according to the action representations. Next, we discuss which properties of action representations are crucial to the performance of this framework and define a pseudometric function to measure these properties. Then, we propose an action encoder architecture to learn the pseudometric-based action representations from the offline dataset. Finally, we give a theoretical analysis of the generalization ability of the value function and the offline policy improvement using MERLION.

Overview. We first introduce a two-phase paradigm for solving such tasks. In the first phase, we train an action eocoder ϕ for generating action representations and then convert the original dataset to a new dataset $\mathcal{D}_e = \{(s, e, s', r)\}$, where $e = \phi(a; s)$ is the state-conditional action representation and $e \in \mathcal{E}$. Note that we introduce how to train ϕ in next subsections. In the second phase, an arbitrary offline RL algorithm would be applied to learn an internal policy $\pi_i(\hat{e}|s)$ from $\mathcal{D}_e$. $\pi_i(\hat{e}|s)$ would provide a latent action $\hat{e} \in \mathcal{E}$ for a given state, but $\hat{e}$ would likely not be a valid action, i.e., it does not equal any action representation e of $a \in \mathcal{A}$. Therefore, we need to map from $\hat{e}$ to an element in $\mathcal{A}$. Here, we adopt a simple nearest neighbor lookup $g(\hat{e})$ introduced in (Lee et al., 2018):

$$g(\hat{e}) = \arg\min_{a \in \mathcal{A}} \|\hat{e} - \phi(a; s)\|_1 \tag{1}$$

where g is a mapping function from the continuous representation space to the discrete action set. It returns the original action whose representation is the closest to $\hat{e}$ by ℓ_1 distance. Therefore, the overall policy $\pi_o = g(\pi_i(e|s))$. Its scheme is also described by Fig.1 (a) and Alg.1 in the appendix.

4.1. Pseudometric Function for Measuring Relations between Actions

In this section, we discuss which properties are important to our action representations and introduce a pseudometric function to measure these properties.

How to formulate the structure of action representations is crucial to the performance of our framework. The agent acting in the representation space would generalize the information of an action to other actions with similar representations. However, if actions with similar representations have different environmental effects, this generalization might result in poor performance. Furthermore, existing offline RL algorithms usually adopt various approaches for behavioral regularizations to avoid serious overestimation errors; they enforce the learned policy to choose actions close to the in-distribution actions. Therefore, different action spaces induce distinct results of the behavioral regularizations, and an inappropriate representation space would destroy the behavioral regularizations. For example, if one o.o.d. action, which induces distinct transitions and rewards, is close to another in-distribution action in the representation space, a *false* result of behavioral regularization would happen. To this end, we expect that the learned action representations' relative distances reflect two major relations between any two discrete actions:

(i) **The Behavioral Relation**: The distance between any two actions in the representation space should reflect the difference between their induced transitions and rewards. Intuitively, if two actions result in the same transitions and rewards at all states, they are identical to the associated MDP. The action representations following this relation should help to improve the generalization of value function approximators across the action space.

(ii) **The Data-distributional Relation**: The distance between any actions in the representation space should reflect whether they are in the same distribution of the experience dataset. If they are not in the same distribution, there is a penalty distance between them. This helps the behavioral regularized policy select actions close to the in-distribution actions so that to avoid serious overestimation errors.

In order to formalize the first relation, we start by defining a distance function $d : \mathcal{A} \times \mathcal{A} \mapsto \mathbb{R}_{\geq 0}$ measuring the

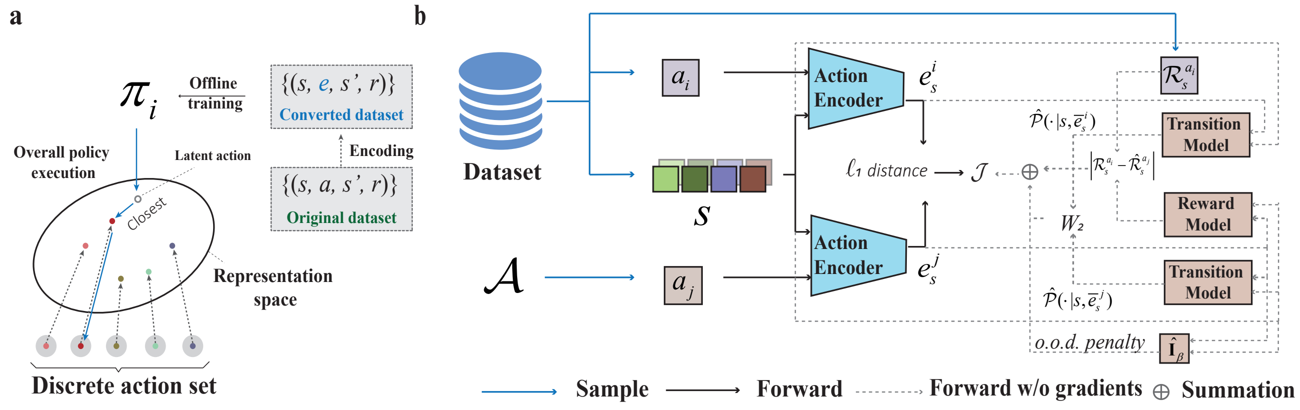

Figure 1. **MERLION framework. (a)** The overall policy scheme. **(b)** The architecture of pseudometric-based action representation learning. The action encoder aims to generate the action representations, where the ℓ_1 distance between any two representations equals the pseudometric distance between their original actions. The action embeddings e_s^i and e_s^j are both conditioned on the same state s. During training, $(s, a_i, \mathcal{R}_s^{a_i})$ is sampled from the dataset, and a_j is uniformly sampled from the finite action set $\mathcal{A}$. The reward of (s, a_j) and the transitions of (s, a_j) and (s, a_i) are estimated by corresponding models. $\hat{I}_\beta$ is a function to estimate whether a_j is an in-distribution action and thus determine whether there is an o.o.d. penalty on the pseudometric distance. The objective $\mathcal{J}$ is the mean square error between $\|e_s^i - e_s^j\|_1$ and the summation of the reward distance, the discounted transition distribution distance, and the o.o.d. penalty.

behavioral similarity between two actions.

Definition 4.1 (Behavioral Metric of Actions)**.** [1]Given an MDP $\mathcal{M}$, a behavioral metric of actions is a function $d : \mathcal{A} \times \mathcal{A} \mapsto \mathbb{R}_{\geq 0}$ such that:

$$d(a_i, a_j) = \mathbb{E}_{s \in \mathcal{S}} \left[|\mathcal{R}_s^{a_i} - \mathcal{R}_s^{a_j}| + \gamma \cdot W_2(\mathcal{P}_s^{a_i}, \mathcal{P}_s^{a_j}) \right] \quad (2)$$

where W_2 is the 2^{nd} Wasserstein distance between two distributions.

This definition is similar with the form of the state bisimulation (Zhang et al., 2021). However, "state" and "action" are intrinsically different in the context of MDP. Simply extending the bisimulation definition of state to action in our setting fails for two reasons. First, one action could exhibit distinct behavioral semantics at different states. If we directly estimate the behavioral relation between two actions by the expectation over $\mathcal{S}$-as is shown in equation 2-the result would suffer from very large variance. Second, o.o.d. actions influence both the action representation learning and the policy learning significantly, especially in the setting of large action spaces. Therefore, we need to balance in-distribution actions and o.o.d. actions when defining the metric. To tackle these challenges, we first extend the definition 4.1 to a *state-conditional* version to reduce the variance of results and then propose to use a distributional penalty to reduce the influence of o.o.d. actions. We first give the conditional definition as follows:

Definition 4.2 (State-conditional Behavioral Metric of Actions)**.** Given an MDP $\mathcal{M}$ and a state s, a state-conditional behavioral metric of actions is a function $d : \mathcal{S} \times \mathcal{A} \times \mathcal{A} \mapsto \mathbb{R}_{\geq 0}$ such that:

$$d(a_i, a_j|s) = |\mathcal{R}_s^{a_i} - \mathcal{R}_s^{a_j}| + \gamma \cdot W_2(\mathcal{P}_s^{a_i}, \mathcal{P}_s^{a_j}) \quad (3)$$

This definition avoids estimating the expectation over $\mathcal{S}$ and brings a higher representation capability: It allows the actions' relations to be changed under different states. E.g., in a maze environment, the actions "move left" and "move right" are behaviorally equivalent at the corner where they both induce a collision, but different at the position where they induce movements with opposite directions.

Data-distribution Detection. In the offline setting, the estimation about relations of o.o.d. actions might be inaccurate. This mismatch would allow those o.o.d. actions to be too close to the in-distribution actions, which have distinct effects from them. So we also add a penalty distance between the o.o.d. and the in-distribution actions, forcing the behavioral regularized policy tends to cautiously choose o.o.d. actions. Thus, we modify the pseudometric function d (Eq.3) to reflect the data-distributional relation between actions:

$$\begin{aligned} d(a_i, a_j|s) =& |\mathcal{R}_s^{a_i} - \mathcal{R}_s^{a_j}| + \gamma \cdot W_2(\mathcal{P}_s^{a_i}, \mathcal{P}_s^{a_j}) \\ &+ p \cdot I_\beta(a_i, a_j|s) \end{aligned} \quad (4)$$

where $p > 0$ is a penalty coefficient, $I_\beta(a_i, a_j|s)$ is a function indicating the relation of actions: $I_\beta(a_i, a_j|s) = 0$ if a_i and a_j are from two data distributions, otherwise $I_\beta(a_i, a_j|s) = 1$.

