A New Representation of Successor Features for Transfer across Dissimilar Environments

Majid Abdolshah [1] **Hung Le** [1] **Thommen Karimpanal George** [1] **Sunil Gupta** [1] **Santu Rana** [1] **Svetha Venkatesh** [1]

Abstract

Transfer in reinforcement learning is usually achieved through generalisation across tasks. Whilst many studies have investigated transferring knowledge when the reward function changes, they have assumed that the dynamics of the environments remain consistent. Many real-world RL problems require transfer among environments with different dynamics. To address this problem, we propose an approach based on successor features in which we model successor feature functions with Gaussian Processes permitting the source successor features to be treated as noisy measurements of the target successor feature function. Our theoretical analysis proves the convergence of this approach as well as the bounded error on modelling successor feature functions with Gaussian Processes in environments with both different dynamics and rewards. We demonstrate our method on benchmark datasets and show that it outperforms current baselines.

1. Introduction

Reinforcement learning (RL) is a computational approach that learns how to attain a complex goal by maximising rewards over time. Successful applications range from Atari games (Mnih et al., 2015), to robotics (Zhang et al., 2017), and self-driving cars (Liang et al., 2018). However, this success is based on solving each task from scratch, and thus training these agents requires vast amounts of data.

Several solutions have been proposed to address this problem. Most works in transfer RL such as Progressive Neural Networks (Rusu et al., 2016) and Inter-Task Mapping setups (Ammar & Taylor, 2011; Gupta et al., 2017; Konidaris & Barto, 2006; Yin & Pan, 2017) assume that the state-action space, or reward distribution space can be disentangled into independent sub-domains. However, learning interpretable, disentangled representations is challenging (Zhu et al., 2020). The dynamics and the reward was decoupled for the first time in (Barreto et al., 2017). Building upon an elegant formulation called the *successor function* (Dayan, 1993), the method allowed flexible transfer learning across tasks that differ in their reward structure. The underlying assumption was that the environmental dynamics remains unchanged and using a Generalised Policy Improvement (GPI) method, the optimal policy is determined. Such successor feature based methods have been shown to efficiently transfer knowledge across RL tasks (Barreto et al., 2018; 2019; 2020). Other works that have built upon successor features include generalised policy updates on successor features (Barreto et al., 2020), a universal type of successor feature based on the temporal difference method (Ma et al., 2020), and Variational Universal Successor Features (Siriwardhana et al., 2019) that perform target driven navigation. Option Keyboard (Barreto et al., 2019) leveraged successor features to combine skills to define and manipulate options. This allows for change in reward, but a slight change of the environment can deteriorate the performance of a new task. If however both the environmental dynamics and the reward differs across tasks, these methods are unable to handle this challenge.

In real-world problems when environment dynamics and rewards change across tasks, both these aspects need careful modelling. Failing to do so can lead to negative transfer of knowledge from the previously seen tasks. Thus RL methods using successor features need to be extended to handle the changes in environmental dynamics. Such work is limited. Zhang et al. (Zhang et al., 2017) aim to address this problem by considering a linear relationship between the source and target successor features. This modelling is restrictive and may fall short in capturing the complexity of the changed environment dynamics. *Thus, the problem of designing a method using a successor feature based approach to transfer knowledge from source to target environment where the dynamics are dissimilar, is still open.*

Our new approach enables the efficient learning of novel successor features to cater to the new target environmental dynamics. This is done by using the distribution of the previous (source) task successor features as a prior for the new tar-

[1]Applied Artificial Intelligence Institute (A^2I^2), Deakin University, Geelong, Australia. Correspondence to: Majid Abdolshah <majid@deakin.edu.au>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

get task. We model both the target and source distributions through Gaussian Processes (GPs). The target distribution is modeled as a noisy version of the source distribution. This approach assumes that the source and target environments lie within some proximity to each other i.e. they are similar within some noisy envelope. However, this adjustable noisy envelope impacts the upper bounded error on the modelling of optimal policy in the target environment. The advantage of this approach is that the source observations provide a head-start for the learning process and additional explorations in the target will provide efficient convergence to the optimal policy. We use a GPI method to estimate the target action value function. We provide theoretical analysis and upper bounds (1) on the difference of action-value functions when the optimal policy derived from environment i is replaced by the optimal policy derived from environment j; (2) on the estimation error of the action-value function of an optimal policy learned in a source environment when executed in the target environment; (3) on the difference of the optimal action-value function in the target environment and our GPI-derived action value function. We evaluate this approach in a variety of benchmark environments with different levels of complexity. Our key contributions are:

- A new method based on successor features and Gaussian Processes that enhances transfer from source to target tasks when the dynamics of the environment are dissimilar;
- A theoretical analysis for the new successor based method; and,
- An empirical comparison on diverse suit of RL benchmarks with different levels of complexity.

