공학석사 학위논문

State Representation for Efficient Task Adaptation in Reinforcement Learning

강화학습에서의 효율적인 작업 적응을 위한 상태 표현

2023년 2월

서울대학교 대학원

협동과정 인공지능전공

양 은 석

State Representation for Efficient Task Adaptation in Reinforcement Learning

by

Eunseok Yang

A DISSERTATION

Submitted to the faculty of the Graduate School
in partial fulfillment of the requirements
for the degree of Master of Engineering
in the Department of Artificial Intelligence
Seoul National University
February 2023

State Representation for Efficient Task Adaptation in Reinforcement Learning

강화학습에서의 효율적인 작업 적응을 위한 상태 표현

지도교수 강 명 주

이 논문을 공학석사 학위논문으로 제출함

2022년 12월

서울대학교 대학원

협동과정 인공지능전공

양 은 석

양 은 석의 공학석사 학위논문을 인준함

2022년 12월

위 원 장 _____ (인)

부 위 원 장 _____ (인)

위 원 _____ (인)

Abstract

State Representation for Efficient Task Adaptation in Reinforcement Learning

Eunseok Yang

Interdisciplinary Program in Artificial Intelligence

The Graduate School

Seoul National University

An intelligent agent is expected to make a series of proper decisions in order to solve a new task by leveraging its own previous experience. The scheme of unsupervised reinforcement learning is analogous: the agent is equipped with generalized ability after it learns a set of potentially useful behaviors or extracts the information from dynamics without any explicit reward from the environment. However, a couple of major challenges remain such as how to obtain a compact yet rich state representations at the pretraining phase and how agents can efficiently adapt to the task at the fine-tuning phase. To this end, this study proposes two different methods to tackle both concerns. First, mixing discovered skills improve the sample efficiency by interpreting the skills as a perspective of how an agent transforms the state. The experiment shows that the various mixing methods affect the final performance. Second, contrastive learning plays a key role in temporal state representation which has an explicit meaning of reachability from one state to another.

It is shown that the agent can directly adapt to the given task without further training when it is optimized.

Key words: Reinforcement learning, unsupervised learning, representation learning, pretraining

Student Number: 2021-29014

Contents

Chapter 1

Introduction

Reinforcement learning (RL) has demonstrated great success in solving sequential decision making problems such as games and robotics manipulation [19, 31]. When interacting with the environment, the agent usually takes positive or negative feedback, which is called a reward, predefined depending on the task to solve, and its behavior is updated with trial-and-errors [33]. However, this approach results in poor performance when even slight changes occur in the task, which is different from human behavior.

The unsupervised reinforcement learning paradigm aims to address this issue. The agent first learns either a representation of state space, a set of possibly useful behaviors, or the environment dynamics without any signal during a pretraining phase, then adapts to the downstream task once it is given. There are a couple of lines of work: exploration-based methods [6, 26] and competence-based methods [10, 24].

The exploration-based methods aim to maximize the number of visiting state for discrete state space. When state space is continuous, the entropy of visited

space is estimated either directly or indirectly. The limitation of this approach is that the pretrained agent has only one strategy to cope with various unknown tasks. On the other hand, competence-based methods introduce a skill or task variable z to align each behavior. For both of these methods, state representation and task adaptation play an essential role.

State representation is a low-dimensional abstraction of a state that reflects the structures of state space and the relationship among states. A good state representation is crucial for sample efficiency and robustness and is achieved by extracting the core semantics and removing distracting noise. Besides, there are exploration methods to leverage the information from the state representation with such as reconstruction [16, 17], bisimulation [12, 7, 13], and contrastive learning [37, 11]. Theoretical works [1, 35] also exist on the impact of representation on performance.

Task adaptation is also critical in determining the performance of the pretrained agent. The task-agnostic agent has no straightforward way to find the optimal behavior for each task, and even for the skill-dependent agent, it is challenging to decide which and when to choose the learned skill. There are several basic approaches to solving this issue. For example, one [10] chooses the skill with the initial best performance and finetune, which does not guarantee the optimality of the chosen skill. Another work [30] considers planning with learned dynamics or utilizing the hierarchical controller choosing proper skill. A recent study [22] points out that the current methods need a better finetuning strategy to achieve a significantly better performance than training from scratch.

This dissertation focuses on both state representation and efficient adaptation, and proposes the following methods. Firstly, mixing discovered skills is key to ef-

ficient task adaptation. When there is a pretrained agent with finite skills, mixing those skills helps improve performance with a few additional computational burdens. Here, in the sense that state representation is changed depending on the skill, mixing skill is to create abundant representations. Sample efficiency is evaluated to determine which mixing is the best. Secondly, contrastive learning can be viewed as a universal reinforcement learning by estimating the future state distribution. The state representation contains a temporal meaning of reachability from one state to another, and this method offers the possibility of direct adaptation with a small number of task guidance.

1.1 Preliminaries

In the standard RL setting, the agent interacts with an unknown environment to learn the optimal behavior given sequential feedback. The environment is modeled with Markov Decision Process (MDP) [27] defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probability P, reward function $r : \mathcal{S} \to \mathbb{R}$, and discount factor $\gamma \in [0, 1)$. $\mathcal{S}$ and $\mathcal{A}$ are either discrete or continuous.

From the initial state $s_0 \in \mathbb{S}$, the agent takes action $a_0 \in \mathcal{A}$, and see reward $r_1 = r(s_1)$ with the next state $s_1 \sim P(s_1 \mid s_0, a_0)$. Sequentially the agent experiences the trajectory $\tau = \{s_0, a_0, r_1, s_1, \dots s_t, a_t, r_{t+1}, \dots\}$. The goal of the agent is to maximize the expected discounted cumulative reward (return) $\mathbb{E}[\sum_{t \geq 0} \gamma^t r_t]$. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, a mapping from state space to the distribution of action space, decides which action to take for each time step. It is called the optimal policy when it maximizes the return.