[1] Note that d is a pseudometric, meaning that it allows the distance between two different actions to be 0.

4.2. Learning Pseudometric-based Action Representations

This section introduces how to learn action representations complying with the pseudometric d (Eq.4). One key challenge is that the data of in-distribution actions are poor due to the large proportion of o.o.d. actions. To overcome this, we propose to use a relation network architecture (Santoro et al., 2017), as is illustrated in Fig.1 (b), to balance in-distribution and o.o.d. actions.

In detail, we train an action encoder $\phi : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{E}$ for generating action representations that possess the desired property $d(a_i, a_j|s) := \|\phi(a_i; s) - \phi(a_j; s)\|_1$. It means that the ℓ_1 distance between any two actions in the learned representation space equals their pseudometric distance. Concretely, the learning objective can be written as follows:

$$J(\phi) = \left(\|e_s^i - e_s^j\|_1 - d(a_i, a_j|s)\right)^2 \tag{5}$$

where $e_s^i = \phi(a_i; s), e_s^j = \phi(a_j; s)$. In practice, this objective requires sampling state-action pairs with the same state s but different actions a_i and a_j. However, it is challenging to search for sufficient such pairs from the dataset. Thus, we modify the objective by using the estimated samples:

$$J(\phi) = \mathbb{E}_{s,a_i,\mathcal{R}_s^{a_i}\sim\mathcal{D},a_j\sim\mathcal{A}}\left(\|e_s^i - e_s^j\|_1 - \hat{d}(a_i, a_j|s)\right)^2 \tag{6}$$

where

$$\begin{aligned}\hat{d}(a_i, a_j|s) =& |\mathcal{R}_s^{a_i} - \hat{\mathcal{R}}(s, \bar{e}_s^j)| \\ &+ \gamma \cdot W_2(\hat{\mathcal{P}}(\cdot|s, \bar{e}_s^i), \hat{\mathcal{P}}(\cdot|s, \bar{e}_s^j)) \\ &+ p \cdot \hat{I}_\beta(a_j|s)\end{aligned} \tag{7}$$

Specifically, $\hat{d}(a_i, a_j|s)$ is an estimation of $d(a_i, a_j|s)$. $\bar{e}_s$ denotes $\phi(a; s)$ with stop gradients, $\hat{\mathcal{R}}$ and $\hat{\mathcal{P}}$ is the reward model and transition model, which have their own training steps. $\hat{I}_\beta(a_j|s)$ is also a trainable model to predict whether a_j is an o.o.d. action. $\hat{I}_\beta(a_j|s) = 1$ if a_j is out of distribution, otherwise $\hat{I}_\beta(a_j|s) = 0$. In practice, we follow the prior work to derive $\hat{I}_\beta(a|s)$ (Fujimoto et al., 2019a): We train a model $G(a|s) \approx \pi_\beta(a|s)$ to predict the probabilities of every action in the dataset and then scale all probabilities by the maximum probability. a_j would be predicted as the o.o.d. action if the relative probability is below a threshold constant τ. This procedure can be summarized as $\hat{I}_\beta(a|s) = \frac{G(a|s)}{\max_{\hat{a}\in\mathcal{A}} G(\hat{a}|s)} \leq \tau$.

During training, we sample the tuple $(s, a_i, \mathcal{R}_s^{a_i})$ from $\mathcal{D}$ and directly sample the other action a_j from the finite action set $\mathcal{A}$. Then, other lacked data would be estimated by the trained models. To a certain extent, this estimation operation would reduce the accuracy of the learned action representations, but our empirical study shows that it is sufficient to derive a latent space with effective representative ability in offline RL tasks. Further details of the action encoder and all models are described in Alg.2 in the appendix.

Theoretical Analysis. We first theoretically analyze the generalization ability of the value function using our action representations. We prove that the value function of any given policy π is Lipschitz with respect to our proposed pseudometric function d.

Theorem 4.3 (Q^π is Lipschiz with respect to d)**.** *Given a policy π, let Q^π be the value function for a given discount factor γ. Q^π is Lipschitz continuous with respect to d with a Lipschitz constant $\frac{1}{1-\gamma}$*

$$|Q^\pi(s, a_i) - Q^\pi(s, a_j)| \leq \frac{1}{1-\gamma} d(a_i, a_j|s) \tag{8}$$

Proof in appendix. This theorem means that, the closer two actions are in terms of d, the more likely they are to share the similar value. If we explicitly force the distance between action representations to comply with d, the value function of the learned policy would be Lipschitz continuous in the action representation space. This continuity brings an effective generalization capability to the value function, and thus, intuitively, reduces the estimation errors of o.o.d. actions (Le Lan et al., 2021). On the other hand, when the policy is regularized to select actions close to the in-distribution actions in our representation space, it would choose similar actions in terms of the long-term return, avoiding selecting actions with distinct effects. This conculsion can be further represented as: If one of a_i and a_j is the o.o.d. action but the other is not, $|Q^\pi(s, a_i) - Q^\pi(s, a_j)| \leq \frac{1}{1-\gamma} \cdot (|\mathcal{R}(s, a_i) - \mathcal{R}(s, a_j)| + \gamma \cdot W_2(\mathcal{P}(s'|s, a_i), \mathcal{P}(s'|s, a_j) + p)$, otherwise, $|Q^\pi(s, a_i) - Q^\pi(s, a_j)| \leq \frac{1}{1-\gamma} \cdot (|\mathcal{R}(s, a_i) - \mathcal{R}(s, a_j)| + \gamma \cdot W_2(\mathcal{P}(s'|s, a_i), \mathcal{P}(s'|s, a_j))$. This indicates that if two actions are from two distributions, the difference between their values tends to have a looser upper bound depending on the constant p, and p plays a role in balancing between penalizing and exploring the o.o.d. actions. If p is large enough, the difference between two values tends to be large, so the offline algorithms would ignore most o.o.d. actions. If p is proper and the two actions' behavioral relation is close, the values of o.o.d. actions and the values of in-distribution actions would be similar but avoid to be the same. This can encourage the offline policy to cautiously explore the effect of o.o.d. actions.

Now, we would show how action representations complying the pseudemetric d improve downstream offline RL tasks. Assuming that the MERLION action encoder $\phi(a; s)$ has a learning error ϵ: If the pseudometric distance between any two action is smaller than ϵ, they would be aggregated together. Building on the policy theoretical analysis from Ajay et al. (2021) and Kumar et al. (2020), we now bound the performance of the policy obtained when offline RL is performed with MERLION.

Theorem 4.4 (Performance bound in offline RL)**.** *Let $\pi_i^*(e|s)$ be the policy obtained by CQL performing with MERLION in the constructed MDP $\overline{\mathcal{M}}$ and $\pi_{i,g}^*(a|s)$ refer to the overall policy when $\pi_i^*(e|s)$ is used together with nearest lookup function g. Let $J(\pi, \mathcal{M})$ refer to the expected return of π in $\mathcal{M}$ and $\phi(a;s)$ is the MERLION action encoder, which has a learning error ϵ. Let π_β refer to the behavioral policy generating $\mathcal{D}$ and $\overline{\pi}_\beta(e|s) \equiv e = \phi(a;s), a \sim \pi_\beta(a|s)$.Then, $J(\pi_{i,g}^*, \mathcal{M}) \geq J(\pi_\beta, \mathcal{M}) - k$ where*

$$k = \mathcal{O}\left(\frac{1}{(1-\gamma)^2}\mathbb{E}_{s\sim d^{\pi_i^*(s)}_{\overline{\mathcal{M}}}}\left[\sqrt{|\mathcal{E}|D_{CQL}(\pi_i^*, \overline{\pi}_\beta)(s)+1}\right]\right) - \frac{\alpha}{1-\gamma}\mathbb{E}_{s\sim d^{\pi_i^*(s)}_{\overline{\mathcal{M}}}}[D_{CQL}(\pi_i^*, \overline{\pi}_\beta)(s)] + \frac{\epsilon + 2\gamma\mathcal{R}_{max}}{1-\gamma} \tag{9}$$

Proof in appendix. This bound suggests that the lower bound over the performance of the learned overall policy depends on three factors: 1) the divergence between the learned latent-space policy and the latent behavioral policy ($D_{CQL}(\pi_i^*, \overline{\pi}_\beta)(s)$), which is controlled by the applied offline algorithm. 2) The number of the projected latent actions $|\mathcal{E}|$. According to Eq.(6) in Kumar et al. (2020), if we directly learn an offline policy from the original action space $\mathcal{A}$, the lower bound of offline policy performance would depend on $|\mathcal{A}|$. In general, we have $|\mathcal{E}| \leq |\mathcal{A}|$. However, in special cases where actions naturally group together, we will have $|\mathcal{E}| \ll |\mathcal{A}|$. For example, in the Maze environment mentioned in the next section, the original actions correspond to the combined state of 8 atuators, and $|\mathcal{A}| = 2^8 = 4096$. However, since the actual effect of an action is the vectorial summation of the displacements associated with the selected actuators, there are only 722 unique efects of actions ($|\mathcal{E}| = 722$). This also gives us an intuition: MERLION provides a tighter bound for policy improvement when the actions group together in the latent spaces. 3) The learning error of MERLION action encoder ϵ. This term is induced by a self-supervised loss function. Compared with errors induced by offline RL algorithms, this term is much easier to be controlled. In conclusion, MERLION provides a therotically gurrantee of policy improvement for offline RL tasks with large discrete action spaces.