2. Background

2.1. Reinforcement Learning

We model the RL framework as a Markov Decision Process (MDP) described as $< \mathcal{S}, \mathcal{A}, p, R >$, where $\mathcal{S}$ is a finite state space, $\mathcal{A}$ represents a finite action space, $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ the transition probabilities, and $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a bounded reward function. A discount factor $\gamma \in (0, 1]$ encodes the importance of future rewards with respect to the current state. The objective of RL is to find an optimal policy $\pi(a|s) : \mathcal{S} \rightarrow \mathcal{A}$ that maps the states to the actions such that it maximises the expected discounted reward. The action-value of policy π is defined as $Q^\pi(s_t, a_t) = \mathbb{E}^\pi[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k})|s_0 = s_t, a_0 = a_t]$, where s_t and a_t are the state of the agent at time step t, and the action that is taken in that state, respectively.

Q-Learning: Q-Learning is an off-policy RL approach that aims to learn the optimal action-value function. This function can be updated recursively as:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}}\big[R(s_t, a_t) + \gamma \max_{a \in \mathcal{A}} \big(Q^\pi(s_{t+1}, a)\big)\big]. \tag{1}$$

After learning the action-value function, the optimal policy can be retrieved by selecting the best action at every state: $\pi^*(s) \in \operatorname{argmax}_{a \in \mathcal{A}} Q^*(s, a)$. In the next section we draw the connection between Q-Learning and successor features.

2.2. Successor Features

The successor feature representation allows decoupling of the dynamics of an MDP from its reward distributions. Baretto et al. (Barreto et al., 2017) generalised the successor representations that was first formulated in Dayan et al. (Dayan, 1993) decomposing the action-value function into a set of features that encode the dynamics of the environment and a weight that acts as a task-specific reward mapper. This decomposition can be formulated as:

$$R(s, a) = \boldsymbol{\phi}(s, a)^{\mathrm{T}} \mathbf{w}, \tag{2}$$

where $\boldsymbol{\phi}(s, a) \in \mathbb{R}^D$ are the features of (s, a) that represent the dynamics of the environment and $\mathbf{w}$ is the reward mapper of the environment. The two components can be learnt through supervised learning (Zhu et al., 2020). Note that $\boldsymbol{\phi}(.,.)$ can be any complex model such as a neural network. Baretto et al. (Barreto et al., 2017) showed that this decomposition can be used in the construction of the action-value function. Let us assume a reward function as in Eq. (2), the action value function can be derived as:

$$\begin{aligned} Q^\pi(s, a) = \mathbb{E}^\pi\big[R(s_{t+1}, a_{t+1}) + \gamma R(s_{t+2}, a_{t+2}) + \\ \ldots | s_t = s, a_t = a\big] = \mathbb{E}^\pi\big[\sum_{t=0}^{\infty} \gamma^t \phi(s_{t+1}, a_{t+1}) \\ | s_t = s, a_t = a\big]\mathbf{w} = \boldsymbol{\psi}(s, a)^{\mathrm{T}} \mathbf{w}, \end{aligned} \tag{3}$$

where $\boldsymbol{\psi}(s, a)^{\mathrm{T}}$ is the *"Successor Feature (SF)"* of (s, a) and summarises the dynamics induced by π. Eq. (3) satisfies the Bellman Equation and and can be learnt through any conventional method. By treating the latent representation $\boldsymbol{\phi}(.,.)$ as the immediate reward in the context of Q-Learning, the successor feature function can be written as:

$$\begin{aligned} \boldsymbol{\psi}^\pi(s, a) = \boldsymbol{\phi}(s_t, a_t) + \gamma \mathbb{E}^\pi\big[\boldsymbol{\psi}^\pi(s_{t+1}, \pi\big(s_{t+1}\big))|s_t = s, \\ a_t = a\big]. \end{aligned} \tag{4}$$

