CPeSFA: Empowering SFs for Policy Learning and Transfer in Continuous Action Spaces

Yining Li [1] **Tianpei Yang** [2] **Wei Guo** [1] **Jianye Hao** [1] **Yan Zheng** [1]

Abstract

Successor Features (SFs), together with Generalized Policy Improvement (GPI), comprise a conventional transfer Reinforcement Learning (RL) algorithm, which can transfer knowledge using the characteristic of decoupling policy with the task. However, SFs are value-based and cannot handle environments with continuous action spaces since GPI cannot transfer knowledge by traversing all possible actions in such a case. Recently, PeSFA (Li et al., 2022) decouples SFs from policies and further endows SFs with generalization capabilities in the policy space. However, it cannot be applied to continuous action spaces. In this paper, we introduce the Continuous PeSFA (CPeSFA) algorithm, an Actor-Critic (AC) architecture designed for learning and transferring policies in continuous action spaces. Our theoretical analysis shows that CPeSFA leverages SFs' generalization in the policy space to accelerate learning rate. Experimental results across Grid World, Reacher, and Point Maze environments demonstrate CPeSFA's superiority and effective knowledge transfer for rapid policy learning in new tasks.

1. Introduction

Reinforcement Learning (RL), a prominent technique within the field of artificial intelligence, addresses the challenge of optimal continuous control and has achieved notable success in the field of games, robot control, etc (Mnih et al., 2015; Silver et al., 2016). In RL, an agent dynamically interacts with the environment (Sutton & Barto, 1998; Lillicrap et al., 2016) to gather essential data. Through experiential knowledge, the agent continually learns the optimal policy for the task, endeavoring to maximize the long-term cumulative rewards within the environment. However, when the interaction cost between the agent and the environment is excessive or the exploration complexity is significant, the agent may encounter difficulties in swiftly acquiring the optimal policy (Hao et al., 2023). Transfer learning offers a compelling resolution to challenges tied to sample inefficiency and sluggish learning rates (Zhu et al., 2020; Taylor & Stone, 2009; Yang et al., 2020). Transfer RL can accelerate the learning of the target task by transferring knowledge from past-learned tasks to the target task, mitigating the challenges of sample inefficiency.

One major category of transfer RL algorithms is based on Successor Features (SFs) (Barreto et al., 2017), which addresses knowledge transfer across scenarios where the dynamics of the environment remain unchanged across different tasks, but the reward functions vary between tasks. SFs decompose the reward function into dynamic-related features and task-specific weights, thereby decoupling the policy from the task. By leveraging SFs learned from historical tasks and the task-specific weights of the target task, the action-value function of the target task can be quickly computed. On this basis, Generalized Policy Improvement (GPI) (Barreto et al., 2017) is employed to select the best action for the target task, effectively transferring knowledge from historical tasks.

Existing classical SFs algorithms leverage SFs' ability to quickly compute action-value functions for historical policies on new tasks, aiding the transfer of historical policies, while GPI is utilized to derive policies with basic performance on new tasks. These SFs methods can be broadly categorized into four types.

The first category of methods (Alver & Precup, 2022; Alegre et al., 2022; Nemecek & Parr, 2021) primarily enhances the performance of policies obtained through GPI method in downstream tasks by constructing a comprehensive set of source policies. However, these approaches face challenges in dealing with more complex environments or task settings, as the complexity of constructing a thorough source task set become prohibitive. The second category (Touati et al., 2023; Kim et al., 2022; Tang et al., 2023) focuses on

[1]College of Intelligence and Computing, Tianjin University, Tianjin, China [2]University of Alberta, Edmonton, Alberta, Canada. Correspondence to: Jianye Hao <jianye.hao@tju.edu.cn>, Tianpei Yang <tpyang@tju.edu.cn>.

Workshop on Theoretical Foundations of Reinforcement Learning and Control at the 41^{st} International Conference on Machine Learning, Vienna, Austria.

leveraging the zero-shot capability of GPI methods based on SFs. However, these approaches primarily rely on the knowledge transfer capability of GPI methods for historical policies, which is limited by the inherent characteristics of GPI methods, such as difficulties in transferring knowledge in continuous action spaces. The third category (Han & Tschiatschek, 2022; Garces et al., 2023) explores methods to alleviate the constraints imposed by SFs on the setting of environment dynamics. While these methods may alleviate the limitations of SFs on environmental dynamics, they introduce additional constraints on other environmental settings, limiting their applicability.

The fourth category of methods (Borsa et al., 2019; Hansen et al., 2020; Liu & Abbeel, 2021) introduce additional inputs to SFs to enhance their generalization, which is also the type to which our approach belongs. This category of methods exhibits greater versatility compared to others and can be easily adapted to various RL architectures with simple modifications. Moreover, their extended generalization of SFs endows them with exceptional attributes, enhancing learning, knowledge transfer, and applicability across diverse scenarios.

Although the aforementioned SFs algorithms can leverage the characteristics of SFs to transfer knowledge through GPI methods or rapidly compute action-value functions across tasks, SFs have inherent drawbacks (Lehnert et al., 2017): 1) SFs cannot ensure that the policy produced by the GPI algorithm is optimal for the target task, necessitating further exploration and learning in the target task. 2) SFs employ a value-based RL framework, which fails to handle tasks with continuous action spaces. In continuous action spaces, the inability to traverse the action space prevents the use of GPI algorithm for transferring historical knowledge. 3) Due to the strong dependence of SFs' transferability on historical policies and the fact that the policies obtained by GPI are not optimal for the target task, negative transfer effects can easily occur in new tasks. Therefore, it is necessary to decouple SFs from policies and facilitate knowledge transfer based on generalization within the policy space.

PeSFA (Li et al., 2022), on the other hand, decouples policies from SFs by incorporating policy representations as additional inputs to SFs. This enables PeSFA to leverage its generalization capabilities in both the policy and task spaces, utilizing knowledge from historical policies to expedite learning in the target task.

However, PeSFA still follows a value-based architecture and is limited to discrete action spaces. To overcome this challenge, in this paper, we modify the value-based PeSFA framework to Actor-Critic (AC) architecture (Grondman et al., 2012) known as Continuous PeSFA (CPeSFA). This modification allows CPeSFA to effectively tackle control problems in continuous action spaces. CPeSFA adapts the Critic network in the AC architecture to the structure of PeSFA, enabling its straightforward application to various AC architecture-based RL algorithms.

Moreover, CPeSFA addresses a limitation of SFs in continuous action spaces, where SFs cannot utilize the GPI algorithm for effective knowledge transfer. By decoupling SFs from both policy and task, CPeSFA achieves a dual level of generalization at the policy and task levels. We further demonstrate that CPeSFA, leveraging policy-level generalization accelerates the learning rate. CPeSFA also efficiently leverages historical policy knowledge after task switching to expedite learning in the new task. Our experimental results also demonstrate that CPeSFA can accelerate the learning rate and effectively utilize the transfer of historical knowledge to expedite policy learning on new tasks.

Our contributions are summarized as follows: (1) We extend PeSFA from value-based to AC architecture, creating CPeSFA, enabling it to learn policies in continuous action spaces and transfer historical knowledge; (2) We conducted theoretical analyses, demonstrating that CPeSFA can leverage the generalization property of SFs in the policy space to expedite policy learning. (3) We conducted experimental comparisons of CPeSFA with baseline algorithms in various environments, confirming the effectiveness of our approach.

2. Related work

Construction of a complete source policy set for SFs. Some methods (Alver & Precup, 2022; Alegre et al., 2022; Nemecek & Parr, 2021) aim to construct comprehensive source policy sets to facilitate the knowledge of source tasks and address more complex downstream tasks using GPI methods. SIP (Alver & Precup, 2022) and SFOLS (Alegre et al., 2022) are approaches that construct relatively comprehensive policy sets. They use policy sets along with GPI methods to address complex downstream tasks efficiently. Policy Caches (Nemecek & Parr, 2021) compute upper bounds for the current policy sets in a new task, determining whether to utilize existing SFs with GPI methods or relearn policies for the task. However, these methods primarily focus on improving GPI's performance by creating source task sets. In complex task settings or environments, constructing these source task sets can be intricate and lead to significant overhead.