5. Experiments

In this section, we empirically show that MERLION could be used as a drop-in extension for improving the policy performance in existing offline RL algorithms for problems with large discrete action spaces. Combing MERLION with two standard offline RL algorithms, BCQ (Fujimoto et al., 2019b) and CQL (Kumar et al., 2020), we evaluate our method in two simulated tasks and two real-world problems. All experiments in this paper are carried out with 5 different random seeds. We believe that the reported results can be further improved by using our framework with other offline RL algorithms; we leave this for future work. Details of architectures and hyperparameters can be found in the appendix.

Evaluation Environments. We consider four environments with large discrete action spaces: A **maze** environment introduced in Chandak et al. (2019). It has 4096 discrete actions. A **multi-step maze** environment that requires select actions from the 6-step planning decisions. It is modified from the maze environment and also has 4096 actions. A video **recommender system** that sets 1000 videos as the action set (Ie et al., 2019). A **dialogue system** that sets 1500 dialogues as the action set. The experience trajectories are policies with sub-optimal performance. They are all trained by online methods. Note that more details of environments and datasets can be found in the appendix.

5.1. Experimental Results

We empirically reveal that optimizing policies on the learned action representation spaces leads to better performance than directly training policies on the original action spaces. We consider two widely-used offline RL algorithms: CQL (Kumar et al., 2020) and BCQ (Fujimoto et al., 2019b). We first learn our pseudometric-based action representations from the given datasets. We utilize our framework to train policies using the two offline RL algorithms. The developed policies are annotated by **MERLION-CQL** and **MERLION-CQL**, respectively. The baselines are the **discrete** versions of **CQL** and **BCQ** and behavior cloning (**BC**).

The average performance curves are illustrated in Fig.2 (a). We also visualize the average performance of the **behavioral** policy. Overall, we find that MERLION-CQL and MERLION-BCQ show faster convergence and better performance than their discrete versions in the four environments. Furthermore, discrete CQL and discrete BCQ cannot outperform the behavior policies and show unstable learning curves in some situations. This indicates that in offline RL with large action sets, directly training policies on the original action spaces might suffer from serious generalization or extrapolation errors. In addition, the poor performance of BC in most environments indicates that simple imitation learning cannot address this problem well.

Comparison with Existing Action Encoders in Offline Setting. A natural question is whether training policies on other kinds of action representation spaces can lead to similar results? To reply to this question, we compare MERLION-CQL against baselines using the same architecture but exchange our action representations with other representations. We first consider both random action representations to examine whether simply projecting discrete

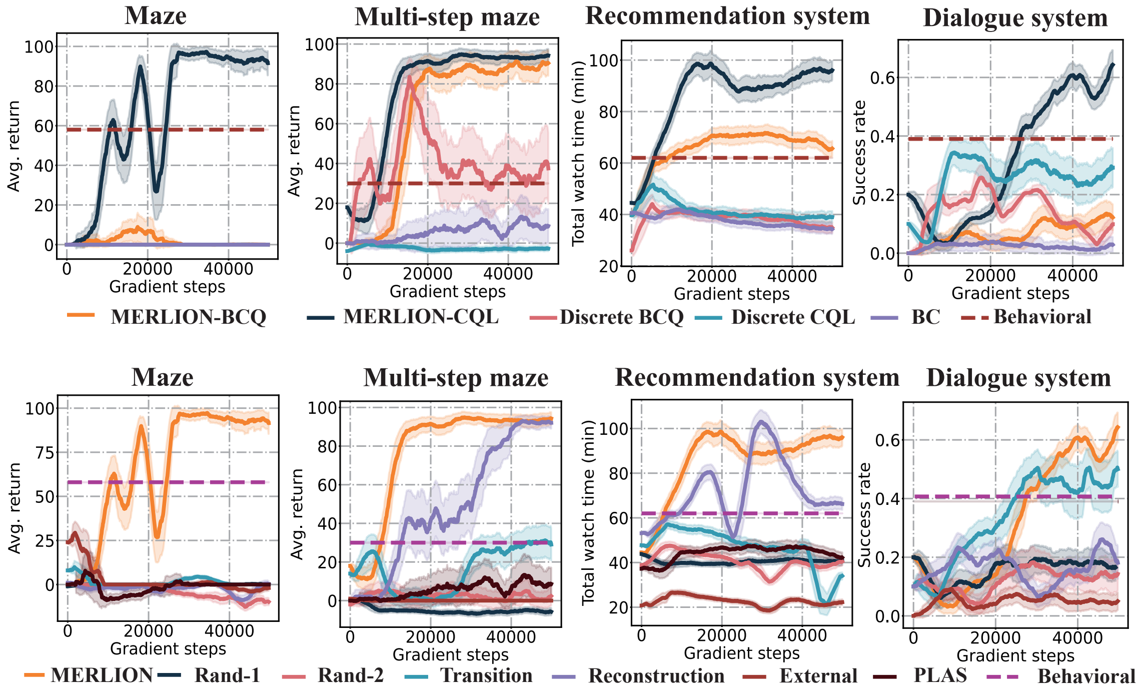

Figure 2. **Experimental results. (a)** Comparing MERLION equipped with BCQ and CQL against directly training offline RL algorithms (Discrete BCQ, Discrete CQL, and BC) on the original action spaces in 4 environments with large action spaces. **(b)** Comparing the performance of MERLION against other widely used action representations (transition-based representation, reconstruction-based representation, two kinds of random representations, and external representation) in 4 environments with large action spaces.

actions into continuous action spaces can improve the performance. The first scheme generates random vectors ranging from -1 to 1 as action representations (**Rand-1**). The other scheme also generates random vectors but forces the ℓ_1 distance between any two action embeddings to be larger than a constant. This scheme avoids actions overlapping together (**Rand-2**). Then, we consider three schemes for learning action representations widely used in online tasks: (1) **Transition** (Chandak et al., 2019): An encoder that captures information about corresponding state transitions into action representations. (2) **Reconstruction**: An encoder-decoder scheme in which the decoder takes the state and the action representations generated by the encoder as input, and its objective is to predict the next state and reward. (3) **External** (Lee et al., 2018): Action representations are given by external information. In the maze environment, the action embeddings are set as the movement vectors. In the multi-step environment, the actions are represented as the concatenation of the base action at each step of the plan. In the recommendation system, the action embedding is the concatenation of video type (expressed as a one-hot vector), video quality, and video length. Note that there is no proper external information to represent dialogue actions, so we did not consider external action representations in this task. Finally, we consider a similar work for offline RL (**PLAS** (Zhou et al., 2020)) since it also trains policies on the latent action spaces.

The average performance curves are illustrated in Fig.2 (b). Both random action representations fail in the four environments, indicating that simply converting discrete action spaces to continuous action spaces cannot address this task. By comparison, transition and reconstruction representations show better performance in certain situations, revealing that generalization over actions with similar effects can improve the policy performance. However, these two representations also show poor performance in most environments. In contrast, our method show superior results among all environments, revealing that action representations complying with the proposed pseudometric are more suitable to offline RL. In addition, external representations and PLAS both show poor performance in all tasks. The reason might be that external representations do not capture the information of dynamics in MDPs, resulting in misleading generalization effects. And the objective of PLAS is to constrain policies more naturally instead of improving the generalization ability of action representations, so it cannot address the tasks with large discrete action spaces well.