Value function measures the return depending on the current policy. A state

value function $V^{\pi}(s)$ of a state s is defined as

$$V^{\pi}(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, \pi\right]. \tag{1.1}$$

Similarly, an action value function (Q-function) $Q^{\pi}(s,a)$ of a state-action pair (s,a) under a policy π is defined as

$$Q^{\pi}(s,a) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a, \pi\right]. \tag{1.2}$$

The relationship between state value function and action-value function is clear with the following Bellman equations

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) Q^{\pi}(s,a) \tag{1.3}$$

$$Q^{\pi}(s,a) = \sum_{s'} P(s' \mid s,a) \left(r(s') + \gamma V^{\pi}(s')\right) \tag{1.4}$$

for discrete state and action space, and summations are replaced by integration for continuous case. The value function of optimal policy π^* is called the optimal value function, then it now satisfies the Bellman optimality equations

$$V^{\pi^*}(s) = \max_{a \in A} Q^{\pi^*}(s,a) \tag{1.5}$$

$$Q^{\pi^*}(s,a) = \sum_{s'} P(s' \mid s,a) \Big(r(s') + \gamma V^{\pi^*}(s')\Big). \tag{1.6}$$

In unsupervised reinforcement learning setting, the model is MDP without reward $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \gamma)$. Without an external reward function, the agent has to behave with its own intrinsic motivation. Therefore, the state value function and

the action value function are defined with the intrinsic reward. The method of designing intrinsic reward is one of the main goals of unsupervised reinforcement learning.

In task-dependent reinforcement learning, the policy depends on some vector z, denoted as $\pi(a \mid s, z)$. z is called a task, skill, or the goal depending on the context. In this case, the reward function r_z changes by z. Sometimes z consists of an explicit meaning (*e.g.* goal-conditioned RL [29]), but sometimes it is implicit (*e.g.* skill learning [14]).

Chapter 2

Related Work

In this chapter, detailed works on unsupervised reinforcement learning and contrastive learning are introduced.

2.1 Skill Learning

Skill learning [14, 10, 30] is a type of unsupervised reinforcement learning that allows the agent to learn various behaviors through interaction with the environment without explicit rewards. With a latent vector z, the skill dependent policy $\pi(a \mid s, z)$ is trained by maximizing mutual information between states S and skills Z,

$$\mathcal{I}(S; Z) = \mathcal{H}(Z) - \mathcal{H}(Z \mid S) \tag{2.1}$$

$$= \mathcal{H}(S) - \mathcal{H}(S \mid Z). \tag{2.2}$$

Intuitively, the first equation indicates that the agent prefers visiting as many state as possible, but the visited space is better determined by skill. The second

equation is derived from the symmetry of mutual information and demonstrates that the roles of state and skill can be exchanged.

Due to the difficulty of calculating the exact form of $I(S; Z)$, it is optimized by the variational lower bound as following

$$\mathcal{I}(S; Z) = D_{\mathrm{KL}}(p(s, z) \,\|\, p(s) \otimes p(z)) \tag{2.3}$$

$$= \mathbb{E}_{s,z\sim p(s,z)}[\log q_\phi(z \mid s)] - \mathbb{E}_{z\sim p(z)}[p(z)] + \mathbb{E}_{s\sim p(s)}[D_{\mathrm{KL}}(p(z \mid s) \,\|\, q_\phi(z \mid s))] \tag{2.4}$$

$$\geq \mathbb{E}_{s,z\sim p(s,z)}[\log q_\phi(z \mid s)] - \mathbb{E}_{z\sim p(z)}[p(z)], \tag{2.5}$$

where z is a discrete random variable. In this case, the skill-dependent reward is defined to $r_z(s) := \log q_\psi(z \mid s) - \log p(z)$. q_ψ learns to discriminate skills given the current state, then push the agent towards those states aligned with each skill. There are several extensions to use two consecutive states or whole trajectory instead of one state to match with one skill.

In another line of works, states are predicted given skills seen in equation (2.2). DADS [30], for example, learns dynamics to optimize the variational lower bound

$$\mathcal{I}(S'; Z \mid S) = \mathbb{E}_{s,s',z\sim p(s,s',z)}\left[\log \frac{q_\phi(s' \mid s, z)}{p(s' \mid s)}\right] + \mathbb{E}_{s,z\sim p(s,z)}\left[D_{\mathrm{KL}}(p(s' \mid s, z)) \,\|\, q_\phi(s' \mid s, z)\right] \tag{2.6}$$

$$\geq \mathbb{E}_{s,s',z\sim p(s,s',z)}\left[\log \frac{q_\phi(s' \mid s, z)}{p(s' \mid s)}\right] \tag{2.7}$$

with the skill-dependent reward $r_z(s) := \log q_\psi(s' \mid s, z) - \log p(s' \mid s)$.

The limitation of skill learning methods is that they lack how to utilize each skill for task although the agent learns various behaviors. Therefore, they perform

bad in a specific setting compared to other unsupervised learning methods [22].