The principal advantage of SFs is that when the knowledge of $\boldsymbol{\psi}^\pi(s, a)$ is observed, one can compute a task-specific reward mapper based on the observations seen in the same environment with different reward function. Given $\boldsymbol{\psi}^\pi(s, a)$,

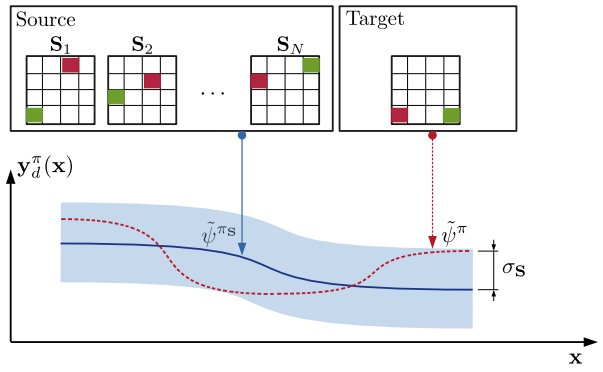

Figure 1. Our proposed approach uses GPs to model the source successor features functions ($\tilde{\psi}^{\pi_\mathbf{S}}$) as noisy measurements for the target successor features functions ($\tilde{\psi}^{\pi}$).

the updated reward mapper $\tilde{\mathbf{w}}$ for a new reward function can be approximated by solving a regression problem in tabular scenarios (Barreto et al., 2020). This enables us to construct the new action-value function $\tilde{Q}^{\pi}(s,a) = \boldsymbol{\psi}^{\pi}(s,a)^{\mathrm{T}}\tilde{\mathbf{w}}$ by only observing few steps of the new reward function in a similar environment. In non-tabular problems, a deep neural network can be used to learn the successor feature functions of an environment by minimising the following loss:

$$L_{\psi}(\boldsymbol{\theta}) = ||\tilde{\boldsymbol{\phi}}(s_t,a_t) + \gamma^t \tilde{\boldsymbol{\psi}}^{\pi}(s_{t+1},a_{t+1};\boldsymbol{\theta}_{\psi}) - \tilde{\boldsymbol{\psi}}^{\pi}(s_t,a_t;\boldsymbol{\theta}_{\psi})||. \quad (5)$$

Similarly, by having the $\boldsymbol{\phi}(.,.)$ function, the new task-specific reward mapper $\tilde{\mathbf{w}}$ is computed as:

$$\mathcal{L}_w(\tilde{\mathbf{w}}) = \mathbb{E}_{\forall(s,a)\in\text{new task}}\Big[\big(r(s,a) - \boldsymbol{\phi}(s,a)^{\mathrm{T}}\tilde{\mathbf{w}}\big)^2\Big]. \quad (6)$$

where $r(s,a)$ is the obtained reward.

3. Method

We now consider source and target environments with both *dissimilar dynamics* and *different reward functions*. The goal of transfer in RL in such problems is to learn an optimal policy for the target environment, by leveraging exterior source information and interior target information (Joy et al., 2019; Shilton et al., 2017). We first define the set of source environments as:

$$\mathcal{M}^{\mathbf{S}} = \Big\{\mathcal{M}(\mathcal{S},\mathcal{A},p_1,R_1),\ldots,\mathcal{M}(\mathcal{S},\mathcal{A},p_N,R_N)\Big\},$$

where $\mathcal{M}(.)$ represents an MDP induced by $\boldsymbol{\phi}(.,.)$ as a feature function used in all environments. We denote the N source environments by $\mathbf{S}_1,\ldots,\mathbf{S}_N$ and train an optimal policy in each environment to create a set of policies: $\Pi^{\mathbf{S}} = \{\pi^{\mathbf{S}_1},\ldots,\pi^{\mathbf{S}_N}\}$. By executing these N optimal policies in their corresponding source environments, N distinct successor feature functions $\boldsymbol{\psi}^{\pi_i}$, $i=\{1,\ldots,N\}$ are computed using the loss function Eq. (5). We define $\boldsymbol{\psi}^{\pi_i} = \boldsymbol{\psi}_1^{\pi_i},\ldots,\boldsymbol{\psi}_D^{\pi_i}$ and $\boldsymbol{\psi}_d^{\pi_i}$ is the d-th dimension. Source successor feature functions can be learnt using neural networks with parameters $\boldsymbol{\theta}_{\psi}$ that can be updated by gradient descent. Algorithm 1 shows how the successor feature functions are computed.