Zero-shot capability of SFs. SFs decouples policy and task, enabling quick inference of the action-value function for the source policy in target task. With GPI method, we can obtain a well-performing policy for the target task. Therefore, many methods concentrate on exploring zero-shot capabilities of SFs. Zero-shot SFs (Touati et al., 2023) explored the zero-shot performance of SFs under various dynamic feature learning methods. In this approach, some

experiments were conducted in continuous action space environments without using transfer methods. Instead, they directly employed a goal-dependent actor network to replace the typical action selection mechanism in value-based methods. Constrained GPI (Kim et al., 2022) focuses on constraining SFs with tighter bounds to limit fitting errors. These bounds are then used during policy learning for the new task, reducing fitting errors and accelerating the learning process. Composition SFs (Liu & Ahmad, 2023) primarily aim at integrating value composition with SFs in a multi-task setting. This integration allows SFs to aggregate knowledge from multiple historical policy sets, effectively leveraging policy knowledge across multiple tasks. However, these methods mainly leverage the decoupling property of SFs between tasks and policies, obtaining a well-performing policy using the GPI method. In continuous action spaces, the inability to traverse the action space hinders the GPI method from fully leveraging its zero-shot capabilities.

Applying SFs in different dynamic seting. Traditional SFs are typically applied in environments where dynamic remain constant but task weights vary. However, some works (Han & Tschiatschek, 2022; Garces et al., 2023) have relaxed this constraint, allowing for knowledge transfer across environments with varying dynamic. TSF (Garces et al., 2023) overcomes limitations of SFs by allowing knowledge transfer across different dynamic environments. It efficiently transforms SFs based on the base dynamic, projecting varied dynamics into a common dynamic distribution. Abstract Successor Option (Han & Tschiatschek, 2022) represents options in the form of SFs, facilitating knowledge transfer of abstract successor options across various environments through shared features. However, these methods are primarily designed for scenarios where dynamics or the environment undergo changes. In contrast, this paper focuses on the traditional setting where the environment dynamics remain constant. Therefore, the research direction of these methods is orthogonal to that of this paper.

Generalization of SFs on Extended Inputs. SFs can also incorporate additional inputs, such as task weights or policy representations, allowing SFs to generalize at the task or policy level and leverage this generalization for knowledge transfer. USFA (Borsa et al., 2019), based on UVFA (Schaul et al., 2015), introduces task weights as an extra input to model the correspondence between task weights and optimal policies. This enables SFs to assist in transfer through the utilization of source policy knowledge by sampling task weights from the task weight space that are similar to the target task during GPI. VISR (Hansen et al., 2020) and APS (Liu & Abbeel, 2021) methods introduce diversity criteria to SFs and use skills as an extra input to the SFs network. They employ a method focused on maximizing behavioral mutual information to learn various skills and policies. PeSFA (Li et al., 2022), on the other hand, introduces policy representations as additional inputs to SFs. This empowers SFs to generalize at both the policy and task levels. However, these methods are primarily designed for settings with discrete action spaces. In continuous action spaces, the value-based SFs algorithm used by these methods struggles to learn the optimal policy.

Policy Representation. The PeVFA (Tang et al., 2022) algorithm takes policy representations as additional inputs to the value function, enabling the value function to generalize at the policy level. This allows for a more accurate estimation of the value corresponding to the optimized policy, providing a better starting point for learning the true value of the policy and thus accelerating the overall learning speed. Off-policy PeVFA (Zhang et al., 2023) extends PeVFA to off-policy reinforcement learning algorithms and adaptively modifies the training approach of PeVFA based on the characteristics of off-policy algorithms.

3. Preliminaries

3.1. Reinforcement Learning

The main objective of reinforcement learning is to enable an agent to learn a policy through interaction with the environment to maximize the obtained reward, and this process can be represented as a Markov Decision Process (MDP) (Feinberg, 1996).

An MDP can be represented in the form of a quintuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $r(s, a)$ is the reward function, representing the reward obtained by taking action a in state s, and $p(\cdot \mid s, a)$ is the transition function, and γ is the discount factor (Feinberg, 1996).

Soft Actor-Critic (SAC) (Haarnoja et al., 2018) is a reinforcement learning algorithm framework based on maximum entropy. The objective of SAC is to maximize the entropy of the policy at each time step on top of maximizing the reward, enhancing the agent's exploration efficiency by making the policy more random. The Soft Q function is defined as follows:

$$Q^{\pi}(s, a) = r(s, a) + \gamma \mathbb{E}[Q(s', a') - \alpha \log \pi(a' \mid s')] \quad (1)$$

where α is the entropy weight. The objective functions of the actor in SAC is:

$$J_Q(\theta_Q) = \mathbb{E}\big[\big(Q_{\theta_Q}(s, a) - (r + \gamma \mathbb{E}[Q_{\bar{\theta_Q}}(s', a') - \alpha \log \pi(a' \mid s')])\big)^2\big] \quad (2)$$

The objective functions of the critic in SAC is:

$$J_{\pi}(\theta_a) = \mathbb{E}[\alpha \log \pi_{\theta_a}(a \mid s) - Q_{\theta_Q}(s, a)] \quad (3)$$

where θ_Q and θ_a are network parameters of the critic and actor network, $\bar{\theta_Q}$ is the target network parameters of the critic network.

3.2. Successor Features

In the setting of Successor Features (Barreto et al., 2017), the environment dynamics of the MDP are the same across different tasks, but the reward functions vary. The reward function is divided into two parts: environment-related feature ϕ and task-related weight w. Therefore, the reward function can be represented as $r_{\mathrm{w}}(s,a) = \phi(s,a,s')^{\top}\mathrm{w}$.

According to the definition of the reward function, the action-value function can be rewritten as follows:

$$Q^{\pi}(s,a) = \mathbb{E}^{\pi}\left[\sum_{i=t}^{\infty}\gamma^{i-t}\phi_{i+1} \mid S_t = s, A_t = a\right]^{\top}\mathrm{w} = \psi^{\pi}(s,a)^{\top}\mathrm{w} \tag{4}$$

Where ψ^{π} represents the SFs under the control of policy π, and w is the task-related weight. Since the SFs decouples the policy from the task, when the task is switched, the action-value function for a policy π under any task w' can be quickly calculated based on the SFs and the task weight w' for the new task, i.e., $Q^{\pi}_{\mathrm{w}'}(s,a) = \psi^{\pi}(s,a)\mathrm{w}'$.

3.3. Policy-extended Successor Features

PeSFA (Li et al., 2022) builds upon SFs by introducing the policy representation as an additional input, enabling SFs to have generalization at the policy level. Assuming a mapping function for policy representation $g : \Pi \to \chi \in \mathbb{R}^n$, any policy $\pi \in \Pi$ can be mapped to an n-dimensional representation $\chi_\pi = g(\pi)$. By incorporating policy representation as an additional input to SFs, we obtain PeSF as $\psi(s,a,\pi) = \psi^{\pi}(s,a)$, and the action-value function can be expressed as follows:

$$Q^{\pi}(s,a,\mathrm{w}) = \psi^{\pi}(s,a)^{\top}\mathrm{w} = \psi(s,a,\chi_\pi)^{\top}\mathrm{w} \tag{5}$$

We refer to $\tilde{\psi}(s,a,\chi_\pi) \approx \psi(s,a,\chi_\pi)$ as PeSFA. PeSFA takes the policy representation as additional input, decoupling SFs from the policy. It essentially fits the SFs corresponding to various policies in the policy representation space.

4. Continuous Policy-extended Successor Features

Traditional SFs employ a value-based RL architecture, which is more conducive to leveraging SFs' property of decoupling policy from the task. However, this method has a drawback: it cannot learn policies or transfer knowledge in environments with continuous action spaces. In continuous action spaces, due to the inability to traverse all possible actions directly, GPI methods cannot be used to select the optimal action for a given state based on the Q network. This limitation hinders SFs from harnessing the advantage of knowledge transfer across tasks in an environment with continuous action space. Additionally, the value-based architecture struggles with continuous control problems when discretizing action spaces, as this method fails to achieve precise control.