2.2 Successor Features

Successor features [5] are the continuous extension of successor representation [9] for discrete state space. Assuming that state is represented to $\phi(s) \in \mathbb{R}^d$, and there is an important assumption that reward $r(s)$ is linear to this feature

$$r(s) = \phi(s)^T z, \tag{2.8}$$

where $z \in \mathbb{R}^d$ is a task vector. Then, the successor features with task-dependent policy π_z are defined as the discounted cumulative sum of the successive state features.

$$\psi^{\pi_z}(s,a) := \mathbb{E}\left[\sum_{t\geq 0} \gamma^t \phi(s_{t+1}) \mid (s_0,a_0) = (s,a), \pi_z\right] \tag{2.9}$$

The key observation here is that action value function is linear to the successor feature.

$$Q^{\pi_z}(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r(s_{t+1}) \mid (s_0,a_0) = (s,a), \pi_z\right] \tag{2.10}$$

$$= \psi^{\pi_z}(s,a)^T z \tag{2.11}$$

Denoting the successor feature $\psi^{\pi_z}(s,a)$ by $\psi(s,a,z)$ for simplification, multi-dimensional Bellman equation [3] holds for successor feature.

$$\psi(s,a,z) = \mathbb{E}_{s'\sim P(s'|s,a), a'\sim\pi_z(a'|s')}\left[\phi(s') + \gamma\psi(s',a',z)\right] \tag{2.12}$$

If the action space is discrete, the policy is learned with Q-learning [36] style. Otherwise, Q function and the policy is alternatively updated in DDPG [23] style. When the reward function is given after the policy is learned, the task vector z is simply calculated with solving linear regression:

$$z^* = \arg\min_{z \in \mathbb{R}^d} \mathbb{E}\left[(r(s) - \phi(s)^T z)^2\right]. \tag{2.13}$$

2.3 Contrastive Learning

The goal of contrastive learning [2] is to learn a representation by pushing the positive pairs closer and the negative pairs farther. The positiveness and negativeness are defined depending on the domain of problems. Firstly, in computer vision [8, 18], the positive sample of an image is an augmented version of it. Secondly, in natural language processing [25], the positive sample of an word is a context in which the word is related. Lastly, in reinforcement learning, pairs are defined in a visual or temporal sense. If a state is an image (pixel-based RL), same augmentation technique as in computer vision can be adopted to achieve a sample efficiency [21]. On the other hand, temporal criteria reflects a sequential nature of reinforcement learning [32]. In this case, the positive example of a state is one of its future state followed by a given policy. Recent work shows that goal-conditioned RL can be viewed as a contrastive learning [11]. Also, there is a theoretical work which shows contrastive state representation helps an efficient exploration [28].

Noise contrastive estimation (NCE) [15] is one of the famous contrastive learning method. Assume that a positive pair is sampled from a distribution $p(x, x_+)$, and

a negative pair is sampled from $p(x)p(x_-)$. Then the objective to maximize is

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,x_+)\sim p(x,x_+),x_-\sim p(x_-)} \left[\log \sigma(f_\theta(x, x_+)) + \log(1 - \sigma(f_\theta(x, x_-)))\right], \quad (2.14)$$

where σ is a sigmoid function, and f_θ is a parametrized estimation. When this binary classification objective is optimized, it is known that

$$f^*(x, x_+) \propto \log \frac{p(x, x_+)}{p(x)}. \quad (2.15)$$

With this property, contrastive learning is used for density estimation of data.

Chapter 3

Method

In this section, two main methods are introduced. The first method deals with the problem of efficient fine-tuning after the agent has learned diverse behaviors. The second method learns state representation with contrastive learning to capture the temporal relationship between states.

3.1 Efficient Task Adaptation by Mixing Discovered Skills

The main idea of the approach is to combine the learned skills to utilize for downstream tasks. The agent acquire skills in the pretrain phase following DIAYN [10] framework which is generally used as a skill discovery baseline.

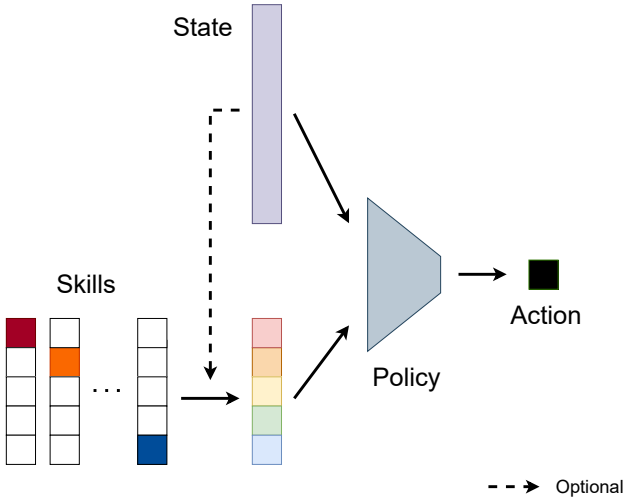

Figure 3.1: Overview of methods to mix a set of learned skills in fine-tuning phase. It is optional whether the current state affects how the skills combine together.

3.1.1 Understanding Skill Fusion

Note that the skill z is a k-dimensional discrete random variable sampled from the distribution $p(z)$. Then a policy can be written as

$$\pi(a|s) = \mathbb{E}_{z\sim p(z|s)}[\pi(a|s,z)]. \tag{3.1}$$

One can fine-tune a *controller* $p(z|s)$ as in a hierarchical RL framework [30], or simply sample skill from $p(z)$ [22] under the assumption that skill distribution is independent to the state distribution. In both cases, skill z is first sampled from $p(z)$ or $p(z|s)$, and action a is derived from the policy depending on the skill. The limitation of those methods is that they just sequentially perform skills rather than

Table 3.1: Fine-tuning method for skill discovery algorithm. Class shows whether the skill of each method is sequentially performed or combined. Each method samples the skill z from either $p(z)$ or $p(z|s)$, and finetune either only policy or all. Two previous works, *URLB DIAYN* [22] and *Original DIAYN* [10] are compared.