Fig.4 illustrates the action representations. Since we set the dimension of representations as 2 in the multi-step maze environment, we directly plot the action representations. The position of each dot in these plots equals the correspond-

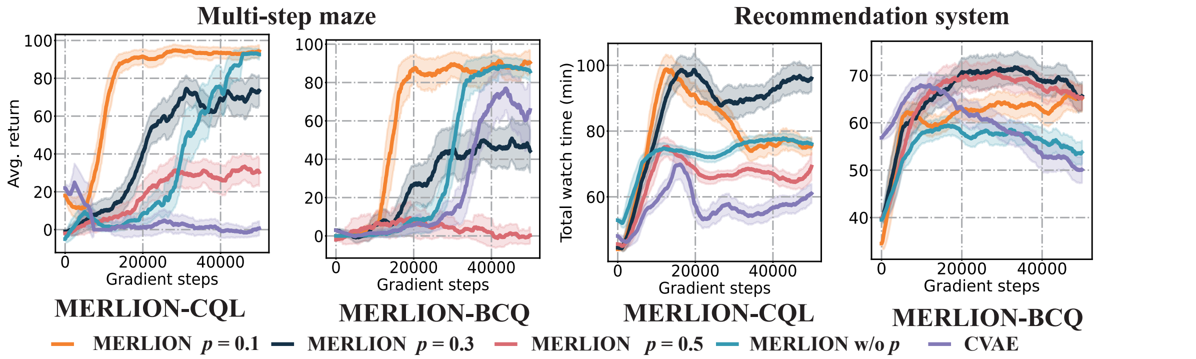

Figure 3. **Ablations.** We perform two offline algorithms (CQL and BCQ) with MERLION in two environments (Multi-step maze and Recommendation system). We consider MERLION with different penalty distances (MERLION with $p = 0.1, 0.3, 0.5$), removing the penalty distance from the learning objective (MERLION w/o p), and removing the distance learning objective (Eq.6) from the learning procedure (CVAE).

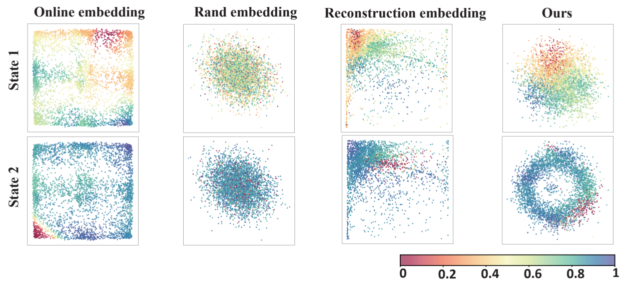

Figure 4. **Visualization of different representation spaces.**

ing action embeddings, and the color corresponds to the ground-truth action value predicted by a model (Chandak et al., 2019) with online training. Two rows are in terms of two different states. Each column from left to right corresponds to the representations learned by the online model, the rand-1 representations, the reconstruction representations, and ours. We see that, in offline settings, the rand action representations cannot generalize well since actions with different values are mixed together. The reconstruction-based representations and our pseudometric-based representations both show generalization over actions: The changes of action value are continuous in both representation spaces to a certain extent. Furthermore, the reconstruction-based representations are scattered throughout the space while our representations with similar values are arranged as a more compact structure, indicating that our learning framework introduces an explicit mechanism to aggregate the action representations with similar behavioral effects. Intuitively, we believe that this is the reason why our method shows faster convergence performance than reconstruction-based representations.

Comparison with Other Action Encoders in Online Setting. MERLION can also be applied in the online setting. We compared MERLION with the transition and the reconstruction representations in the online setting. The results are illustrated in Fig.7 in the appendix. As illustrated, three action representations can all achieve fast convergence in the online setting, while only MERLION shows stable performance in the offline setting (Fig.2 (b)). This indicates that merely by using the data in the offline dataset, MERLION is able to learn effective action representations.

Ablations We perform several ablations on the multi-step maze environment and recommendation system to figure out the importance of each component in our framework. Fig.3 illustrates the results. We first consider the effects of different scales of the penalty constant. We set $p = 0.1, 0.3, 0.5$ respectively. The results show that a proper penalty constant would bring better performance, but a large penalty constant might make the performance collapse in some situations. The reason might be that large penalty distances destroy the generalization structure of action representations. Then, we consider removing the penalty constant (denoted as **MERLION w/o p**). The results of this ablation show that without the distance penalty mechanism to ensure the distribution detection, there is performance degradation in our framework. Finally, we remove the objective function (Eq.6) for controlling the distances between actions. As a result, the architecture of this scenario is similar to a conditional variational auto-encoder that takes current states and actions as input and aims to reconstruct next states and rewards (denoted as **CVAE**) from the action embeddings. This ablation shows worse performance than **MERLION w/o p** in many situations, indicating that controlling the pseudometric distances between actions provides extra information for action representations, leading to better policy performance.

6. Conclusions

This paper proposes MERLION, a novel action representation learning framework for offline RL with large action spaces. In the learned action representation spaces, the distances between representations of discrete actions reflect their behavioral and data-distributional relations. We derive theoretically statements about the benefits of learning policies based on MERLION action representations. Experimental results and ablations show that our method significantly outperform prior works in a variety of environments.

Acknowledgements

This research was supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-RP-2019-0013), National Satellite of Excellence in Trustworthy Software Systems (Award No: NSOETSS2019-01), and NTU.

References

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. OPAL: Offline primitive discovery for accelerating offline reinforcement learning. In *ICLR*, 2021. URL https://openreview.net/forum?id=V69LGwJ0lIN.

Chandak, Y., Theocharous, G., Kostas, J., Jordan, S. M., and Thomas, P. S. Learning action representations for reinforcement learning. In *ICML*, pp. 941–950, 2019.

Chandak, Y., Theocharous, G., Nota, C., and Thomas, P. S. Lifelong learning with a changing action set. In *AAAI*, pp. 3373–3380, 2020.

Dadashi, R., Rezaeifar, S., Vieillard, N., Hussenot, L., Pietquin, O., and Geist, M. Offline reinforcement learning with pseudometric learning. In *ICML*, pp. 2307–2318, 2021.

Eric, M., Goel, R., Paul, S., Sethi, A., Agarwal, S., Gao, S., and Hakkani-Tur, D. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*, 2019.

Ferns, N., Panangaden, P., and Precup, D. Metrics for finite markov decision processes. In *UAI*, pp. 2052–2062, 2004.

Fujimoto, S., Conti, E., Ghavamzadeh, M., and Pineau, J. Benchmarking batch deep reinforcement learning algorithms. *arXiv preprint 1910.01708*, 2019a.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *ICML*, pp. 2052–2062, 2019b.

Gulcehre, C., Wang, Z., Novikov, A., Paine, T. L., Colmenarejo, S. G., Zolna, K., Agarwal, R., Merel, J., Mankowitz, D., Paduraru, C., Dulac-Arnold, G., Li, J., Norouzi, M., Hoffman, M., Nachum, O., Tucker, G., Heess, N., and de Freitas, N. Rl unplugged: A suite of benchmarks for offline reinforcement learning. In *NeurIPS*, pp. 7248–7259, 2021.

Ie, E., wei Hsu, C., Mladenov, M., Jain, V., Narvekar, S., Wang, J., Wu, R., and Boutilier, C. Recsim: A configurable simulation platform for recommender systems. *arXiv preprint 1909.04847*, 2019.

Kostrikov, I., Tompson, J., Fergus, R., and Nachum, O. Offline reinforcement learning with fisher divergence critic regularization. In *ICML*, pp. 5774–5783, 2021.

Kumar, A., Fu, J., Tucker, G., and Levine, S. Stabilizing off-policy Q-Learning via bootstrapping error reduction. In *NeurIPS*, pp. 11761–11771, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-Learning for offline reinforcement learning. In *NeurIPS*, pp. 1179–1191, 2020.

Le Lan, C., G. Bellemare, M., and Castro, P. S. Metrics and continuity in reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 8261–8269, 2021.

Lee, K., Kim, S.-A., Choi, J., and Lee, S.-W. Deep reinforcement learning in continuous action spaces: a case study in the game of simulated curling. In *ICML*, volume 80, pp. 2937–2946, 2018.

Lee, S., Zhu, Q., Takanobu, R., Li, X., Zhang, Y., Zhang, Z., Li, J., Peng, B., Li, X., Huang, M., and Gao, J. Convlab: Multi-domain end-to-end dialog system platform. In *ACL*, pp. 64–69, 2019.

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

Li, J., Monroe, W., Ritter, A., Jurafsky, D., Galley, M., and Gao, J. Deep reinforcement learning for dialogue generation. In *EMNLP*, 2016.

Matsushima, T., Furuta, H., Matsuo, Y., Nachum, O., and Gu, S. Deployment-efficient reinforcement learning via model-based offline optimization. In *ICLR*, 2021. URL https://openreview.net/forum?id=3hGNqpI4WS.

Nachum, O., Dai, B., Kostrikov, I., Chow, Y., Li, L., and Schuurmans, D. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint 1912.02074*, 2019.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. A simple neural network module for relational reasoning. In *NeurIPS*, 2017. URL https://proceedings.

neurips.cc/paper/2017/file/e6acf4b0f69f6f6e60e9a815938aa1ff-Paper.pdf.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *ICML*, pp. 1889–1897, 2015.

Singh, A., Liu, H., Zhou, G., Yu, A., Rhinehart, N., and Levine, S. Parrot: Data-driven behavioral priors for reinforcement learning. In *ICLR*, 2021.

Tennenholtz, G. and Mannor, S. The natural language of actions. In *ICML*, pp. 6196–6205, 2019.

Thomas, P. S. *Safe reinforcement learning*. Ph.D. diss., University of Massachusetts Libraries, 2015.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IROS*, pp. 5026–5033, 2012.