Therefore, we propose the Continuous PeSFA (CPeSFA) algorithm based on the Actor-Critic (AC) architecture, extending the traditional value-based SFs algorithm to continuous action spaces. Furthermore, CPeSFA leverages the generalization capabilities of CPeSFA in the policy space to achieve knowledge transfer, a challenge that traditional SFs struggle with in continuous action spaces. As CPeSFA is an AC-based algorithm, it can seamlessly be applied to any AC-based reinforcement learning algorithm, such as the Soft Actor-Critic (SAC) algorithm.

4.1. Framework Overview

CPeSFA can be divided into three components: the policy network module, the CPeSFA module, and the policy representation module. The policy network module consists of a standard actor network, which takes the state as input and outputs the corresponding actions for interaction with the environment. Additionally, the actor network serves as input to the policy representation module and can be encoded into the respective policy representation. The CPeSFA module involves a CPeSFA network, which takes the state, action, and corresponding policy representation as input to generate the SFs for the given policy. The policy representation module includes an encoding network that takes the parameters of the actor and encodes them into the policy representation, which is then input into the CPeSFA module. The architecture diagram is shown in Figure 1.

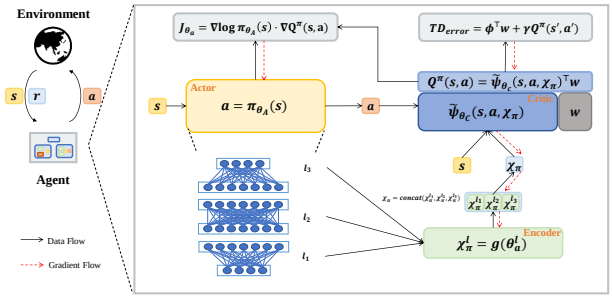


Figure 1: The overall Architecture of CPeSFA

4.2. Continuous PeSFA

Building upon PeSFA, we modified the original value-based architecture of PeSFA into the AC framework to enable learning policies in continuous action spaces. We refer to this adapted version as Continuous PeSFA (CPeSFA).

In order to apply PeSFA to AC architecture algorithms and endow it with adaptability across different AC algorithms, we modify the Critic part of the AC framework to adopt the CPeSFA structure. Specifically, we modify the action-value

function to be calculated by CPeSFA combined with the corresponding task weight. The action-value function of policy π under task w can be expressed as a composition of CPeSFA and the task weight w: $Q^{\pi}(s,a) = \psi(s,a,\pi)^{\top}\mathrm{w}$.

In the CPeSFA framework, the loss function for Actor is:

$$J_{\pi}(\theta_A) = \mathbb{E}[\log \pi(s,a)(\psi_{\theta}(s,a,\chi_{\pi})^{\top}\mathrm{w})] \tag{6}$$

And loss function for Critic is:

$$\begin{aligned} J_Q(\theta) = \mathbb{E}[(&\psi_{\theta}(s,a,\chi_{\pi})^{\top}\mathrm{w} \\ &- (r + \gamma\psi_{\theta^-}(s',a',\chi_{\pi})^{\top}\mathrm{w}))^2] \end{aligned} \tag{7}$$

where θ and θ_A represent the parameters of CPeSFA and the actor, respectively. θ^- denotes the target network parameters of CPeSFA, χ_{π} represents the representation of policy π, and w stands for the task weight.

Due to the explicit definition of the actor in CPeSFA using the AC architecture, we can employ the Origin Policy Representation (OPR) (Tang et al., 2022) algorithm to encode the network parameters of the actor as the policy representation. This approach, in contrast to PeSFA's (Li et al., 2022) use of the Surface Policy Representation (SPR)(Tang et al., 2022) algorithm to encode state-action pairs as policy representation data, is more stable and less susceptible to the influence of data collected from the environment. Additionally, when conventional SFs are extended to continuous action spaces, the inability to traverse the action space and select the action with the maximum value prevents the use of GPI algorithms, rendering SFs incapable of knowledge transfer. However, CPeSFA, through decoupling SFs from policies, enables the utilization of its policy-level generalization in continuous action spaces, leveraging historical knowledge to accelerate learning. This realization grants CPeSFA the ability to transfer knowledge in continuous action spaces. In the subsequent Section 4.3, we will further demonstrate how CPeSFA utilizes its policy-level generalization to expedite learning.

4.3. Theoretical Analysis of Generalization in Policy Representation Space for Continuous PeSFA

CPeSFA inherits the excellent properties of PeSFA, allowing the agent to learn SFs in continuous action space environments. By leveraging the generalization in policy space, CPeSFA accurately estimates the SFs of the optimized policy during the iterative optimization process when learning the optimal policy for a given task, thereby accelerating the learning rate. Next we will show the theorem and excellent properties of CPeSFA. The theoretical derivation and proof process and more details can be found in Appendix C.

Furthermore, due to the decoupling property of policies and tasks in CPeSFA, when the agent learns a policy for a new task, it can quickly compute the action-value function with the weight of the given task and the SFs of the policy, i.e., $Q^{\pi}_{\mathrm{w}'}(s,a) = \psi(s,a,\chi_{\pi})^{\top}\mathrm{w}'$. Moreover, CPeSFA can utilize policy-level generalization to fit SFs for policies from historical tasks when learning the policy for a new task. This provides a better starting point for estimating the policy value for the new task, accelerating policy learning, and achieving the effect of utilizing knowledge from historical task policies for transfer.

For illustrative purposes, we employ a consistent policy representation function, denoted as $\chi_{\pi} = \pi$. We define the approximation loss of CPeSFA for any policy π from the policy space Π as $f_{\theta}(\pi) = \|\psi_{\theta}(\pi) - \psi^{\pi}\|$. The approximation error of the action value function corresponding to task w can be defined as:

$$\begin{aligned} f^{\mathrm{w}}_{Q_{\theta}}(\pi) &= \|\psi_{\theta}(\pi)^{\top}\mathrm{w} - Q^{\pi}_{\mathrm{w}}\| \\ &= \|(\psi_{\theta}(\pi) - \psi^{\pi})^{\top}\mathrm{w}\| \end{aligned}$$

Next, we will explain how CPeSFA can expedite the learning process by leveraging its generalization capabilities in the policy space during policy evaluation and optimization. We define the optimization process for policy evaluation under task w as: $\mathcal{P}^{\mathrm{w}}_{\pi} = \Theta \to \hat{\Theta}$, and for $\hat{\theta} = \mathcal{P}^{\mathrm{w}}_{\pi}(\theta)$, the approximation error is further reduced, i.e., $f^{\mathrm{w}}_{Q_{\hat{\theta}}}(\pi) \leq \gamma f^{\mathrm{w}}_{Q_{\theta}}(\pi)$, where $\gamma \in [0,1)$. Therefore, the process of policy evaluation and policy optimization can be represented as $\theta_0 \xrightarrow{\mathcal{P}^{\mathrm{w}}_{\pi_1}} \theta_1 \xrightarrow{\mathcal{P}^{\mathrm{w}}_{\pi_2}} \theta_2 \xrightarrow{\mathcal{P}^{\mathrm{w}}_{\pi_3}} \dots$.

Theorem 4.1. *In the process of policy evaluation and optimization, for any $t \geq 0$, if $f^{\mathrm{w}}_{Q_{\theta_t}}(\pi_t) + f^{\mathrm{w}}_{Q_{\theta_t}}(\pi_{t+1}) \leq \|Q^{\pi_t} - Q^{\pi_{t+1}}\|$, then $f^{\mathrm{w}}_{Q_{\theta_t}}(\pi_{t+1}) \leq \|(\psi_{\theta_t}(\pi_t) - \psi^{\pi_{t+1}})^{\top}\mathrm{w}\|$*

According to the above theorem, it is evident that during the process of policy evaluation and policy optimization, CPeSFA places an upper bound on the estimation of the action values for the optimized policy. This upper bound is determined by the difference between the estimate of the action values for the previous policy and the true action values of the optimized policy. Therefore, in the process of policy evaluation and optimization, CPeSFA can provide a more accurate starting point by more accurately estimating the SFs of the optimized policy. This results in a more precise estimation of action values, thereby accelerating the rate of policy learning.