Class	Method	$p(z)$	$p(z\|s)$	Finetune
Sequential	DIAYN (URLB)	Uniform	-	$\pi(a\|s,z)$
Sequential	DIAYN (orig.)	Dirac delta	-	$p(z)$ & $\pi(a\|s,z)$
Fusion	Same (ours)	Uniform	-	$\pi(a\|s,z)$
Fusion	Simple (ours)	Cat.	-	$p(z)$ & $\pi(a\|s,z)$
Fusion	Scratch (ours)	-	Cat.	$p(z\|s)$ & $\pi(a\|s,z)$
Fusion	Controller (ours)	-	Cat. (init.)	$p(z\|s)$ & $\pi(a\|s,z)$

combine and mix the skills in an underlying representation level.

Instead, the policy can be formulated as

$$\pi(a|s, \mathbb{E}_{z\sim p(z|s)}[\psi(z)]) \tag{3.2}$$

for some representation function ψ. In this work, ψ is fixed as a one-hot vector representation of each discrete skill z in a natural manner. With this formulation, the skill space is expanded to the k-dimensional simplex $\{\tilde{z} \mid \mathbf{1}^\top \tilde{z} = 1, \tilde{z} \succeq 0\}$. In the following sections, state-agnostic perspective fusion and state-aware perspective fusion are introduced, respectively depending on which distribution the skill z follows, $p(z)$ or $p(z|s)$.

3.1.2 State-agnostic Fusion

First, let the skill follow the distribution $p(z)$ regardless of the input state. The simplest way of fine-tuning a policy is to choose one best skill, $p(z)$ is fixed as Dirac delta, as in the original DIAYN paper. However, it utilizes just one skill for a task

which has a clear limitation when the task is complicated that only one skill is not enough, also, it is often expensive to know which skill is best-performing especially when the skill set is large.

Now, two state-agnostic fusion methods proposed alleviate these issues. First, $p(z)$ is fixed to be uniform distribution, which combines the learned skills in the same weight. This simply takes advantage of mixing every skill without any additional parameters. Second, $p(z)$ is trained as a categorical distribution, with additional trainable parameter w_i for $\mathbb{E}_{z\sim p(z)}[\psi(z)] = \sum_i w_i \psi(z_i)$, where $\sum_i w_i = 1$ and $w_i \geq 0$. The latter is generalized version of the original DIAYN and the former method.

The representation level skill mix is interpreted as follows. Since the skill z is represented as a one-hot vector, the weight parameters in a policy are switched on up to the location of 1 in a skill vector. Therefore, if all positions of a skill vector are non-zero, all weights are activated and more diverse representations are obtained. We can view this as the same state from different perspectives through skill.

3.1.3 State-aware Fusion

In the method above, skill is combined without considering the input state. However, it is expected that better performance can be achieved if the combination of skills is changed adaptively according to the state. While it is possible to train the controller $p(z|s)$ from scratch, we propose a method to reuse a skill discriminator $q_\phi(z|s)$ trained in the pretraining phase. In DIAYN, the skill discriminator $q_\phi(z|s)$ predicts which skill is in charge of the input state so that each skill visits distinct states. Therefore, it is possible to guide the agent in the natural direction rather

than randomly combining skills by using a discriminator as an initializer of the controller.

3.2 Contrastive State Representation for Unsupervised RL

The core idea of the method is to learn contrastive state representations with which the agent can directly adapt to the given reward function.

3.2.1 Contrastive State Representation

First, the discounted state occupancy measure is defined.

$$p^{\pi_z}(s' \mid s, a) := (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_{t+1} = s' \mid (s_0, a_0) = (s, a)) \tag{3.3}$$

This measure has meaning how likely the state s' is visited starting from the state s and action a following the policy π_z. Also, assume the ratio of the state occupancy measure p^{π_z} to an arbitrary state measure ρ from which a negative sample is drawn is linearly factorizable.

$$\frac{p^{\pi_z}(s' \mid s, a)}{\rho(s')} = \psi^*(s, a, z)^T \phi^*(s') \tag{3.4}$$

Note that it is an extension of low-rank MDP, which assumes the transition $P(s \mid s, a)$ can be linearly factorized. Then it is possible to find the representations ψ^* and ϕ^* by contrastive learning argued in section 2.3. In detail, the equation 3.4

holds when the objective

$$\max_{f} \mathbb{E}_{\substack{t\sim Geo(1-\gamma), s_-\sim\rho(s) \\ s_+\sim p^{\pi_z}(s_t|s,a)}} \left[\log \frac{1}{1+f(s,a,z,s_-)} + \log \frac{f(s,a,z,s_-)}{1+f(s,a,z,s_-)} \right] \tag{3.5}$$

is optimized, where $f(s,a,z,s') = \psi(s,a,z)^T \phi(s')$. Similar approaches are introduced in recent works, but they only use for exploration or deal with goal-conditioned RL.

What matters now is which criteria the policy is updated on without the explicit reward function. The answer here is to choose an action to maximize

$$\psi^*(s,a,z)^T z \tag{3.6}$$

given a task z. The reason for it is clear with the following guarantee. [4] has shown that Q-function with respect to any reward function $r(s)$ is written as

$$\begin{aligned} Q^{\pi_z}(s,a) &= \int p^{\pi_z}(s' \mid s,a) r(s')\, ds' && (3.7)\\ &= \int \frac{p^{\pi_z}(s' \mid s,a)}{\rho(s')} \rho(s') r(s')\, ds' && (3.8)\\ &= \mathbb{E}_{s'\sim\rho}\left[\psi^*(s,a,z)^T \phi^*(s') r(s')\right] && (3.9)\\ &= \psi^*(s,a,z)^T \mathbb{E}_{s'\sim\rho}\left[\phi^*(s') r(s')\right]. && (3.10) \end{aligned}$$

Therefore, given a reward function, the optimal policy with respect to this is directly driven by choosing $z = \mathbb{E}_{s'\sim\rho}\left[\phi^*(s') r(s')\right]$ and act with $\arg\max_a \psi^*(s,a,z)^T z$.