Wang, T., Gupta, T., Mahajan, A., Peng, B., Whiteson, S., and Zhang, C. RODE: Learning roles to decompose multi-agent tasks. In *ICLR*, 2021. URL https://openreview.net/forum?id=TTUVg6vkNjK.

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

Yang, M. and Nachum, O. Representation matters: Offline pretraining for sequential decision making. In Meila, M. and Zhang, T. (eds.), *ICML*, pp. 11784–11794, 2021.

Young, S. J. Using pomdps for dialog management. In *SLT*, 2006.

Zhang, A., McAllister, R., Calandra, R., Gal, Y., and Levine, S. Learning invariant representations for reinforcement learning without reconstruction. In *ICLR*, 2021. URL https://openreview.net/forum?id=-2FCwDKRREu.

Zhou, W., Bajracharya, S., and Held, D. PLAS: Latent action space for offline reinforcement learning. In *CoRL*, pp. 1719–1735, 2020.

A. Appendix

A.1. Additional Theorems and Proofs

Theorem 4.3 (Q^π is Lipschiz with respect to d). *Given a policy $\pi(a|s)$, let Q^π be the value function for a given discount factor γ. Q^π is Lipschitz continuous with respect to d with a Lipschitz constant $\frac{1}{1-\gamma}$*

$$|Q^\pi(s,a_i) - Q^\pi(s,a_j)| \le \frac{1}{1-\gamma} d(a_i,a_j|s) \tag{10}$$

Proof. follows by expanding each Q, rearranging terms and then simplifying the expression.

$$\begin{aligned} |Q^\pi(s,a_i) - Q^\pi(s,a_j)| &= |\mathcal{R}(s,a_i) + \gamma \cdot \sum_{s'} \mathcal{P}(s'|s,a_i)V^\pi(s') - (\mathcal{R}(s,a_j) + \gamma \cdot \sum_{s'} \mathcal{P}(s'|s,a_j))V^\pi(s')| \\ &\le |\mathcal{R}(s,a_i) - \mathcal{R}(s,a_j)| + \gamma \cdot \sum_{s'} |\mathcal{P}(s'|s,a_i) - \mathcal{P}(s'|s,a_j)| \cdot V^\pi(s') \end{aligned} \tag{11}$$

We assume that $\mathcal{R} \le 1$, so we get

$$V^\pi(s) = \sum_{t=0}^{\infty} \gamma^t \cdot \mathcal{R}_t \le \frac{1}{1-\gamma} \tag{12}$$

The assumption of $R \le 1$ is equivalent to that the reward function is bounded, i.e., $\mathcal{R} \in [-\mathcal{R}_{max}, \mathcal{R}_{max}]$ where $\mathcal{R}_{max}$ is a real number. Because we can simply normalize the reward signals as $\mathcal{R} \leftarrow \mathcal{R}/\mathcal{R}_{max}$ without loss of generality. Combing the above equations, we have

$$\begin{aligned} |Q^\pi(s,a_i) - Q^\pi(s,a_j)| &\le |\mathcal{R}(s,a_i) - \mathcal{R}(s,a_j)| + \gamma \cdot \sum_{s'} |\mathcal{P}(s'|s,a_i) - \mathcal{P}(s'|s,a_j)| \cdot V^\pi(s') \\ &\le |\mathcal{R}(s,a_i) - \mathcal{R}(s,a_j)| + \frac{\gamma}{1-\gamma} \cdot \sum_{s'} |\mathcal{P}(s'|s,a_i) - \mathcal{P}(s'|s,a_j)| \\ &= \frac{1}{1-\gamma} \cdot \left((1-\gamma) \cdot |\mathcal{R}(s,a_i) - \mathcal{R}(s,a_j)| + \gamma \cdot \sum_{s'} |\mathcal{P}(s'|s,a_i) - \mathcal{P}(s'|s,a_j)| \right) \\ &\le \frac{1}{1-\gamma} \cdot \left(|\mathcal{R}(s,a_i) - \mathcal{R}(s,a_j)| + \gamma \cdot \sum_{s'} |\mathcal{P}(s'|s,a_i) - \mathcal{P}(s'|s,a_j)| \right) \end{aligned} \tag{13}$$

Since in Eq.(4), $p > 0$ and $I_\beta(a_i,a_j|s) \in \{0,1\}$, we get

$$\begin{aligned} |Q^\pi(s,a_i) - Q^\pi(s,a_j)| &\le \frac{1}{1-\gamma} \cdot \left(|\mathcal{R}(s,a_i) - \mathcal{R}(s,a_j)| + \gamma \cdot \sum_{s'} |\mathcal{P}(s'|s,a_i) - \mathcal{P}(s'|s,a_j)| + p \cdot I_\beta(a_i,a_j|s) \right) \\ &\le \frac{1}{1-\gamma} d(a_i,a_j|s) \end{aligned} \tag{14}$$

According to theorem 4.3, we can derive the bound between value functions in the original MDP and the MDP constructed by the learned MERLION action representations. Assuming that the MERLION action encoder $\phi(a;s)$ has a learning error ϵ: If the pseudometric distance between any two action is smaller than ϵ, they would be aggregated together.

Lemma A.1 (Value bound based on ϕ)**.** *Given the MERLION action encoder $\phi(a;s)$ which has a learning error ϵ and maps from actions in the original MDP $\mathcal{M}$ to actions in the MDP $\overline{\mathcal{M}}$ constructed by the action representations. The value functions for a given policy π in the orginal MDP and its converted version $\overline{\pi}$ in the constructed MDP are bounded as:*

$$|Q^\pi_{\mathcal{M}}(s,a) - Q^{\overline{\pi}}_{\overline{\mathcal{M}}}(s,\phi(a;s))| \le \frac{\epsilon}{1-\gamma} \tag{15}$$

Proofs. First, let us give some notation and assumptions concerning the form of the constructed MDP $\overline{\mathcal{M}}$. We assume $\overline{\mathcal{M}} = (\mathcal{S}, \mathcal{A}_\phi, \mathcal{P}_\phi, \mathcal{R}_\phi, \gamma)$, where $\mathcal{S}$ is the same state space, $\mathcal{A}_\phi$ is the action representation space where the ℓ_1 distances between any two representations corresponds to our defined pseudometric distance, and there exists a learning error of ϕ: If the pseudometric distance between any two action is smaller than ϵ, they would be aggregated together. Recall that the nearest neighbor lookup g is as follow:

$$g(e) = \arg\min_{a \in \mathcal{A}} \|e - \phi(a; s)\|_1 \tag{16}$$

As a result, we can define the relationships between the transition probabilities and rewards in the two MDPs, i.e.

$$\begin{aligned} \mathcal{P}_\phi(s'|s, e) = \mathcal{P}(s'|s, g(e)) \\ \mathcal{R}_\phi(s, e) = \mathcal{R}(s, g(e)) \end{aligned} \tag{17}$$

The converted policy $\overline{\pi}(e|s)$ can be regarded as the policy $\pi'(a'|s)$ executed in $\mathcal{M}$, where $a' = g(e)$, $e = \phi(a; s)$. Due to the learning error, $\pi'(a'|s)$ takes a different action than $\pi(a|s)$ with probability at most ϵ_p, so we get $|\pi'(a'|s) - \pi(a|s)| \leq \epsilon_p$.Then, we can represent the stationary distribution $d_{\pi'}(s_t)$ of the policy π' as follows:

$$d_{\pi'}(s_t) = (1 - \epsilon_p)^t d_\pi(s_t) + (1 - (1 - \epsilon_p)^t) d_m(s_t) \tag{18}$$

where $(1-\epsilon_p)^t$ indicates the probability that π' takes the same action sequence as π, and d_m refers to some other distributions. Given this, we bound $|d_{\pi'}(s_t) - d_\pi(s_t)|$ by:

$$\begin{aligned} |d_{\pi'}(s_t) - d_\pi(s_t)| &= (1 - (1 - \epsilon_p)^t)|d_m(s_t) - d_\pi(s_t)| \\ &\leq 2(1 - (1 - \epsilon_p)^t) \end{aligned} \tag{19}$$

Proofs of both equations mentioned above are based on (Schulman et al., 2015). Now, we obtain

$$\begin{aligned} &|V^{\overline{\pi}_\beta}_{\overline{\mathcal{M}}}(s) - V^{\pi_\beta}_{\mathcal{M}}(s)| = |V^{\pi'}_{\beta}{}_{\mathcal{M}}(s) - V^{\pi_\beta}_{\mathcal{M}}(s)| \\ &= |\sum_t \sum_{s_t} d_{\pi'}(s_t) \cdot \gamma^t \cdot \mathbb{E}_{a \sim \pi'_\beta} \mathcal{R}^a_{s_t} - \sum_t \sum_{s_t} d_\pi(s_t) \cdot \gamma^t \cdot \mathbb{E}_{a \sim \pi_\beta} \mathcal{R}^a_{s_t}| \\ &\leq \sum_t \gamma^t \cdot \mathcal{R}_{max} |d_{\pi'}(s_t) - d_\pi(s_t)| \leq \mathcal{R}_{max} \sum_t \gamma^t \cdot \big(2(1 - (1 - \epsilon_p)^t\big) \\ &\leq 2\mathcal{R}_{max} \sum_t \gamma^t \leq \frac{2\mathcal{R}_{max}}{1 - \gamma} \end{aligned} \tag{20}$$