Algorithm 1 Extracting Successor Feature Functions of Source Environments.

Input:
Source observation and learned policies $\mathcal{D}^{\mathbf{S}_{i=1\ldots N}} = \{\}$, $\Pi^{\mathbf{S}}$.
Discount factor γ, exploration rate ε_e.
Feature function $\boldsymbol{\phi}(.,.) \in \mathbb{R}^D$.
$M_{\max}$ maximum number of episodes.
Output: Successor feature functions in source environments $\tilde{\boldsymbol{\psi}}^{\pi_1},\ldots,\tilde{\boldsymbol{\psi}}^{\pi_N}$.
Initialise $\tilde{\boldsymbol{\psi}}^{\pi_{1\ldots N}} : \mathcal{S}\times\mathcal{A}\rightarrow\mathbb{R}^D$.
for $i \in 1,\ldots,N$ **do**
  while $M_{\max}$ **do**
    Sample the initial state randomly $s\in\mathcal{S}$.
    while $t \in$ steps not terminated **do**
      if $\varepsilon-$ greedy **then**
        a_t =Uniform $(\mathcal{A})$. *//random action*
      else
        $a_t = \pi^{\mathbf{S}_i}(s_t)$.
      end if
      Execute a_t, observe s_{t+1}.
      $a_{t+1} = \pi^{\mathbf{S}_i}(s_{t+1})$.
      Minimise the loss $L_{\psi^{\pi_i}}(\boldsymbol{\theta})$. *//Eq. (5)*
      Store the transition $\big[s_t,a_t,\tilde{\boldsymbol{\psi}}^{\pi_i}(s_t,a_t;\boldsymbol{\theta}_{\psi^{\pi_i}})\big]$ in $\mathcal{D}^{\mathbf{S}_i}$.
      Perform gradient descent w.r.t. $\boldsymbol{\theta}_{\psi^{\pi_i}}$.
    end while
  end while
end for

Using state-action pairs as observations, we construct a set of successor feature function samples as: $\mathcal{D}^{\mathbf{S}_i} = \Big\{\big(\mathbf{x}_1^{\mathbf{S}_i},\tilde{\boldsymbol{\psi}}^{\pi_i}(\mathbf{x}_1^{\mathbf{S}_i})\big),\big(\mathbf{x}_2^{\mathbf{S}_i},\tilde{\boldsymbol{\psi}}^{\pi_i}(\mathbf{x}_2^{\mathbf{S}_i})\big),\ldots,\big(\mathbf{x}_n^{\mathbf{S}_i},\tilde{\boldsymbol{\psi}}^{\pi_i}(\mathbf{x}_n^{\mathbf{S}_i})\big)\Big\}$, where $\mathbf{x}_t^{\mathbf{S}_i} = (s_t,a_t)$ is a tuple of state-action in $\mathbf{S}_i$ at time t following policy π_i and $\tilde{\boldsymbol{\psi}}^{\pi_i}(\mathbf{x}_t^{\mathbf{S}_i})$ is the successor feature for $\mathbf{x}_t^{\mathbf{S}_i}$. If observations from target environment exist, a set of successor feature function samples in the target environment can be constructed as $\mathcal{D}^{\mathcal{T}} = \Big\{\big(\mathbf{x}_1^{\mathcal{T}},\tilde{\boldsymbol{\psi}}^{\pi}(\mathbf{x}_1^{\mathcal{T}})\big),\big(\mathbf{x}_2^{\mathcal{T}},\tilde{\boldsymbol{\psi}}^{\pi}(\mathbf{x}_2^{\mathcal{T}})\big),\ldots,\big(\mathbf{x}_m^{\mathcal{T}},\tilde{\boldsymbol{\psi}}^{\pi}(\mathbf{x}_m^{\mathcal{T}})\big)\Big\}$, where $\mathbf{x}_t^{\mathcal{T}} = (s_t,a_t)$ records a tuple of state-action visited at time step t following policy π in the target environment $\mathcal{T}$, and $\tilde{\boldsymbol{\psi}}^{\pi}(\mathbf{x}_t^{\mathcal{T}})$ is an approximation of successor feature of $\mathbf{x}_t^{\mathcal{T}}$ in the target environment.