Algorithm 1: Contrastive state representation

Input: number of episodes N_e, length of episode N_t, replay buffer (D), discount factor γ, initialized $\psi(s, a, z)$ and $\phi(s')$, and initialized policy π_θ

for $e = 1, \ldots, N_e$ **do**
  Sample $z \sim$ Unif(z)
  Observe the initial state s_1
  for $t = 1, \ldots, N_t$ **do**
    Select an action $a_t \sim \pi_\theta(a_t \mid s_t, z)$
    Observe the next state $s_{t+1} \sim P(s_{t+1} \mid s_t, a_t)$
    Sample $k \sim \text{Geo}(1 - \gamma)$
    Sample a batch $\{(s_i, a_i, s_{i+k}, \tilde{z})\}$ from the replay buffer $\mathcal{D}$
    Update $\psi(s, a, z)$ and $\phi(s')$ with the objective (3.5)
    Update the policy π_θ with the objective (3.6)
  end
  Store z-augmented trajectory $(z; s_1, a_1, s_2, a_2, \ldots, s_{N_t})$ to the replay buffer $\mathcal{D}$
end

Chapter 4

Experiment

Note that the experiments are conducted for the methods in Section 3.1.

4.1 Experiments

The methods are evaluated on Deepmind Control Suite [34], which contains three continuous control environments and twelve downstream tasks. The environments include 6-DOF *Walker*, 12-DOF *Quadruped*, and 9-DOF *Jaco arm*.

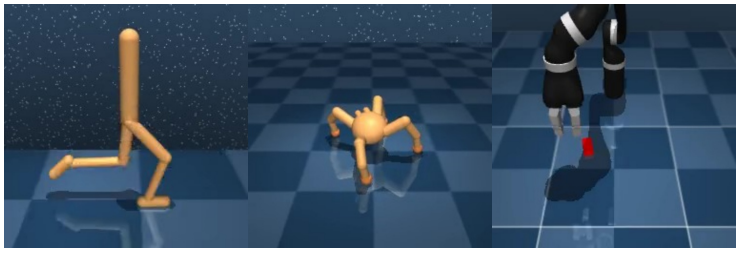

Figure 4.1: Continuous control environments from DeepMind Control Suite. Left: Walker, center: Quadruped, right: Jaco arm.

The experiments follow the same evaluation process to URLB [22]. We pretrain

each agent without explicit rewards, then finetune for the downstream task. In the case of skill discovery method, the agent learns a set of useful behaviors which we call skills. Then the agent uses these skills to achieve higher cumulative rewards quickly in fine-tuning. The pretrained agent is fixed to DIAYN based on DDPG [23] and finetune it in various ways.

4.1.1 Sample-efficiency and Final Performance

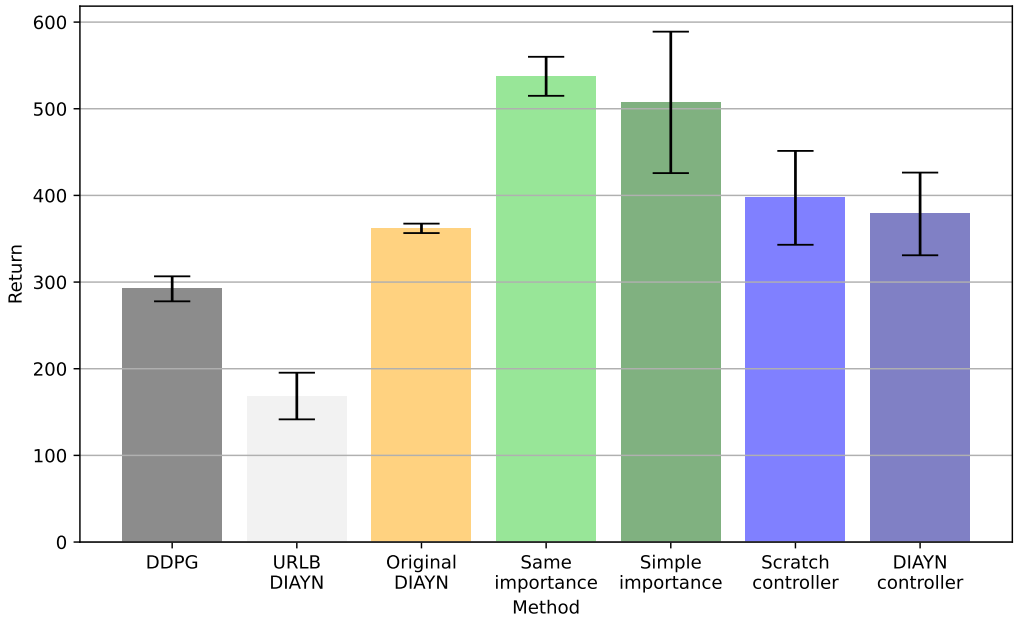

Figure 4.2: Results of 100k steps finetuning of skill adapting methods on the task Walker Run. Simple importance and Same importance method shows better sample efficiency than the others.

We compare our methods to three baselines, *DDPG* trained from scratch, DIAYN implemented in URLB (*URLB DIAYN*) and DIAYN proposed in the original paper (*Original DIAYN*). For *Original DIAYN*, we did not count the steps for se-

lecting the best skill in order to make a baseline more challenging. Also, we choose the task Walker-run for the comparison because it is known as the most challenging task among the twelve downstream tasks in URLB [22]. We fine-tune each method for 100k steps. The experiment in other environments is shown in section 4.1.2.