Next, we can get

$$\begin{aligned} &|Q^\pi_{\mathcal{M}}(s, a) - Q^{\overline{\pi}}_{\overline{\mathcal{M}}}(s, \phi(a; s))| \\ &= |\mathcal{R}^a_s + \gamma \cdot \sum_{s'} \mathcal{P}^a_s V^\pi_{\mathcal{M}}(s') - \mathcal{R}^{\phi,g}_s - \gamma \cdot \sum_{s'} \mathcal{P}^{\phi,g}_s V^\pi_{\mathcal{M}}(s') \\ &+ \gamma \cdot \sum_{s'} \mathcal{P}^{\phi,g}_s V^\pi_{\mathcal{M}}(s') - \gamma \cdot \sum_{s'} \mathcal{P}^{\phi,g}_s V^{\overline{\pi}}_{\overline{\mathcal{M}}}(s')| \\ &\leq |\mathcal{R}^a_s - \mathcal{R}^{\phi,g}_s| + \gamma \sum_{s'} |\mathcal{P}^a_s - \mathcal{P}^{\phi,g}_s| V^\pi_{\mathcal{M}}(s') + \gamma \sum_{s'} \mathcal{P}^{\phi,g}_s |V^\pi_{\mathcal{M}}(s') - V^{\overline{\pi}}_{\overline{\mathcal{M}}}(s')| \\ &\leq \frac{\epsilon}{1 - \gamma} + \frac{2\gamma\mathcal{R}_{max}}{1 - \gamma} = \frac{\epsilon + 2\gamma\mathcal{R}_{max}}{1 - \gamma} \end{aligned} \tag{21}$$

We have argued that there exists some near-optimal value functions based on the constructed MDP $\overline{\mathcal{M}}$, if ϕ is sufficiently learned. Now, we would show how action representations complying the pseudemetric d improve dowenstream offline RL. Building on the policy theoretical analysis from Ajay et al. (2021) and Kumar et al. (2020), we now bound the performance of the policy obtained when offline RL is performed with MERLION.

Theorem 4.4 (Performance bound in offline RL). *Let $\pi^*_i(e|s)$ be the policy obtained by CQL performing with MERLION in the constructed MDP $\overline{\mathcal{M}}$ and $\pi^*_{i,g}(a|s)$ refer to the overall policy when $\pi^*_i(e|s)$ is used together with nearest lookup function g. Let $J(\pi, \mathcal{M})$ refer to the expected return of π in $\mathcal{M}$ and $\phi(a; s)$ is the MERLION action encoder, which has* *a learning error ϵ. Let π_β refer to the behavioral policy genearting $\mathcal{D}$ and $\overline{\pi}_\beta(e|s) \equiv e = \phi(a;s), a \sim \pi_\beta(a|s)$.Then, $J(\pi^*_{i,g}, \mathcal{M}) \geq J(\pi_\beta, \mathcal{M}) - k$ where*

$$
\begin{aligned}
k = \mathcal{O}\left(\frac{1}{(1-\gamma)^2}\mathbb{E}_{s\sim d^{\pi_i^*(s)}_{\overline{\hat{\mathcal{M}}}}}\left[\sqrt{|\mathcal{E}|D_{CQL}(\pi_i^*, \overline{\pi}_\beta)(s)+1}\right]\right) \\
- \frac{\alpha}{1-\gamma}\mathbb{E}_{s\sim d^{\pi_i^*(s)}_{\overline{\mathcal{M}}}}[D_{CQL}(\pi_i^*, \overline{\pi}_\beta)(s)] + \frac{\epsilon + 2\gamma\mathcal{R}_{max}}{1-\gamma}
\end{aligned} \tag{22}
$$

Proof. First, let's break $|J(\pi^*_{i,g}, \mathcal{M}) - J(\pi_\beta, \mathcal{M})|$ into

$$
\begin{aligned}
|J(\pi^*_{i,g}, \mathcal{M}) - J(\pi_\beta, \mathcal{M})| \leq{}& |J(\pi^*_{i,g}, \mathcal{M}) - J(\pi_i^*, \overline{\mathcal{M}})| \\
&+ |J(\pi_i^*, \overline{\mathcal{M}}) - J(\overline{\pi}_\beta, \overline{\mathcal{M}})| \\
&+ |J(\overline{\pi}_\beta, \overline{\mathcal{M}}) - J(\pi_\beta, \mathcal{M})|
\end{aligned} \tag{23}
$$

where $\overline{\pi}_\beta$ is the converted behavioral policy executed on the constructed MDP $\overline{\mathcal{M}}$ mentioned in the proofs of lemma A.1. It can be defined as $e \sim \overline{\pi}_\beta \equiv e = \phi(a;s), a \sim \pi_\beta(a|s)$. According to theorem 3.6 in Kumar et al. (2020), we apply it to $\overline{\mathcal{M}}$, we get

$$
\begin{aligned}
&|J(\pi_i^*, \overline{\mathcal{M}}) - J(\overline{\pi}_\beta, \overline{\mathcal{M}})| \\
&\leq 2\left(\frac{C_{r,\delta}}{1-\gamma} + \frac{\mathcal{P}_\phi, \delta}{(1-\gamma)^2}\right)\mathbb{E}_{s\sim d^{\pi_i^*(s)}_{\overline{\hat{\mathcal{M}}}}}\left[\sqrt{\frac{|\mathcal{E}|}{|\mathcal{D}(s)|}D_{CQL}(\pi_i^*, \overline{\pi}_\beta)(s)+1}\right] \\
&- \frac{\alpha}{1-\gamma}\mathbb{E}_{s\sim d^{\pi_i^*(s)}_{\overline{\mathcal{M}}}}[D_{CQL}(\pi_i^*, \overline{\pi}_\beta)(s)] = k_1
\end{aligned} \tag{24}
$$

Note that this theorem holds only when the number of actions is *finite*. Note that, in our setting, the latent action space is continuous for the ease of learning, but the number of learned action representations is finite because each original action corresponds to one projected latent action. But note that two different projected actions may have the same representation, therefore $|\mathcal{E}| \leq |\mathcal{A}|$.

Then, we try to bound $|J(\pi^*_{i,g}, \mathcal{M}) - J(\pi_i^*, \overline{\mathcal{M}})|$. According to the definition $\pi_{i,g}(a|s) \equiv a = g(e), e \sim \pi_i(e|s)$, a latent action e executed in $\overline{\mathcal{M}}$ is equivalent to a executed in $\mathcal{M}$. Therefore, $\pi_i(e|s)$ executed in $\overline{\mathcal{M}}$ is also equivalent to the overall policy $\pi_{i,g}(a|s)$ executed in $\overline{\mathcal{M}}$. So, the decisions of π_i^* and $\pi^*_{i,g}$ are in one-to-one correspondence. we get $|J(\pi^*_{i,g}, \mathcal{M}) - J(\pi_i^*, \overline{\mathcal{M}})| = 0$.

Now, we try to bound $|J(\overline{\pi}_\beta, \overline{\mathcal{M}}) - J(\pi_\beta, \mathcal{M})|$:

$$
\begin{aligned}
|J(\overline{\pi}_\beta, \overline{\mathcal{M}}) - J(\pi_\beta, \mathcal{M})| &= |\sum_{s\sim\mathcal{S}} d_0(s)V^{\overline{\pi}_\beta}_{\overline{\mathcal{M}}}(s) - \sum_{s\sim\mathcal{S}} d_0(s)V^{\pi_\beta}_{\mathcal{M}}(s)| \\
&= |\sum_{s\sim\mathcal{S}} d_0(s)\sum_{e\sim\mathcal{E}}\overline{\pi}_\beta(e|s)Q^{\overline{\pi}_\beta}_{\overline{\mathcal{M}}}(s,e) - \sum_{s\sim\mathcal{S}} d_0(s)\sum_{a\sim\mathcal{A}}\pi_\beta(a|s)Q^{\pi_\beta}_{\mathcal{M}}(s,a)| \\
&= |\sum_{s\sim\mathcal{S}} d_0(s)\sum_{a\sim\mathcal{A}}\pi_\beta(a|s)Q^{\overline{\pi}_\beta}_{\overline{\mathcal{M}}}(s,e) - \sum_{s\sim\mathcal{S}} d_0(s)\sum_{a\sim\mathcal{A}}\pi_\beta(a|s)Q^{\pi_\beta}_{\mathcal{M}}(s,a)| \\
&= \sum_{s\sim\mathcal{S}} d_0(s)\sum_{a\sim\mathcal{A}}\pi_\beta(a|s)|Q^{\overline{\pi}_\beta}_{\overline{\mathcal{M}}}(s,e) - Q^{\pi_\beta}_{\mathcal{M}}(s,a)|
\end{aligned} \tag{25}
$$

According to lemma A.1, $|Q^{\overline{\pi}_\beta}_{\overline{\mathcal{M}}}(s, e^a_s) - Q^{\pi_\beta}_{\mathcal{M}}(s,a)| = \frac{\epsilon+2\gamma\mathcal{R}_{max}}{1-\gamma}$, so $|J(\overline{\pi}_\beta, \overline{\mathcal{M}}) - J(\pi_\beta, \mathcal{M})| = \frac{\epsilon+2\gamma\mathcal{R}_{max}}{1-\gamma} = k_2$. Finally, we get $k = k_1 + k_2$. We apply $\mathcal{O}$ to get the notation in the theorem.