Corollary 4.2. *When switching task weights, the change in estimation error mainly depends on the difference in task weights and the estimation error of SFs for the policy π. The approximation error of the action-value function for the same policy under the new task can be expressed as:*

$$\begin{aligned} f^{\mathrm{w}_2}_{Q_{\theta_t}}(\pi_{t+1}) \leq \prod_{i=0}^{t}\gamma_t f_{Q^{\mathrm{w}_1}_{\theta_0}}(\pi_1) + \sum_{i=0}^{t-1}\prod_{j=i+1}^{t}\gamma_j\mathcal{M}_i \\ + \mathcal{M}_t + f_{\theta_t}(\pi_{t+1})\|\mathrm{w}_2 - \mathrm{w}_1\| \end{aligned}$$

From the above derivation, it can be inferred that the upper bound of the fitting error for the policy under the new

task is determined by the repetitive contraction of the initial approximation error from the historical task, coupled with the local generalization error along the policy optimization path and the disparity in weights between the new task and historical task. Therefore, when CPeSFA switches tasks, it can also utilize the generalization capabilities in the policy space, providing the agent with more accurate estimates of action values for policies under different tasks. Furthermore, leveraging CPeSFA's generalization capabilities across policy space and task space, it enables rapid estimation of the action-value function for the current policy under the new task by utilizing knowledge of historical policy. This facilitates the transfer and utilization of knowledge from historical tasks and policies.

Corollary 4.3. *If we switch tasks during the process of policy evaluation and optimization from* w_1 *to* w_2*, i.e.,* $\theta^{\mathrm{w}_1}_{t-1} \xrightarrow{\mathcal{P}^{\mathrm{w}_2}_{\pi_t}} \theta^{\mathrm{w}_2}_{t} \xrightarrow{\mathcal{P}^{\mathrm{w}_2}_{\pi_{t+1}}} \theta^{\mathrm{w}_2}_{t+1} \xrightarrow{\mathcal{P}^{\mathrm{w}_2}_{\pi_{t+2}}} \ldots$*, and* $f^{\mathrm{w}_1}_{Q_{\theta_t}}(\pi_t) + f^{\mathrm{w}_2}_{Q_{\theta_t}}(\pi_{t+1}) \leq \|Q^{\pi_t}_{\mathrm{w}_1} - Q^{\pi_{t+1}}_{\mathrm{w}_2}\|$*, there exists an upper bound when using CPeSFA to estimate the action values function of the optimized policy under the new task:*

$$f^{\mathrm{w}_2}_{Q_{\theta_t}}(\pi_{t+1}) \leq \|\psi_{\theta_t}(\pi_t)^\top(\mathrm{w}_2 - \mathrm{w}_1)\| + \|(\psi_{\theta_t}(\pi_t) - \psi(\pi_{t+1}))^\top \mathrm{w}_2\|$$

This upper bound is related to the change in tasks during the switch and the difference between CPeSFA's estimate of SFs for the current policy and the true SFs for the optimized policy. Compared to a standard critic, CPeSFA can more rapidly estimate action values in the new task by leveraging its property of decoupling policy from the task. Additionally, it can utilize its generalization properties in the policy space to more accurately estimate the actions of the optimized policy in the new task, thus accelerating the learning rate of the policy.

4.4. The training algorithm of CPeSFA

The CPeSFA algorithm can be combined with any AC architecture RL algorithm. For the ease of presentation of our algorithm's training process and corresponding experiments, we opt to use the SAC algorithm as the foundational framework to implement CPeSFA. The overall algorithm of CPeSFA is shown in Algorithm1.

The policy network module and the CPeSFA module together form the actor and critic components of the AC architecture in SAC. The policy network module comprises a conventional actor network, and its update formula is similar to SAC. It only needs to replace the action value function in the target distribution that the actor aims to match with the form of CPeSFA. i.e., $\pi(\cdot \mid s) \propto e^{Q(s,\cdot)} = e^{\psi(s,\cdot,\pi)^\top \mathrm{w}}$. For the policy representation χ_π in this paper, we use the encoding network to encode the parameters θ_a of the actor into the corresponding policy representation, i.e., $\chi_\pi = f_{\theta_p}(\theta_a)$, where θ_p represents the parameters of the policy representation encoding network. Then the actor's objective function is as follows:

$$J_\pi(\theta_a) = \mathbb{E}[\alpha \log \pi_{\theta_a}(a \mid s) - \psi_\theta(s, a, f_{\theta_p}(\theta_a))^\top \mathrm{w}] \quad (8)$$

Where θ_a represents the parameters of the policy network.

On the other hand, the critic part is a combination of the CPeSFA network and the task weight vector. The CPeSFA module first encodes the policy network into policy representations using the policy representation module. It then inputs these representations into the PeSFA network, combined with the given task weight vector, to form the critic part, i.e., $Q^\pi(s,a) = \tilde{\psi}(s,a,\pi)^\top \mathrm{w}$. The target for TD learning can now be represented as:

$$y = r + \gamma \mathbb{E}[\psi_{\theta^-}(s', a', f_{\theta_p^-}(\theta_a))^\top \mathrm{w} - \alpha \log \pi_{\theta_a}(a' \mid s')] \quad (9)$$

where θ_p^- represents the target policy representation encoding network, updated using $\theta_p^- \leftarrow \tau\theta_p + (1-\tau)\theta_p^-$. CPeSFA is updated as:

$$J_Q(\theta, \theta_p) = \mathbb{E}[(y - \psi_\theta(s, a, f_{\theta_p}(\theta_a))^\top \mathrm{w})^2] \quad (10)$$

During the update of the CPeSFA network, gradients are backpropagated to the policy representation module, enabling an end-to-end update of the policy representation encoding network.

The policy representation module primarily consists of encoding networks θ_p that encode the network parameters of each layer of the actor network into policy representations. The representations obtained from each layer of the actor are concatenated to form the complete policy representation. Therefore, the number of encoding networks in the policy representation module is consistent with the layers of the actor. Each layer of the actor has a corresponding encoding network to encode the respective network weights as the policy representation, i.e., $\chi_\pi^{l_i} = f^i_{\theta_p}(\theta_a^{l_i})$, where l_i denotes the i-th layer of the actor. Thus, when the actor has three layers, the policy representation can be expressed as $\chi_\pi = f_{\theta_p}(\theta_a) = concat(\chi_\pi^{l_1}, \chi_\pi^{l_2}, \chi_\pi^{l_3})$.

5. Experiments

In this section, we will present the primary experimental results of CPeSFA and compare them with baseline methods. To more effectively demonstrate the robustness and efficacy of the proposed CPeSFA algorithm, we will assess its performance across various environments by integrating it with both the discrete and continuous versions of the SAC (Haarnoja et al., 2018) algorithm.

We primarily utilize the Grid World (Barreto et al., 2017), Reacher (Barreto et al., 2017), and Point Maze (Fu et al., 2020) environments to assess the performance of CPeSFA. We compare CPeSFA with two baseline algorithms, namely

SFs (Barreto et al., 2017) and PeSFA (Li et al., 2022). For each algorithm across various environments and tasks, we conducted testing on multiple seeds and averaged the results to mitigate experimental variability. The experimental outcomes demonstrate the effectiveness of our algorithm. More implementation details can be found in the AppendixA.