Same importance and *Simple importance* are both state-agnostic fusion methods, but the former fixes $p(z)$ and latter finetunes. *DIAYN controller* initializes a controller $p(z|s)$ with DIAYN skill discriminator, but *scratch controller* does not. The detail of our methods is in Table 3.1.

We observe the following results in Figure 4.2. *URLB DIAYN* underperforms other methods even including DDPG trained from scratch, showing that it does not fully use the potential of learned skills, and suggesting that fine-tuning is critical to measure the performance of the skill discovery method. On the other hand, both of our state-agnostic methods perform well, showing that the simple skill mix methods help to achieve high return quickly. The performance of state-aware methods is worse than state-agnostic methods and similar to the baseline (*Original DIAYN*) at 100k step. We presume training additional parameters for controller disturbs the fast adaptation.

We also compare our methods for the final performance. We show in figure 4.3 that state-aware fusion method outperforms state-agnostic fusion at some point. Followed by a certain amount of steps, the capacity of state-aware methods works. Moreover, we observe that *DIAYN controller* outperforms the *Scratch controller*. This shows that the DIAYN discriminator trained at pretrained stage is a good initializer for the skill controller. This transferred weight not only helped better performance at the final step, but also helped robust training which has lower variance.

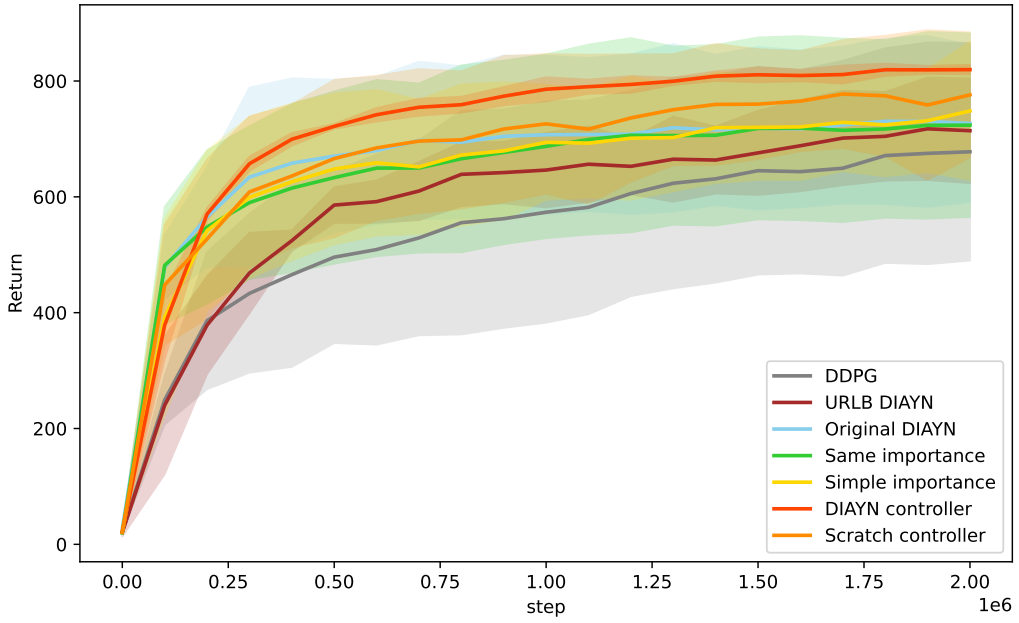

Figure 4.3: 2M steps finetuning results on *Walker run* task. State-agnostic methods shows fast learning, but state-aware method achieves higher return at the end.

4.1.2 Comparison to Other URLB Methods

We evaluate how quickly the agent adapt to twelve downstream tasks in the same setting to URLB [22], 2M pre-train steps and 100k fine-tune steps. Among our proposed methods, as we have seen on the experiment above, *same importance* method is the most sample-efficient. Therefore, we finetune pretrained DIAYN with *same importance* for 3 seeds per task. We compared our method with 9 unsupervised RL algorithms including DIAYN with a vanilla fine-tuning. Expert performance is a DDPG result after 2M steps of fine-tuning [20]. *Other best* is a state-of-the-art among twelve unsupervised RL reported in URLB.