We assume that the target successor feature function $\tilde{\boldsymbol{\psi}}_d^{\pi}$, fol-

lowing policy π has a measurement noise of $\epsilon_d \sim \mathcal{N}(0, \sigma^2)$, i.e.:

$$\mathbf{y}_d^{\pi}(\mathbf{x}^{\mathcal{T}}) = \tilde{\boldsymbol{\psi}}_d^{\pi}(\mathbf{x}^{\mathcal{T}}) + \epsilon_d,\ \forall d \in \{1, \ldots, D\},$$

where $\mathbf{y}_d^{\pi}(\mathbf{x}^{\mathcal{T}})$ represents the noisy value of successor features in the target environment. Likewise, source successor features are assumed to have a measurement noise of $\epsilon_d \sim \mathcal{N}(0, \sigma^2)$, i.e.:

$$\mathbf{y}_d^{\pi_i}(\mathbf{x}^{\mathbf{S}_i}) = \tilde{\boldsymbol{\psi}}_d^{\pi_i}(\mathbf{x}^{\mathbf{S}_i}) + \epsilon_d,\ \forall d \in \{1, \ldots, D\},\ i = \{1, \ldots, N\}.$$

We model the samples of successor features from the source environment as noisy variants of successor features in the target environment. Under this model, the target successor feature for observation $\mathbf{x}^{\mathcal{T}}$ is defined as:

$$\tilde{\boldsymbol{\psi}}_d^{\pi}(\mathbf{x}^{\mathcal{T}}) = \tilde{\boldsymbol{\psi}}_d^{\pi_i}(\mathbf{x}^{\mathcal{T}}) + \epsilon_d^{\mathbf{S}_i},\ \forall d \in \{1, \ldots, D\},$$

where $\epsilon_d^{\mathbf{S}_i}$ is the modelling noise of target successor feature functions. We assume the modeling noise to be Gaussian distributed with variance $\sigma_{\mathbf{S}}^2$ as $\epsilon_d^{\mathbf{S}_i} = \epsilon_d^{\mathbf{S}} \sim \mathcal{N}(0, \sigma_{\mathbf{S}}^2),\ i = \{1, \ldots, N\}$. Intuitively, this allows the use of the source samples as a noisy version for the target successor feature values. The value of the "modeling noise" variance $\sigma_{\mathbf{S}}^2$ depends on the difference between source and target environments (Shilton et al., 2017). Using the successor feature samples from both source and target environments, we model the target successor feature function using a Gaussian Process . The overall idea is illustrated in Figure 1. Without loss of generality, we use $\mathcal{GP}(0, k(.,.))$, i.e. a GP with a zero mean function, a symmetric positive-definite covariance function $k(\mathbf{x}, \mathbf{x}') : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ a.k.a. kernel, and we also assume $k(\mathbf{x}, \mathbf{x}) = 1, \forall \mathbf{x} \in \mathbb{X}$. A covariance matrix $\mathbf{K}$ based on the combined source and target samples of successor features can be written as:

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1^{\mathbf{S}}, \mathbf{x}_1^{\mathbf{S}}) & \ldots & k(\mathbf{x}_1^{\mathbf{S}}, \mathbf{x}_n^{\mathbf{S}}) & k(\mathbf{x}_1^{\mathbf{S}}, \mathbf{x}_1^{\mathcal{T}}) & \ldots & k(\mathbf{x}_1^{\mathbf{S}}, \mathbf{x}_m^{\mathcal{T}}) \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ k(\mathbf{x}_n^{\mathbf{S}}, \mathbf{x}_1^{\mathbf{S}}) & \ldots & k(\mathbf{x}_n^{\mathbf{S}}, \mathbf{x}_n^{\mathbf{S}}) & k(\mathbf{x}_n^{\mathbf{S}}, \mathbf{x}_1^{\mathcal{T}}) & \vdots & k(\mathbf{x}_n^{\mathbf{S}}, \mathbf{x}_m^{\mathcal{T}}) \\ k(\mathbf{x}_1^{\mathcal{T}}, \mathbf{x}_1^{\mathbf{S}}) & \ldots & k(\mathbf{x}_1^{\mathcal{T}}, \mathbf{x}_n^{\mathbf{S}}) & k(\mathbf{x}_1^{\mathcal{T}}, \mathbf{x}_1^{\mathcal{T}}) & \vdots & k(\mathbf{x}_1^{\mathcal{T}}, \mathbf{x}_m^{\mathcal{T}}) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_m^{\mathcal{T}}, \mathbf{x}_1^{\mathbf{S}}) & \ldots & k(\mathbf{x}_m^{\mathcal{T}}, \mathbf{x}_n^{\mathbf{S}}) & k(\mathbf{x}_m^{\mathcal{T}}, \mathbf{x}_1^{\mathcal{T}}) & \ldots & k(\mathbf{x}_m^{\mathcal{T}}, \mathbf{x}_m^{\mathcal{T}}) \end{bmatrix}, \tag{7}$$