First, we will conduct experiments in Grid World with a discrete action space under the mini-grid engine (Chevalier-Boisvert et al., 2023). In the Grid World environment, the agent aims to maximize the cumulative reward by navigating towards the goal while picking up objects with higher reward values whenever possible,

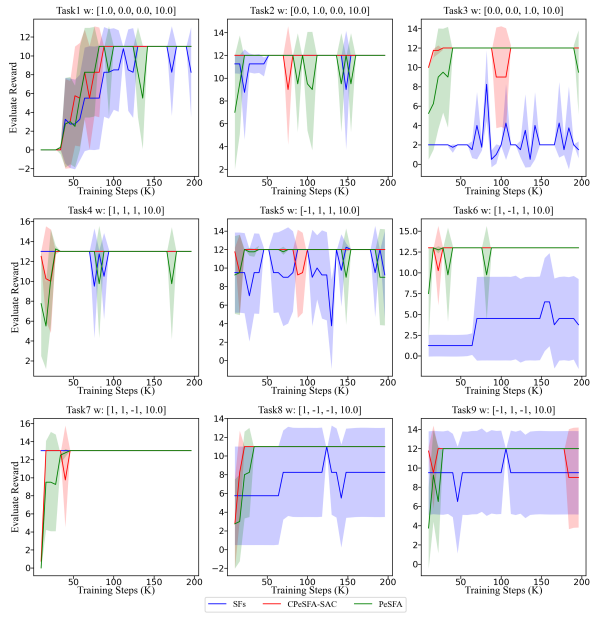


Figure 2: Evaluate reward of each task on Grid World

We design a series of tasks to verify whether CPeSFA can leverage its generalization in the policy representation space to expedite the learning rate of the agent on individual tasks. We also investigate whether CPeSFA can utilize SFs' decoupling of policies and tasks, along with policy-level generalization, to quickly find optimal policies for new tasks using learned policy spaces from historical tasks.

As shown in Figure 2, we present reward curves depicting the performance of CPeSFA and baseline algorithms across multiple tasks in the Grid World environment. The x-axis of each subplot represents the training steps, while the y-axis represents the average reward over multiple testing rounds. The title of each subplot indicates the corresponding task weight vector. It can be observed that, except for the first task where CPeSFA exhibits a slower learning rate, CPeSFA converges faster than the baseline algorithms on the remaining tasks, achieving higher reward. This suggests that CPeSFA effectively leverages the generalization in the policy space of SFs to expedite learning. Through the iterative process of policy evaluation and optimization, CPeSFA uses the generalization in the policy space to more accurately estimate the SFs corresponding to the optimized policy. This provides a better starting point for policy optimization, thus accelerating the learning rate of policies on individual tasks.

Moreover, from the result of tasks 2-9, it can be observed that CPeSFA consistently converges rapidly. This indicates that CPeSFA can utilize the learned policy from historical tasks to accelerate the overall learning rate. Due to the decoupling of SFs from policies in CPeSFA, the model can quickly identify the optimal policy for new tasks in the policy space. It can leverage the generalization in the policy space, find the optimal policy swiftly, and utilize the learned SFs corresponding to historical tasks in the policy space as a better starting point, thereby accelerating the learning rate of policies even after task switching. Additionally, Figure 5a presents the cumulative average reward curves. It shows that CPeSFA consistently outperforms the baseline algorithms throughout the training process.

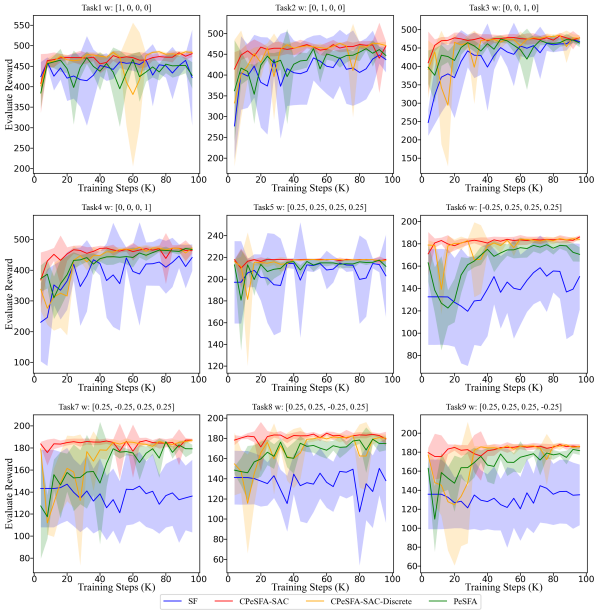


Figure 3: Evaluate reward of each task On Reacher

Subsequently, we evaluated the performance of CPeSFA in the Reacher environment. The Reacher environment is a continuous control task under the Mujoco engine (Todorov et al., 2012), where the goal is to manipulate a robotic arm to a specified location. In this environment, we conduct experiments using both the SAC algorithm based on continuous action space and discrete action space in conjunction with CPeSFA. We refer to them as CPeSFA-SAC and CPeSFA-SAC-Discrete, respectively. The baseline algorithms for comparison are tested on the discrete action space of the Reacher environment.

As seen in Figure 3, the CPeSFA algorithm exhibits notably superior performance in continuous action spaces. In the initial learning stage of the first task, CPeSFA-SAC demonstrates excellent results, converging more rapidly compared to other algorithms. Moreover, after switching to new tasks, CPeSFA-SAC leverages its generalization capabilities and knowledge acquired from historical tasks to swiftly learn the optimal policy for new tasks.

Examining the performance of CPeSFA-Discrete, it is observed that CPeSFA-Discrete performs slightly worse than CPeSFA-SAC. However, it still capitalizes on the advantages of CPeSFA in quickly learning the optimal policy after task switching, yielding better experimental results than most of the baseline algorithms in the majority of tasks. Figure 5b displays the cumulative average reward curves in the Reacher environment. CPeSFA-SAC outperforms other algorithms significantly, and CPeSFA-SAC performs better than CPeSFA-Discrete, indicating that CPeSFA excels in continuous action space environment with more nuanced control methods than the coarse-grained control of discrete action space.

Furthermore, our extension of the PeSFA to continuous action spaces in CPeSFA, allowing SFs to leverage its decoupling feature of policies and tasks, facilitates the transfer of historical knowledge. This extension enables algorithms based on the SFs framework to achieve excellent results in continuous action spaces through more refined control. The cumulative average reward curves also demonstrate that the performance of CPeSFA-SAC is significantly better than PeSFA, underscoring the necessity of extending SFs algorithms to continuous action spaces.

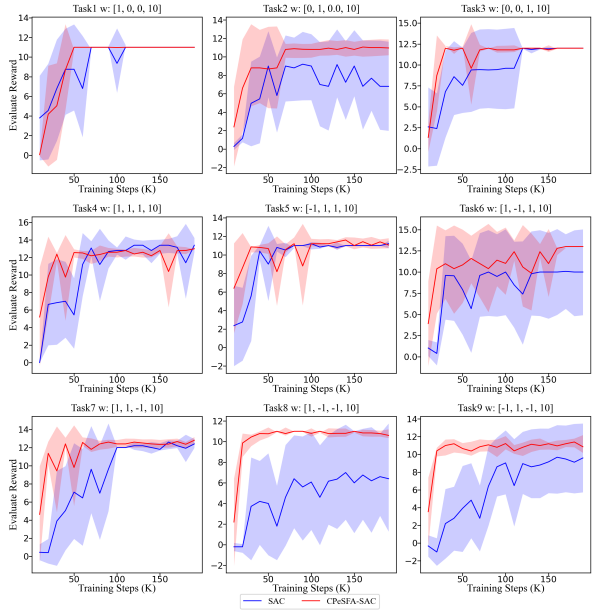
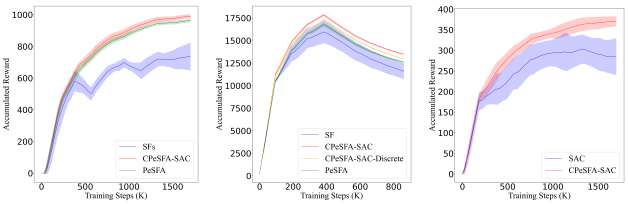


Figure 4: Evaluate reward of each task on Point Maze

Finally, we validate the effectiveness of the CPeSFA algorithm in the Point Maze environment under D4RL (Fu et al., 2020). The Point Maze environment is designed with a scenario similar to the Grid World environment, but both the state and action spaces are continuous.

We aim to evaluate the performance of CPeSFA in navigation tasks in continuous action environment. Additionally, we discretize the action space to test the effects of PeSFA and SFs in this environment. However, through experimentation, we found that PeSFA and SFs algorithms completely fail to converge or learn effective policies in this discretized continuous action environment. This underscores the necessity of extending SFs algorithms to continuous action spaces. Therefore, we will compare the performance of CPeSFA with the SAC algorithm in this environment.