As shown in Table A.1, although the simple fine-tune method without any

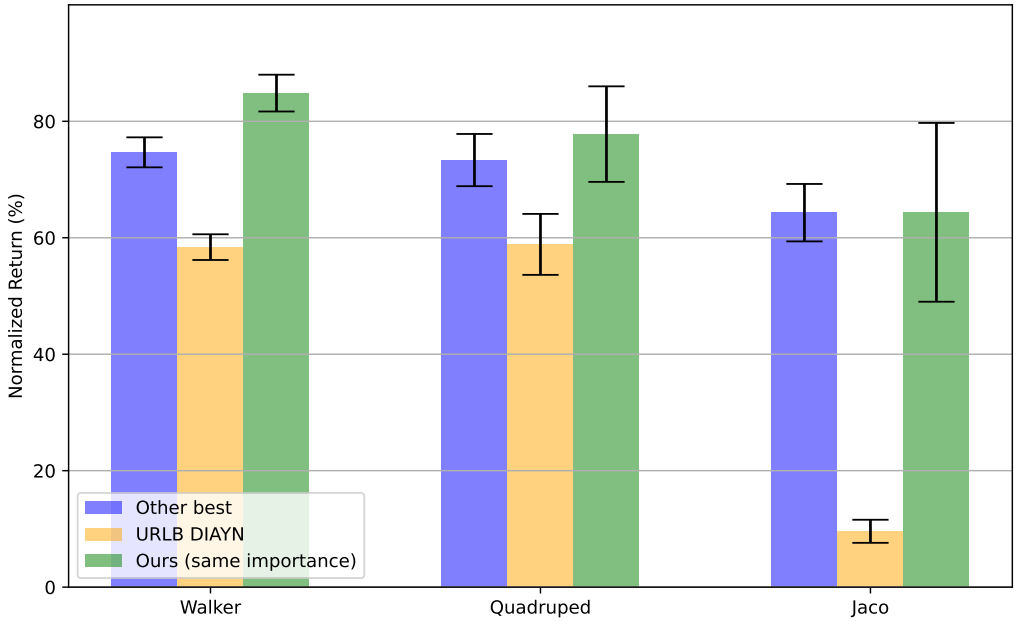

Figure 4.4: Results of finetuning for 100k on twelve tasks and three environments. Returns are normalized by the expert score which is trained for 2M steps with DDPG.

additional weight is applied, it outperforms every other methods in 9 out of 12 tasks. Even for the tasks that other methods work better(Quadruped-stand, Jaco-reach-bottom-right, Jaco-reach-top-right), proposed method just slightly underperforms. Especially, it achieves an enormous improvement compared to the reported original DIAYN. From this experiment, we can recognize how important the fine-tuning method is for evaluating a skill discovery methods.

Chapter 5

Conclusion

This work studies two unsupervised reinforcement learning methods through the lens of state representation and efficient task adaptation. Skill mixing method hypotheses that different skill converts a state into a different representation. Additional state representation is derived by expanding skill space continuously by mixing skills. The experiments show that this helps improve performance by a high margin. The contrastive method shows that the temporal relationship between states can be learned with contrastive learning. It is possible for the agent to directly adapt to the downstream task by iteratively updating the representations and policy. The experimental guarantee remains for future work.

Bibliography

[1] A. Agarwal, S. Kakade, A. Krishnamurthy, and W. Sun, *Flambe: Structural complexity and representation learning of low rank mdps*, Advances in neural information processing systems, 33 (2020), pp. 20095–20107.

[2] S. Becker and G. E. Hinton, *Self-organizing neural network that discovers surfaces in random-dot stereograms*, Nature, 355 (1992), pp. 161–163.

[3] R. Bellman, *Dynamic programming*, Science, 153 (1966), pp. 34–37.

[4] L. Blier, C. Tallec, and Y. Ollivier, *Learning successor states and goal-dependent values: A mathematical viewpoint*, arXiv preprint arXiv:2101.07123, (2021).

[5] D. Borsa, A. Barreto, J. Quan, D. Mankowitz, R. Munos, H. Van Hasselt, D. Silver, and T. Schaul, *Universal successor features approximators*, arXiv preprint arXiv:1812.07626, (2018).

[6] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, *Exploration by random network distillation*, arXiv preprint arXiv:1810.12894, (2018).

[7] P. S. Castro and D. Precup, *Using bisimulation for policy transfer in mdps*, in Twenty-Fourth AAAI Conference on Artificial Intelligence, 2010.

[8] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, *A simple framework for contrastive learning of visual representations*, in International conference on machine learning, PMLR, 2020, pp. 1597–1607.

[9] P. Dayan, *Improving generalization for temporal difference learning: The successor representation*, Neural computation, 5 (1993), pp. 613–624.

[10] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, *Diversity is all you need: Learning skills without a reward function*, arXiv preprint arXiv:1802.06070, (2018).

[11] B. Eysenbach, T. Zhang, R. Salakhutdinov, and S. Levine, *Contrastive learning as goal-conditioned reinforcement learning*, arXiv preprint arXiv:2206.07568, (2022).

[12] N. Ferns, P. Panangaden, and D. Precup, *Metrics for finite markov decision processes.*, in UAI, vol. 4, 2004, pp. 162–169.

[13] C. Gelada, S. Kumar, J. Buckman, O. Nachum, and M. G. Bellemare, *Deepmdp: Learning continuous latent space models for representation learning*, in International Conference on Machine Learning, PMLR, 2019, pp. 2170–2179.

[14] K. Gregor, D. J. Rezende, and D. Wierstra, *Variational intrinsic control*, arXiv preprint arXiv:1611.07507, (2016).

[15] M. Gutmann and A. Hyvärinen, *Noise-contrastive estimation: A new estimation principle for unnormalized statistical models*, in Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 297–304.

[16] D. Ha and J. Schmidhuber, *World models*, arXiv preprint arXiv:1803.10122, (2018).

[17] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, *Dream to control: Learning behaviors by latent imagination*, arXiv preprint arXiv:1912.01603, (2019).

[18] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, *Momentum contrast for unsupervised visual representation learning*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 9729–9738.

[19] J. Kober, J. A. Bagnell, and J. Peters, *Reinforcement learning in robotics: A survey*, The International Journal of Robotics Research, 32 (2013), pp. 1238–1274.

[20] M. Laskin, H. Liu, X. B. Peng, D. Yarats, A. Rajeswaran, and P. Abbeel, *Cic: Contrastive intrinsic control for unsupervised skill discovery*, arXiv preprint arXiv:2202.00161, (2022).

[21] M. Laskin, A. Srinivas, and P. Abbeel, *Curl: Contrastive unsupervised representations for reinforcement learning*, in International Conference on Machine Learning, PMLR, 2020, pp. 5639–5650.

[22] M. Laskin, D. Yarats, H. Liu, K. Lee, A. Zhan, K. Lu, C. Cang, L. Pinto, and P. Abbeel, *Urlb: Unsupervised reinforcement learning benchmark*, arXiv preprint arXiv:2110.15191, (2021).