A.2. Algorithms

A.3. Architecture, Hyperparameters, and Infrastructure

In our implementations, the nearest neighbor lookup function g can be regarded as finding the minimum distance between $\hat{e}$ and all actions' embeddings. So, we directly adopted the `torch.min()` function in the PyTorch python package. Its time complexity is $\mathcal{O}(N)$, so there would not be a serious scale problem when the action set is enormous.

Algorithm 1 Train policy

```
Data: Offline Dataset $\mathcal{D}$, the action encoder $\phi$, Any offline RL algorithm $Y$, and number of iterations $T$.
Result: Policy $\pi_{i,\theta}(\hat{e}|s)$.
for $t = 0$ to $T$ do
    Sample data $(s, a, s', r) \sim \mathcal{D}$
    $e \leftarrow \phi(a; s)$
    Update policy $\pi_{i,\theta}(\hat{e}|s)$ with $(s, e, s', r)$ using the given algorithm $Y$
end for
```

Algorithm 2 Pseudometric-based representation learning

```
Data: Offline Dataset $\mathcal{D}$ and number of iterations $T$.
Result: Action encoder $\phi$.
for $t = 0$ to $T$ do
    Sample data $(s, a_i, \mathcal{R}(s, a_i)) \sim \mathcal{D}$
    Sample the other action $a_j \sim \mathcal{A}$
    Compute representations $e_s^i = \phi(a_i; s)$ and $e_s^j = \phi(a_j; s)$
    Estimate the transition distributions: $\hat{\mathcal{P}}(\cdot|s, e_s^i)$ and $\hat{\mathcal{P}}(\cdot|s, e_s^j)$
    Estimate the reward of $(s, a_j)$: $\hat{\mathcal{R}}(s, e_s^j)$
    Estimate whether $a_j$ is an o.o.d. action: $\hat{I}_\beta(a_j|s)$
    Compute the estimated pseudometric distance: $\hat{d}(a_i, a_j|s)$  Eq.3
    Train encoder: $J(\phi) = (\|e_s^i - e_s^j\|_1 - \hat{d}(a_i, a_j|s))^2$
    Train transition model: $J(\hat{\mathcal{P}}, \phi) = (\hat{\mathcal{P}}(\cdot|s, e_s^i) - s')^2$
    Train reward model: $J(\hat{\mathcal{R}}, \phi) = (\hat{\mathcal{R}}(s, e_s^i) - \mathcal{R}(s, a_i))^2$
    Train prediction model: $J(G) = CrossEntropy(G(\cdot|s), a_i)$
end for
```

For $\hat{I}_\beta(a|s)$. We train a model $G(a|s) \approx \pi_\beta(a|s)$ to predict the probabilities of every action under a given state s and then scale all probabilities by the maximum probability. a_j would be predicted as the o.o.d. action if its relative probability is below a threshold constant τ. This procedure can be summarized as $\hat{I}_\beta(a|s) = \frac{G(a|s)}{\max\limits_{\hat{a} \in \mathcal{A}} G(\hat{a}|s)} \leq \tau$. We have searched over $\tau = \{0, 0.1, 0.3, 0.5, 0.7, 0.9, 1\}$ and found that the best setting is $\tau = 0.3$

Furthermore, the transition model $\hat{\mathcal{P}}$, the reward model $\hat{\mathcal{R}}$, and the distribution detection model G are all implemented by neural networks. Their learning objective can be represented as follow:

$$
\begin{aligned}
&J(\hat{\mathcal{P}}) = (\hat{\mathcal{P}}(\cdot|s, e_s^i) - s')^2 \\
&J(\hat{\mathcal{R}}) = (\hat{\mathcal{R}}(s, e_s^i) - \mathcal{R}(s, a_i))^2 \\
&J(G) = CrossEntropy(G(\cdot|s), a_i)
\end{aligned}
\tag{26}
$$

During training, all data are randomly sampled from the offline dataset. We set the batch size as 128 and set the training gradient steps for all models as 10000. We control the scale of the learning objective function in all models by controlling the optimization procedure. It is conducted using Adam with a learning rate of 10^{-2}, and with no momentum or weight decay. We set the dimension of the action representations $|\mathcal{E}| = 2, 2, 10, 30$ and the penalty coefficient $p = 0.01, 0.1, 0.1, 0.3$ respectively in the maze environment, the multi-step maze environment, the recommender system, and the dialogue system respectively. Details of the neural network architectures used in our experiments are provided in Fig.5.

For the applied offline RL algorithms: BCQ (Fujimoto et al., 2019b), CQL (Kumar et al., 2020), and their discrete versions, we all adopt their open-source implementations released by the authors. For the discrete CQL, We have searched over the crucial hyperparameter $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, which determines the extent of conservative estimation of value functions. The best settings for four environments (Maze, Multi-step maze, Recommendation system, Dialogue system) are 0.5, 0.5, 0.5, 0.3, respectively. For the discrete BCQ, we have searched over $0.1, 0.3, 0.5, 0.7, 0.9$ for the hyperparameter of the threshold τ, which determines the range of the candidate actions. The best settings for four environments (Maze,

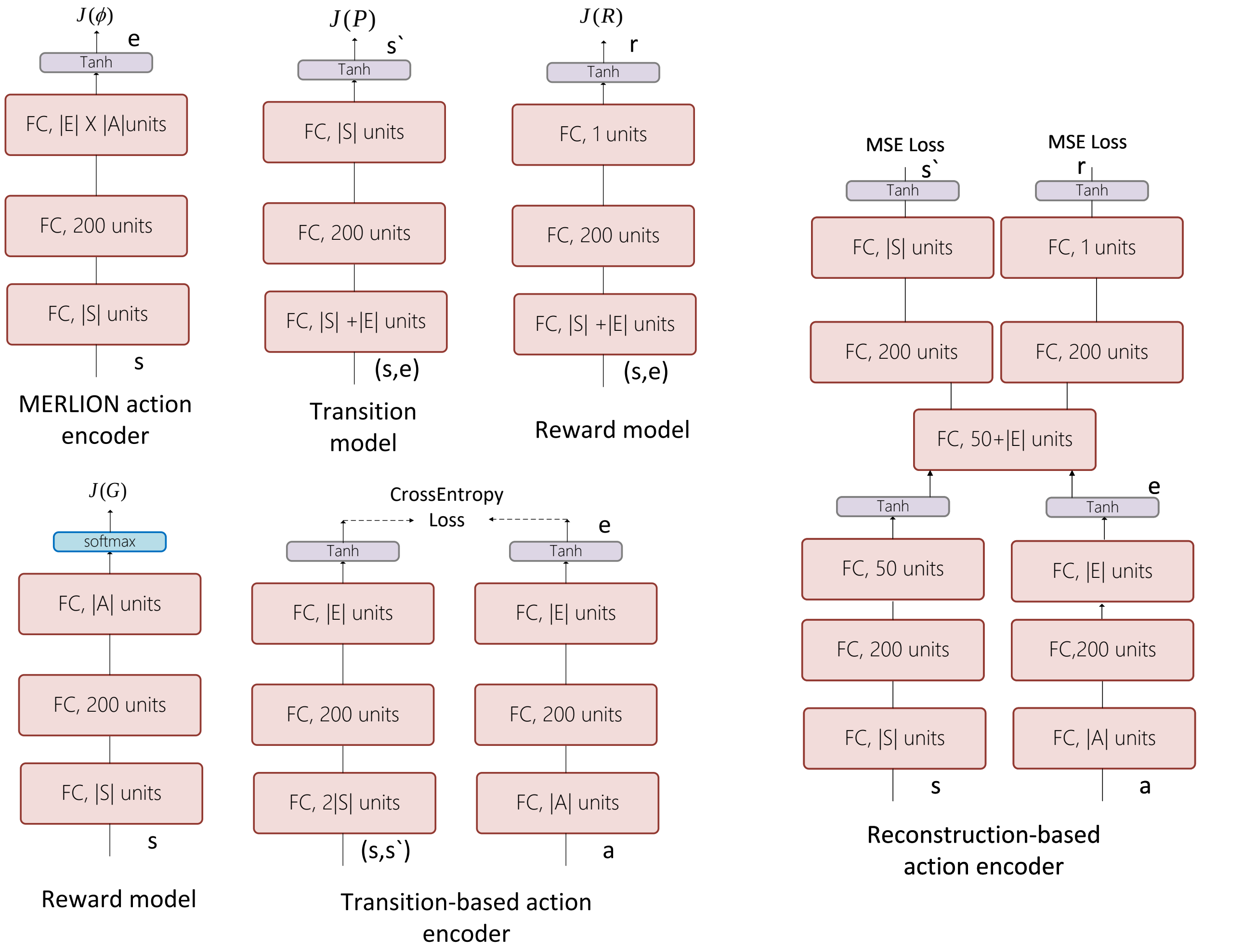

Figure 5. **Details of the network architectures used in our experiments.**

Multi-step maze, Recommendation system, Dialogue system) are 0.3, 0.3, 0.3, 0.3, respectively.