(a) Grid World (b) Reacher (c) Point Maze

Figure 5: Accumulated reward of the total training phase

Figures 4 and 5c respectively illustrate the reward curves for each task and the cumulative average reward curves for algorithms in the Point Maze environment. It is evident that the CPeSFA algorithm significantly outperforms SAC. CPeSFA exhibits a faster learning rate and overall robustness across tasks compared to SAC. Furthermore, even after task switching, CPeSFA can swiftly learn the optimal policy for new tasks, demonstrating the transfer of historical knowledge to accelerate the learning rate in new tasks.

6. Conclusion

This paper introduces the Continuous PeSFA (CPeSFA) algorithm, extending the value-based PeSFA algorithm to an easily integrable form within the Actor-Critic (AC) architecture. CPeSFA is designed to learn policies in continuous action spaces while inheriting the advantageous features of the PeSFA algorithm. We demonstrate that CPeSFA leverages the generalization property of SFs in the policy space to expedite the learning rate. Furthermore, CPeSFA achieves the transfer capability that is challenging for traditional SFs in continuous action spaces. Finally, the experiments results demonstrate that CPeSFA outperforms other baseline algorithms significantly and effectively transfers historical knowledge for rapid learning. In the future, we will further enhance the transferability of CPeSFA. Leveraging CPeSFA as a foundational framework, we aim to develop new transfer algorithms for SFs in continuous action spaces, capitalizing on the generalization capability in policy space.

Acknowledgements

The authors express their gratitude to the reviewers for their highly valuable feedback.

References

Alegre, L. N., Bazzan, A., and Da Silva, B. C. Optimistic linear support and successor features as a basis for optimal policy transfer. In *International Conference on Machine Learning*, pp. 394–413. PMLR, 2022.

Alver, S. and Precup, D. Constructing a good behavior basis for transfer using generalized policy updates. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., Silver, D., and van Hasselt, H. Successor features for transfer in reinforcement learning. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4055–4065, 2017.

Borsa, D., Barreto, A., Quan, J., Mankowitz, D. J., van Hasselt, H., Munos, R., Silver, D., and Schaul, T. Universal successor features approximators. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Chevalier-Boisvert, M., Dai, B., Towers, M., de Lazcano, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., and Terry, J. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

Feinberg, A. Markov decision processes: Discrete stochastic dynamic programming (martin l. puterman). *SIAM Rev.*, 38(4):689, 1996.

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

Garces, K., Xuan, J., and Zuo, H. Transformed successor features for transfer reinforcement learning. In *Australasian Joint Conference on Artificial Intelligence*, pp. 298–309. Springer, 2023.

Grondman, I., Busoniu, L., Lopes, G. A. D., and Babuska, R. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Trans. Syst. Man Cybern. Part C*, 42(6):1291–1307, 2012.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018.

Han, D. and Tschiatschek, S. Option transfer and SMDP abstraction with successor features. In Raedt, L. D. (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pp. 3036–3042, 2022.

Hansen, S., Dabney, W., Barreto, A., Warde-Farley, D., de Wiele, T. V., and Mnih, V. Fast task inference with variational intrinsic successor features. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

Hao, J., Yang, T., Tang, H., Bai, C., Liu, J., Meng, Z., Liu, P., and Wang, Z. Exploration in deep reinforcement learning: From single-agent to multiagent domain. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Kim, J., Park, S., and Kim, G. Constrained GPI for zero-shot transfer in reinforcement learning. In *NeurIPS*, 2022.

Lehnert, L., Tellex, S., and Littman, M. L. Advantages and limitations of using successor features for transfer in reinforcement learning. *CoRR*, abs/1708.00102, 2017.

Li, Y., Yang, T., Hao, J., Zheng, Y., and Tang, H. Efficient deep reinforcement learning via policy-extended successor feature approximator. In *International Conference on Distributed Artificial Intelligence*, pp. 29–44. Springer, 2022.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Liu, H. and Abbeel, P. APS: active pretraining with successor features. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6736–6747, 2021.

Liu, Y. T. and Ahmad, A. Multi-task reinforcement learning in continuous control with successor feature-based concurrent composition. *CoRR*, abs/2303.13935, 2023.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

Nemecek, M. W. and Parr, R. Policy caches with successor features. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8025–8033, 2021.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1312–1320, 2015.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489, 2016.

Sutton, R. S. and Barto, A. G. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. 1998.

Tang, H., Meng, Z., Hao, J., Chen, C., Graves, D., Li, D., Yu, C., Mao, H., Liu, W., Yang, Y., et al. What about inputting policy in value function: Policy representation and policy-extended value function approximator. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8441–8449, 2022.

Tang, K., Kan, N., Jiang, Y., Li, C., Dai, W., Zou, J., and Xiong, H. Successor feature-based transfer reinforcement learning for video rate adaptation with heterogeneous qoe preferences. *IEEE Transactions on Multimedia*, 2023.

Taylor, M. E. and Stone, P. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10:1633–1685, 2009. URL https://dl.acm.org/doi/10.5555/1577069.1755839.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Touati, A., Rapin, J., and Ollivier, Y. Does zero-shot reinforcement learning exist? In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

Yang, T., Hao, J., Meng, Z., Zhang, Z., Hu, Y., Chen, Y., Fan, C., Wang, W., Liu, W., Wang, Z., and Peng, J. Efficient deep reinforcement learning via adaptive policy transfer. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 3094–3100, 2020.

Zhang, M., Jianye, H., Tang, H., and Zheng, Y. Boosting off-policy rl with policy representation and policy-extended value function approximator. In *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023.

Zhu, Z., Lin, K., and Zhou, J. Transfer learning in deep reinforcement learning: A survey. *CoRR*, abs/2009.07888, 2020.

A. Details of environments

We employed three environments to validate the effectiveness of CPeSFA, namely Grid World, Reacher, and Point Maze. The specific configurations of these three environments will be detailed in the following sections.

Table 1: Task weight per environment.

task	Grid World/Point Maze	Reacher
1	$[1, 0, 0, 10]$	$[1, 0, 0, 0]$
2	$[0, 1, 0, 10]$	$[0, 1, 0, 0]$
3	$[1, 0, 1, 10]$	$[0, 0, 1, 0]$
4	$[1, 1, 1, 10]$	$[0, 0, 0, 1]$
5	$[-1, 1, 1, 10]$	$[0.25, 0.25, 0.25, 0.25]$
6	$[1, -1, 1, 10]$	$[-0.25, 0.25, 0.25, 0.25]$
7	$[1, 1, -1, 10]$	$[0.25, -0.25, 0.25, 0.25]$
8	$[1, -1, -1, 10]$	$[0.25, 0.25, -0.25, 0.25]$
9	$[-1, 1, -1, 10]$	$[0.25, 0.25, 0.25, -0.25]$