where $k(\mathbf{x}_i^{\mathbf{S}}, \mathbf{x}_j^{\mathbf{S}}),\ i, j = \{1, ..., n\}$ is the self-covariance among source observations of $\mathbf{S} \in \{\mathbf{S}_1, \ldots, \mathbf{S}_N\}$ and $k(\mathbf{x}_i^{\mathbf{S}}, \mathbf{x}_j^{\mathcal{T}}),\ i = \{1, ..., n\}, j = \{1, ..., m\}$ denotes the covariance between source and target observations. After incorporating the source, target measurement noise with the modeling noise, the covariance matrix can be written as:

$$\mathbf{K}_* = \mathbf{K} + \begin{bmatrix} (\sigma_{\mathbf{S}}^2 + \sigma^2)\mathbf{I}_{n \times n} & \mathbf{0} \\ \mathbf{0} & \sigma^2 \mathbf{I}_{m \times m} \end{bmatrix}. \tag{8}$$

Intuitively, a higher value of $\sigma_{\mathbf{S}}^2$ implies higher uncertainty about similarity between source and target environments. Having defined the required components, using the property of GP (Rasmussen, 2003), the predictive mean and variance for a new target observation $\mathbf{x}^{\mathcal{T}} = (s, a)$ is derived as:

$$\mu_{m,d}(\mathbf{x}^{\mathcal{T}}) = \mathbf{k}^{\mathrm{T}} \mathbf{K}_*^{-1} \mathbf{y}_d, \tag{9}$$

$$\sigma_{m,d}^2(\mathbf{x}^{\mathcal{T}}) = k(\mathbf{x}^{\mathcal{T}}, \mathbf{x}^{\mathcal{T}}) - \mathbf{k}^{\mathrm{T}} \mathbf{K}_*^{-1} \mathbf{k}, \tag{10}$$

where $\mathbf{K}_*$ is the kernel matrix as defined in Eq. (8) and $\mathbf{k} = [k(\mathbf{x}_i, \mathbf{x}^{\mathcal{T}})], \forall \mathbf{x}_i \in \mathcal{D}^{\mathbf{S}} \bigcup \mathcal{D}^{\mathcal{T}}$. We use the posterior mean as in Eq. (9) as the predicted value of the d-th successor feature dimension for the target observation as $\tilde{\boldsymbol{\psi}}_d^{\pi}(\mathbf{x}^{\mathcal{T}})) = \mu_{m,d}(\mathbf{x}^{\mathcal{T}})$.

Using GPI (Barreto et al., 2017), we identify the optimal policy by selecting the best action of the best policy as:

$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \max_{\pi \in \Pi^{\mathbf{S}}} \tilde{Q}^{\pi}(s, a). \tag{11}$$

We note that $\tilde{Q}^{\pi}(\mathbf{x}^{\mathcal{T}}) \approx \tilde{\boldsymbol{\psi}}^{\pi}(\mathbf{x}^{\mathcal{T}})^{\mathrm{T}} \tilde{\boldsymbol{w}}$, where $\tilde{\boldsymbol{w}}$ is obtained by minimising the loss function in Eq. (6) as the agent interacts with the target environment. We term our method *Successor Features for Dissimilar Environments (SFDE)* and is detailed in Algorithm 2.

3.1. Theoretical Analysis

This section answers the key question of "what are the effects of relaxing the assumption of similarity among source and target environments on the convergence and transfer via successor features?". We prove (1) an upper bound on the difference of action-value functions when the optimal policy derived from environment i is replaced by the optimal policy derived from environment j (*Theorem 1*); (2) an upper bound on the estimation error of the action-value function of an optimal policy learned in $\mathbf{S}_j$ when executed in target environment $\mathcal{T}$ (*Lemma 1*); (3) Using (1) and (2) an upper bound on the difference of the optimal action-value function in the target environment and our GPI-derived action value function (*Theorem 2*).