A.4. Environments and Datasets

Maze: We first adopt the maze environment introduced in Chandak et al. (2019). The state is the coordinates of the agent's current location, and the agent has n equally spaced actuators (each actuator corresponds to a unit of movement in one direction). It can choose whether each actuator should be on or off. The effect of an action is the vectorial summation of the movements associated with the selected actuators. Thus, the size of the action space $|\mathcal{A}| = 2^n$. We set $n = 12$, so there are 4096 actions in total. There are three reward types: a reward of 10 is given if the agent visits a rewarding region, a reward of -20 is given for visiting a penalizing region, and a reward of 100 is given with the episode ending if it is on a goal region. The noise was added to the action 10% of the time, and the maximum episode length was 60 steps.

Multi-step maze: Selecting one decision from all possible m-step plans is a general problem with a large action set. If an environment has i actions available at each step, the number of m-step plans would increase exponentially. In this setting, $|\mathcal{A}| = m^i$. We implement a version of this task on the maze environment mentioned above. The reward settings are the same in this task, but the agent only has 4 base actions: down, right, upward, and left. It needs to choose a 6-step plan per 6 time steps. Thus, the actual action size in this environment is $4^6 = 4096$.

Recommender environment: We consider a real-world domain with a large action set. The agent needs to recommend

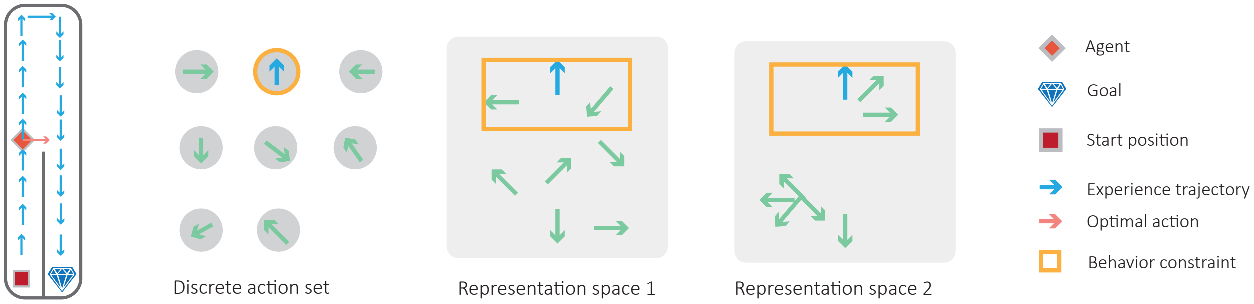

Figure 6. **An example of the influence of different action spaces on the behavioral regularizations of offline RL algorithms.**

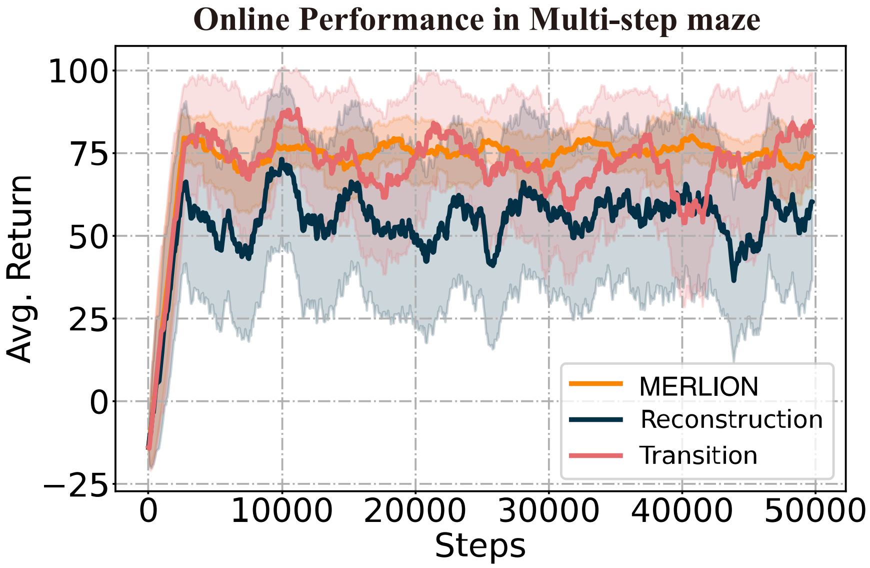

Figure 7. **Online performance of MERLION, the transition-based action representations, and the reconstruction-based representations in the Multi-step maze environment.**

different videos to the simulated users, and its goal is to extend the user's viewing time. We utilize the open-source platform Recsim (Ie et al., 2019) and adopt the provided scenario called `interest_evolution` as our simulated environment. There are 1000 candidate videos in the action set. All videos are split into 20 types, and each video has its quality and video length. The sampled user has different initial interest values ranging from -1 to 1 on different video types at each episode. The user would decide whether to view the recommended video according to the interest value of the video's type. The higher the interest value, the more likely the user is to view the video. The user has a latent attribute called time budget to control whether to continue to view another video. The episode ends when the time budget is smaller than 0. The initial time budget is 100, and it would be deducted by the video length if the user has viewed a video. In addition, the viewed video's quality would also influence the user's interest value and time budget: a high-quality video brings an increase, but a low-quality video leads to a decrease. To improve the diversity of the recommended videos, we deduct the time budget by 2.5 if two videos of the same type are recommended successively. The observed state at each time step is the users' interest values and the history of its viewed videos' types. A reward equaling the video length is given once one video is viewed.

Dialogue environment: Building dialogue systems for providing information (Young, 2006) or improving engagement (Li et al., 2016) is a real-world challenge. One solution is to employ RL to optimize dialog strategies in multi-turn dialog models. We adopt an open-source platform called convlab (Lee et al., 2019) to simulate a dialogue environment in which each action is a dialogue. We focus on the MultiWOZ domain (Eric et al., 2019). Its main task is to help a tourist in various situations involving multiple sub-domains, such as requesting basic information about attractions and booking a hotel room. Specifically, there are 7 sub-domains - Attraction, Hospital, Police, Hotel, Restaurant, Taxi, Train. We adopt dialogues in the top 1500 using frequencies as our action set. In detail, we utilize the provided scenario called `UserPolicyAgendaMultiwoz` to simulate the tourists' policies and use `MultiWozStateEncoder` to encode the histories of dialogs into vectorized states.

Data collection: Since there are no open-source datasets for the offline RL tasks with large discrete action spaces, we collected logged experience trajectories generated from online RL policies. These policies were all developed by the open-source implementation of the work (Chandak et al., 2020), which can be used to address tasks with large discrete action spaces by setting the hyperparameter of the action change number as 1. We trained policies until they achieved sub-optimal performance. Then, policies of checkpoints were used to collect transition data, and the noise was added to the action 50% of the time. In this way, we collected 100000 pieces of transition data in each environment.

A.5. Example

To better understand the influence of action spaces to the behavioral regularization of offline algorithms, we give a toy example iluustrated in Fig.6. The task is to reach the goal from the starting position as fast as possible, and collisions with the wall induce negative reward. The agent needs to learn a policy from the logged suboptimal (blue) trajectory. Assuming that a simple behavioral regularization scenario is applied to the target policy: the agent can only select actions within a particular range near the in-distribution action. At the agent's current position, we can find that different action spaces induce different results of the behavioral regularization. The target policy is limited to selecting the sub-optimal in-distribution action (move upward) in the discrete action space since other actions are isolated from it. By contrast, the policy is permitted to choose actions near the in-distribution action in the other representation spaces where there could be some other actions in the range of behavioral regularization. This is crucial to the policy improvement since the agent has the opportunity to choose the optimal action (move right). However, actions' relative distances also largely influence the results of the behavioral regularization. In representation space 1, behaviorally different actions (the transitions and rewards induced by them are different) are grouped together, so the agent is more likely to choose 'bad' actions. By contrast, if the relative distances between behaviorally similar actions are close (like in representation space 2), the agent tends to select the optimal or other 'safe' actions.

A.6. Online Performance of MERLION

MERLION action representations can be directly applied on the online setting. We use the framework mentioned in Chandak et al. (2019) which can incorporate action representaions and online RL algorithms. We get the online performance of MERLION, the transion-based representations, and the reconstruction-based representations in the Multi-step maze environment. The results are illustrated in Fig.7. We can find that, in the online setting, three action representations can all achieve fast convergence, while only MERLION can get good performance in the offline setting (Fig.2 (b)). This indicates that the MERLION is more effective in offline RL tasks with large discrete action spaces.