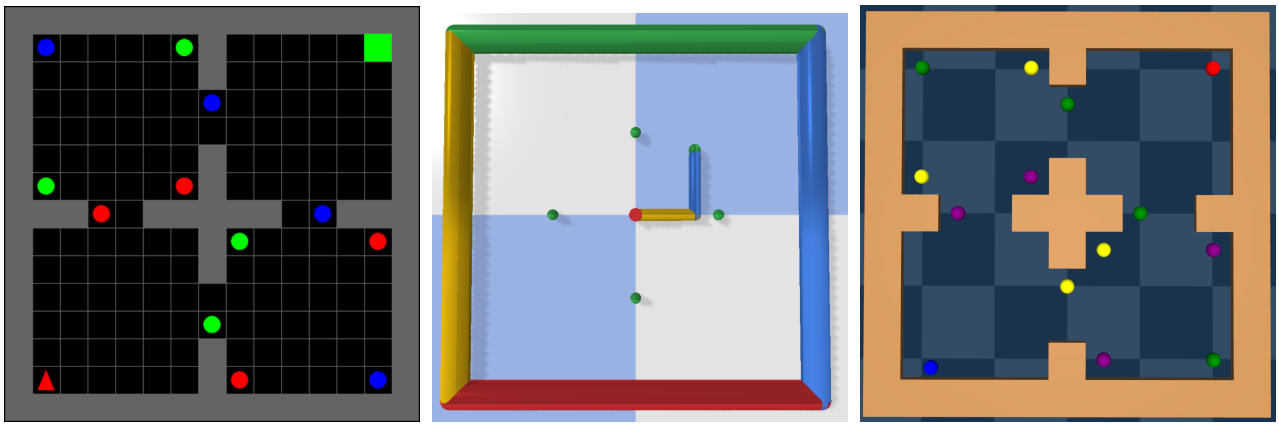
In the Grid World environment, the agent starts from an initial position in a 15x15 grid and aims to maximize the cumulative reward by navigating towards the goal while picking up objects with higher reward values whenever possible, as illustrated in Figure 6a. There are a total of 12 objects in the environment, with four different types, each having four objects. The state space dimension of this environment is 43, including one-hot vectors for the x and y coordinates and one-hot vectors indicating whether the agent has picked up a specific object. The action space is four-dimensional, denoted as $A = \{UP, RIGHT, DOWN, LEFT\}$. The reward function is defined as $r(s, a, s') = \phi(s, a, s')^\top \mathrm{w}$, where the dynamic features $\phi(s, a, s') \in \{0, 1\}^4$ indicate whether the agent picks up a particular object in a one-step transition. The dynamic feature ϕ in SFs is a four-dimensional one-hot vector, representing whether the agent picks up a certain type of object in a one-step transition. If the agent picks up an object of a particular type, the corresponding position in ϕ is 1; otherwise, it is 0. The task-specific weight w vector is a four-dimensional vector representing the reward values for each object type. Therefore, $r = \phi(s, a, s')^\top \mathrm{w}$ signifies the reward obtained by the agent in a single-step transition.

Reacher is an environment with continuous action space, where the agent controls two joint-connected robotic arms to bring the end endpoint as close as possible to target points with higher reward values. The closer the endpoint of the robotic arm is to the target point, the greater the reward, as illustrated in Figure 6b. There are four target points in this environment, and the coordinates of the targets, with the starting point of the robotic arms as the origin, are $(0, 0.14)$, $(0.14, 0)$, $(-0.14, 0)$, and $(0, -0.14)$. The state space dimension of this environment is 5, representing dynamic features of the robotic arms. The action dimension is 2, representing the magnitude and direction of the forces applied by the two robotic arms. In this environment, the dynamic feature ϕ in SFs is a four-dimensional vector, with each dimension given by $\phi^i = 1 - 4 \cdot d$, where d is the Euclidean distance of the endpoint to target point i. The task weight is a four-dimensional vector representing the reward magnitudes for the four target points.

In the continuous action space of this environment, the action dimension is 2, controlling the magnitude of the force in two directions for the agent's two robotic arms. In the case of a discrete action space, we discretize the action space into 9 discrete actions, denoted as $\{-1, 0, 1\}^2$. In this environment, we conduct experiments using both the SAC algorithm based on continuous action space and discrete action space in conjunction with CPeSFA. We refer to them as CPeSFA-SAC and CPeSFA-Discrete, respectively. The baseline algorithms for comparison are tested on the discrete action space of the Reacher environment.

Point Maze is an environment with continuous action space, similar to the Grid World environment. The settings for SFs, including ϕ and w, are consistent with those in the Grid World environment, as illustrated in Figure 6c.. However, the state space and action space differ. The state space dimension in this environment is 17, representing information related to the velocity and position of the ball, as well as a one-hot vector indicating whether the agent has picked up a certain object. The action space dimension is 2, representing the magnitude and direction of the force applied to the ball along the x and y axes.

Task weights for each environment are set as shown in Table 1.



(a) Grid World (b) Reacher (c) Point Maze

Figure 6: Environmental schematic diagram

B. Details of Training

The parameter settings for CPeSFA, PeSFA, and SFs algorithms used in this paper are shown in the following table.

Table 2: CPeSFA's hyperparameters per environment.

	Grid World	**Reacher**	**Point Maze**
CPeSFA network$\tilde{\psi}$	MLP([256, 256])	MLP([256, 256])	MLP([256, 256])
Actor networkπ_{θ_a}	MLP([64, 64])	MLP([64, 64])	MLP([64, 64])
Minibatch size	128	128	128
learning rate	0.001	0.001	0.001
gamma	0.9	0.95	0.99
Optimiser	ADAM	ADAM	ADAM
policy representation dim	48	16	32
OPR encoder network f_{θ_p}	MLP([32, 32])	MLP([32, 32])	MLP([32, 32])

Table 3: PeSFA's hyperparameters per environment.

	Grid World	**Reacher**	**Point Maze**
PeSFA network$\tilde{\psi}$	MLP([256, 256])	MLP([256, 256])	MLP([256, 256])
Minibatch size	128	128	128
learning rate	0.001	0.001	0.001
gamma	0.9	0.95	0.99
Optimiser	ADAM	ADAM	ADAM
policy representation dim	32	32	24
SPR encoder network	MLP([64, 64])	MLP([64, 64])	MLP([64, 64])
SPR decoder network	MLP([128, 128])	MLP([128, 128])	MLP([128, 128])

Table 4: SF's hyperparameters per environment.

	Grid World	**Reacher**	**Point Maze**
Critic network	MLP([256, 256])	MLP([256, 256])	MLP([256, 256])
Minibatch size	128	128	128
learning rate	0.001	0.001	0.001
gamma	0.9	0.95	0.99
Optimiser	ADAM	ADAM	ADAM

Algorithm 1 CPeSFA-SAC

Input: PeSFA net $\tilde{\psi}_\theta$, Actor $\tilde{\pi}_{\theta_a}$, policy representation encoder net f_{θ_p}, replay buffer D_{memory}
for $task\_id \leftarrow 1, 2, ..., task\_num$ **do**
  for $epoch \leftarrow 1, 2, ..., max\_epoch$ **do**
    select initial state $s \in \mathcal{S}$
    Execute action a and observe s', r, ϕ
    push (s, a, ϕ, s', r) into D_{memory}
    if step % training_gap == 0 **then**
      sample (s, a, ϕ, r, s') from D_{memory}
      update $\tilde{\psi}_\theta$ and θ_p using equation 10
      update actor θ_a using equation 9
      update target networks $\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-$,$\theta_p^- \leftarrow \tau\theta_p + (1-\tau)\theta_p^-$
    end if
  end for
end for

C. Details of theoretical derivation and proof

Theorem C.1. *In the process of policy evaluation and optimization, for any $t \geq 0$, if $f^{\mathrm{w}}_{Q_{\theta_t}}(\pi_t) + f^{\mathrm{w}}_{Q_{\theta_t}}(\pi_{t+1}) \leq \|Q^{\pi_t} - Q^{\pi_{t+1}}\|$, then $f^{\mathrm{w}}_{Q_{\theta_t}}(\pi_{t+1}) \leq \|(\psi_{\theta_t}(\pi_t) - \psi^{\pi_{t+1}})^\top \mathrm{w}\|$*

Proof. According to the triangle inequality, if $f^{\mathrm{w}}_{Q_{\theta_t}}(\pi_t) + f^{\mathrm{w}}_{Q_{\theta_t}}(\pi_{t+1}) \leq \|Q^{\pi_t} - Q^{\pi_{t+1}}\|$, we can obtain:

$$\begin{aligned} &f^{\mathrm{w}}_{Q_{\theta_t}}(\pi_t) + f^{\mathrm{w}}_{Q_{\theta_t}}(\pi_{t+1}) \leq \|Q^{\pi_t} - Q^{\pi_{t+1}}\| \\ &\leq \|Q_{\theta_t}(\pi_t) - Q^{\pi_t}\| + \|Q_{\theta_t}(\pi_t) - Q^{\pi_{t+1}}\| \\ &= f^{\mathrm{w}}_{Q_{\theta_t}}(\pi_t) + \|Q_{\theta_t}(\pi_t) - Q^{\pi_{t+1}}\| \end{aligned}$$