[23] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, *Continuous control with deep reinforcement learning*, arXiv preprint arXiv:1509.02971, (2015).

[24] H. Liu and P. Abbeel, *Aps: Active pretraining with successor features*, in International Conference on Machine Learning, PMLR, 2021, pp. 6736–6747.

[25] A. Mnih and K. Kavukcuoglu, *Learning word embeddings efficiently with noise-contrastive estimation*, Advances in neural information processing systems, 26 (2013).

[26] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, *Curiosity-driven exploration by self-supervised prediction*, in International conference on machine learning, PMLR, 2017, pp. 2778–2787.

[27] M. L. Puterman, *Markov decision processes*, Handbooks in operations research and management science, 2 (1990), pp. 331–434.

[28] S. Qiu, L. Wang, C. Bai, Z. Yang, and Z. Wang, *Contrastive ucb: Provably efficient contrastive self-supervised learning in online reinforcement learning*, in International Conference on Machine Learning, PMLR, 2022, pp. 18168–18210.

[29] T. Schaul, D. Horgan, K. Gregor, and D. Silver, *Universal value function approximators*, in International conference on machine learning, PMLR, 2015, pp. 1312–1320.

[30] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman, *Dynamics-aware unsupervised discovery of skills*, arXiv preprint arXiv:1907.01657, (2019).

[31] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., *Mastering the game of go with deep neural networks and tree search*, nature, 529 (2016), pp. 484–489.

[32] A. Stooke, K. Lee, P. Abbeel, and M. Laskin, *Decoupling representation learning from reinforcement learning*, in International Conference on Machine Learning, PMLR, 2021, pp. 9870–9879.

[33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[34] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al., *Deepmind control suite*, arXiv preprint arXiv:1801.00690, (2018).

[35] M. Uehara, X. Zhang, and W. Sun, *Representation learning for online and offline rl in low-rank mdps*, arXiv preprint arXiv:2110.04652, (2021).

[36] C. J. Watkins and P. Dayan, *Q-learning*, Machine learning, 8 (1992), pp. 279–292.

[37] Y. Wu, G. Tucker, and O. Nachum, *The laplacian in rl: Learning representations with efficient approximations*, arXiv preprint arXiv:1810.04586, (2018).

Appendix A

Miscellaneous

Table A.1: Result of fine-tuning for 1×10^5 frames after pre-training for 2×10^6 frames.

Domain	Task	Expert	Other best	URLB DIAYN	Ours (same imp.)
Walker	Flip	799	515±17	381±17	**658±51**
	Run	796	439±34	242±11	**537±22**
	Stand	984	923±9	860±26	**936±11**
	Walk	971	828±29	661±26	**917±23**
Quadruped	Jump	888	590±33	578±46	**645±20**
	Run	888	465±37	415±28	**558±43**
	Stand	920	**840±33**	706±48	719±158
	Walk	866	721±56	406±64	**845±74**
Jaco	bottom left	193	134±8	17±5	**136±36**
	bottom right	203	**122±4**	31±4	119±39
	top left	191	124±20	11±3	**127±9**
	top right	223	**140±7**	19±4	138±42

A.1 Results

We compare our method *same importance* to *expert* which is trained by DDPG with 2M steps from [20], *other best* which shows the best performance among 9

unsupervised RL algorithms in URLB [22], and *URLB DIAYN*.

A.2 Hyperparameters

We present the best performing hyperparamter borrowed from URLB [22].

Table A.2: Hyperparameters in our experiments.

DDPG hyperparameter	Value
Replay buffer capacity	10^6
Action repeat	1
Seed frames	4000
n-step returns	3
Mini-batch size	24
Seed frames	4000
Discount (γ)	0.99
Optimizer	Adam
Learning rate	10^{-4}
Agent update frequency	2
Critic target EMA rate (τ_Q)	0.01
Features dim.	1024
Hidden dim.	1024
Exploration stddev clip	0.3
Exploration stddev value	0.2
Number pre-training frames	2×10^6
Number fine-tuning frames	1×10^5
DIAYN hyperparameter	**value**
Skill dim	16
Skill sampling frequency (steps)	50
Discriminator net arch.	$512 \to 1024 \to 1024 \to 16$ ReLU MLP

국문초록

지능형 에이전트는 자신의 이전 경험을 활용하여 새로운 작업을 해결하기 위해 일련의 적절한 결정을 내릴 것으로 예상된다. 이는 비지도 강화학습 체계와 유사한데, 에이전트는 환경으로부터 명시적인 보상 없이 잠재적으로 유용한 행동들을 학습하거나 환경에서 정보를 추출한 후 일반화된 능력을 갖추게 된다. 그러나 사전학습 단계에서 어떻게 간결하면서도 풍부한 상태 표현을 얻을 것인지, 그리고 미세조정 단계에서 어떻게 에이전트가 작업에 효율적으로 적응할 수 있을지에 관한 주요 과제가 남아있다. 이를 위해 본 연구에서는 두 가지 과제를 모두 해결하기 위한 두 개의 서로 다른 방법을 제안한다. 첫째, 발견된 기술을 혼합함으로써 에이전트가 상태를 변환하는 방법에 대한 관점으로 기술을 해석하여 샘플 효율성을 향상시킨다. 실험 결과 다양한 혼합 방법이 최종 성능에 영향을 미치는 것으로 나타났다. 둘째, 대조 학습은 어떤 상태에서 다른 상태로의 도달 가능성에 대한 명시적인 의미를 갖는 시간적 상태 표현에 핵심적인 역할을 한다. 에이전트가 최적화될 때 주어진 작업에 직접 적응할 수 있는 것으로 나타났다.

주요어휘: 강화학습, 비지도학습, 표현학습, 사전학습

학번: 2021-29014