Theorem 1

Let $\mathbf{S}_i$ and $\mathbf{S}_j$ be two different source environments with dissimilar transition dynamics p_i and p_j respectively. Let $\delta_{ij} \triangleq \max_{s,a} |r_i(s, a) - r_j(s, a)|$, where $r_i(.,.)$ and $r_j(.,.)$ are the reward functions of environment $\mathbf{S}_i$ and $\mathbf{S}_j$ respectively. We denote π_i^* and π_j^* as optimal policies in $\mathbf{S}_i$ and $\mathbf{S}_j$. It can be shown that the difference of their action-value functions is upper bounded as:

$$Q_i^{\pi_i^*}(s, a) - Q_i^{\pi_j^*}(s, a) \leq \frac{2\delta_{ij}}{1 - \gamma} + \frac{\gamma \left|\left| \mathbf{P}_i(s, a) - \mathbf{P}_j(s, a) \right|\right|}{(1 - \gamma)} \times \frac{\left(\left|\left| \mathbf{Q}_i^i - \mathbf{Q}_j^i \right|\right| + \left|\left| \mathbf{Q}_j^j - \mathbf{Q}_i^j \right|\right| \right)}{(1 - \gamma)}, \tag{12}$$

where $Q_i^{\pi_k^*}$ shows the action-value function in environment $\mathbf{S}_i$ by following an optimal policy that is learned in the environment $\mathbf{S}_k \in \{\mathbf{S}_1, \ldots, \mathbf{S}_N\}$. We also define $\mathbf{P}_i(s,a) = [p_i(s'|s,a), ...]_{\forall s' \in \mathcal{S}}$, $\mathbf{P}_j(s,a) = [p_j(s'|s,a), ...]_{\forall s' \in \mathcal{S}}$, $\mathbf{Q}_i^i = [\max_{b \in \mathcal{A}} Q_i^{\pi_i^*}(s',b), ...]_{\forall s' \in \mathcal{S}}$, $\mathbf{Q}_j^j = [\max_{b \in \mathcal{A}} Q_j^{\pi_j^*}(s',b), ...]_{\forall s' \in \mathcal{S}}$, $\mathbf{Q}_i^j = [\max_{b \in \mathcal{A}} Q_i^{\pi_j^*}(s',b), ...]_{\forall s' \in \mathcal{S}}$, γ as the discount factor, and $||.||$ to be $2-$norm (Euclidean norm).

Proof: We provide a sketch of the proof which involves two key steps. The left side of the inequality (12) can be rewritten as:

$$
\begin{aligned}
Q_i^{\pi_i^*}(s,a) - Q_i^{\pi_j^*}(s,a) &= Q_i^{\pi_i^*}(s,a) - Q_j^{\pi_j^*}(s,a) \\
&\quad + Q_j^{\pi_j^*}(s,a) - Q_i^{\pi_j^*}(s,a) \\
&\leq \underbrace{|Q_i^{\pi_i^*}(s,a) - Q_j^{\pi_j^*}(s,a)|}_{\text{(I)}} \\
&\quad + \underbrace{|Q_j^{\pi_j^*}(s,a) - Q_i^{\pi_j^*}(s,a)|}_{\text{(II)}}.
\end{aligned}
$$

We can prove that (I) $\leq \frac{\delta_{ij}}{1-\gamma} + \gamma \left|\left| \mathbf{P}_i - \mathbf{P}_j \right|\right| \times \left|\left| \mathbf{Q}_i^i - \mathbf{Q}_j^j \right|\right| / (1-\gamma)$ and (II) $\leq \frac{\delta_{ij}}{1-\gamma} + \gamma \left|\left| \mathbf{P}_i - \mathbf{P}_j \right|\right| \times \left|\left| \mathbf{Q}_j^j - \mathbf{Q}_i^j \right|\right) / (1-\gamma)$, leading to the upper bound. Detailed proof is available in supplementary material.