After simplification, we obtain the following equation:

$$f^{\mathrm{w}}_{Q_{\theta_t}}(\pi_{t+1}) \leq \|Q_{\theta_t}(\pi_t) - Q^{\pi_{t+1}}\| = \|(\psi_{\theta_t}(\pi_t) - \psi^{\pi_{t+1}})^\top \mathrm{w}\|$$

□

Corollary C.2. *When switching task weights, the change in estimation error mainly depends on the difference in task weights and the estimation error of SFs for the policy π. The approximation error of the action-value function for the same policy under the new task can be expressed as:*

$$f^{w_2}_{Q_{\theta_t}}(\pi_{t+1}) \leq \prod_{i=0}^{t} \gamma_t f_{Q^{w_1}_{\theta_0}}(\pi_1) + \sum_{i=0}^{t-1} \prod_{j=i+1}^{t} \gamma_j \mathcal{M}_i + \mathcal{M}_t + f_{\theta_t}(\pi_{t+1}) \|w_2 - w_1\|$$

Proof. For the same network parameters and policy, when the task weight is switched from w_1 to w_2, the difference in error for the same policy can be expressed in the following form:

$$\begin{aligned} f^{\mathrm{w}_2}_{Q_\theta}(\pi) - f^{\mathrm{w}_1}_{Q_\theta}(\pi) &\leq \|\psi_\theta(\pi)^\top(\mathrm{w}_2 - \mathrm{w}_1) - (Q^\pi_{\mathrm{w}_2} - Q^\pi_{\mathrm{w}_1})\| \\ &= \|\psi_\theta(\pi)^\top(\mathrm{w}_2 - \mathrm{w}_1) - (\psi^\pi)^\top(\mathrm{w}_2 - \mathrm{w}_1)\| \\ &= \|(\psi_\theta(\pi) - \psi^\pi)^\top(\mathrm{w}_2 - \mathrm{w}_1)\| \\ &\leq \|\psi_\theta(\pi) - \psi^\pi\|\|\mathrm{w}_2 - \mathrm{w}_1\| = f_\theta(\pi)\|\mathrm{w}_2 - \mathrm{w}_1\| \end{aligned}$$

From the definition in PeVFA (Tang et al., 2022), in the policy evaluation and optimization process $\theta_{t-1} \xrightarrow{\mathcal{P}^{\mathrm{w}}_{\pi_t}} \theta_t$, if $f^{\mathrm{w}}_{Q_\theta}$ is L-continuous at policy π_t, then we have $f^{\mathrm{w}}_{Q_{\theta_t}}(\pi_{t+1}) \leq \gamma_t f^{\mathrm{w}}_{Q_{\theta_{t-1}}}(\pi_t) + \mathcal{M}_t(\pi_t, \pi_{t+1}, L)$, where $\mathcal{M}_t(\pi_t, \pi_{t+1}, L) = L_t \cdot d(\pi_t, \pi_{t+1})$, and the following corollary can be obtained.

$$f^{\mathrm{w}_1}_{Q_{\theta_t}}(\pi_{t+1}) \leq \prod_{i=0}^{t} \gamma_t f^{\mathrm{w}_1}_{Q_{\theta_0}}(\pi_1) + \sum_{i=0}^{t-1} \prod_{j=i+1}^{t} \gamma_j \mathcal{M}_i + \mathcal{M}_t$$

When switching tasks, the approximation error of the action-value function for the same policy under the new task can be expressed as:

$$\begin{aligned} f^{\mathrm{w}_2}_{Q_{\theta_t}}(\pi_{t+1}) &\leq f^{\mathrm{w}_1}_{Q_{\theta_t}}(\pi_{t+1}) + f_{\theta_t}(\pi_{t+1})\|\mathrm{w}_2 - \mathrm{w}_1\| \\ &\leq \prod_{i=0}^{t} \gamma_t f^{\mathrm{w}_1}_{Q_{\theta_0}}(\pi_1) + \sum_{i=0}^{t-1} \prod_{j=i+1}^{t} \gamma_j \mathcal{M}_i + \mathcal{M}_t + f_{\theta_t}(\pi_{t+1})\|\mathrm{w}_2 - \mathrm{w}_1\| \end{aligned}$$

□

Corollary C.3. *If we switch tasks during the process of policy evaluation and optimization from* w_1 *to* w_2*, i.e.,* $\theta_{t-1}^{\mathrm{w}_1} \xrightarrow{\mathcal{P}_{\pi_t}^{\mathrm{w}_2}} \theta_t^{\mathrm{w}_2} \xrightarrow{\mathcal{P}_{\pi_{t+1}}^{\mathrm{w}_2}} \theta_{t+1}^{\mathrm{w}_2} \xrightarrow{\mathcal{P}_{\pi_{t+2}}^{\mathrm{w}_2}} \ldots$*, and* $f_{Q_{\theta_t}}^{\mathrm{w}_1}(\pi_t) + f_{Q_{\theta_t}}^{\mathrm{w}_2}(\pi_{t+1}) \leq \|Q_{\mathrm{w}_1}^{\pi_t} - Q_{\mathrm{w}_2}^{\pi_{t+1}}\|$*, there exists an upper bound when using CPeSFA to estimate the action values function of the optimized policy under the new task:*

$$f_{Q_{\theta_t}}^{\mathrm{w}_2}(\pi_{t+1}) \leq \|\psi_{\theta_t}(\pi_t)^\top(\mathrm{w}_2 - \mathrm{w}_1)\| + \|(\psi_{\theta_t}(\pi_t) - \psi(\pi_{t+1}))^\top \mathrm{w}_2\|$$

Proof. According to the condition $f_{Q_{\theta_t}}^{\mathrm{w}_1}(\pi_t) + f_{Q_{\theta_t}}^{\mathrm{w}_2}(\pi_{t+1}) \leq \|Q_{\mathrm{w}_1}^{\pi_t} - Q_{\mathrm{w}_2}^{\pi_{t+1}}\|$, we have

$$\begin{aligned}
&f_{Q_{\theta_t}}^{\mathrm{w}_1}(\pi_t) + f_{Q_{\theta_t}}^{\mathrm{w}_2}(\pi_{t+1}) \leq \|Q_{\mathrm{w}_1}^{\pi_t} - Q_{\mathrm{w}_2}^{\pi_{t+1}}\| \\
&\leq \|Q_{\theta_t}^{\mathrm{w}_1}(\pi_t) - Q_{\mathrm{w}_1}^{\pi_t}\| + \|Q_{\theta_t}^{\mathrm{w}_1}(\pi_t) - Q_{\mathrm{w}_2}^{\pi_{t+1}}\| \\
&= f_{Q_{\theta_t}}^{\mathrm{w}_1}(\pi_t) + \|Q_{\theta_t}^{\mathrm{w}_1}(\pi_t) - Q_{\mathrm{w}_2}^{\pi_{t+1}}\| \\
&= f_{Q_{\theta_t}}^{\mathrm{w}_1}(\pi_t) + \|\psi_{\theta_t}(\pi_t)^\top \mathrm{w}_1 - \psi(\pi_{t+1})^\top \mathrm{w}_2\| \\
&= f_{Q_{\theta_t}}^{\mathrm{w}_1}(\pi_t) + \|\psi_{\theta_t}(\pi_t)^\top \mathrm{w}_1 - \psi_{\theta_t}(\pi_t)^\top \mathrm{w}_2 + \psi_{\theta_t}(\pi_t)^\top \mathrm{w}_2 - \psi(\pi_{t+1})^\top \mathrm{w}_2\| \\
&= f_{Q_{\theta_t}}^{\mathrm{w}_1}(\pi_t) + \|\psi_{\theta_t}(\pi_t)^\top(\mathrm{w}_2 - \mathrm{w}_1) - (\psi_{\theta_t}(\pi_t) - \psi(\pi_{t+1}))^\top \mathrm{w}_2\|
\end{aligned}$$

After simplification, we obtain the following equation:

$$f_{Q_{\theta_t}}^{\mathrm{w}_2}(\pi_{t+1}) \leq \|\psi_{\theta_t}(\pi_t)^\top(\mathrm{w}_2 - \mathrm{w}_1)\| + \|(\psi_{\theta_t}(\pi_t) - \psi(\pi_{t+1}))^\top \mathrm{w}_2\|$$

□