Theorem 1 uses δ_{ij} as a metric of maximum immediate reward dissimilarities in the environments $\mathbf{S}_i$ and $\mathbf{S}_j$. Clearly, the higher δ_{ij}, the less similar $\mathbf{S}_i$ and $\mathbf{S}_j$ are, and the upper bound will be looser accordingly. This upper bound also depends on the value of $\left|\left| \mathbf{P}_i - \mathbf{P}_j \right|\right|$ that captures the difference in dynamics of $\mathbf{S}_i$ and $\mathbf{S}_j$ - larger the value, looser the upper bound. However, $\left|\left| \mathbf{Q}_i^i - \mathbf{Q}_j^j \right|\right| + \left|\left| \mathbf{Q}_j^j - \mathbf{Q}_i^j \right|\right|$ incorporates the difference of action-value functions that are related to both the dynamics and the future discounted reward in the two environments. Hence, a larger value of this term implies that $\mathbf{S}_i$ and $\mathbf{S}_j$ are expected to produce different sum of discounted future reward by following their corresponding policies. Note that if $\mathbf{S}_i = \mathbf{S}_j$ (in terms of both dynamics and reward), the upper bound will vanish to 0 as the two environments are identical. Clearly, our bound is an extension of the bound in (Barreto et al., 2018) for environments with dissimilar dynamics. In the special case of identical environments i.e. when $\left|\left| \mathbf{P}_i - \mathbf{P}_j \right|\right| = 0$, the two bounds become the same.

We now prove an upper bound on the estimation error of the action-value function of an optimal policy learned in $\mathbf{S}_j$ when executed in the target environment $\mathcal{T}$.

Algorithm 2 Successor Features for Dissimilar Environments.

Input:
Source environments $\mathcal{D}^{\mathbf{S}_{1\ldots N}}$ and target observations $\mathcal{D}^{\mathcal{T}} = \{\}$.
Set the amount of noises for source and target $\sigma_{\mathbf{S}}^2, \sigma^2$.
SFs of source environments $\tilde{\boldsymbol{\psi}}^{\pi_1}, \ldots, \tilde{\boldsymbol{\psi}}^{\pi_N}$.
Feature function $\boldsymbol{\phi}(.,.) \in \mathbb{R}^D$.
Initialise reward mapper weight for target environment $\tilde{\boldsymbol{w}}$.
Output: Optimal policy $\pi^*(s)$ for the target environment.
while $t \in$ steps not terminated **do**
  for $\forall \mathbf{S} \in \{\mathbf{S}_1, \ldots, \mathbf{S}_N\}$ **do**
    for $a' \in \mathcal{A}$ **do**
      $\mathbf{x}_t^{\mathcal{T}} = (s_t, a')$.
      $\mathcal{GP}_d^{\mathbf{S}}(\{\mathcal{D}^{\mathbf{S}}, \mathcal{D}^{\mathcal{T}}\}),\ \forall d = \{1, \ldots, D\}$. //Fit the GPs with source and target data
      Calculate $\mathbf{K}_*$, k. //(Eq. 8)
      $\tilde{\boldsymbol{\psi}}_d^{\pi_{\mathbf{S}}}(\mathbf{x}_t^{\mathcal{T}})) = \mu_{m,d}^{\mathbf{S}}(\mathbf{x}_t^{\mathcal{T}}).\ \forall d = \{1, \ldots, D\}$. //(Eq. 9)
    end for
  end for
  Reconstruct the action-value functions of all source environments given the target observations $\tilde{Q}^{\pi_i}(s_t, a') = \tilde{\boldsymbol{\psi}}^{\pi_i}(s_t, a')\tilde{\boldsymbol{w}}, \forall a' \in \mathcal{A}, \forall i = \{1, \ldots, N\}$.
  $\pi^*(s) = \underset{a' \in \mathcal{A}}{\operatorname{argmax}} \max_{\pi \in \Pi^{\mathbf{S}}} \tilde{Q}^{\pi}(\mathbf{x}_t^{\mathcal{T}})$. //Performing GPI
  Add the new target observation to $\mathcal{D}^{\mathcal{T}}$.
  Update the estimation of $\tilde{\boldsymbol{w}}$ based on Eq. (6).
end while

Lemma 1

Let $\pi_1^*, ..., \pi_N^*$ be N optimal policies for $\mathbf{S}_1, \ldots, \mathbf{S}_N$ respectively and $\tilde{Q}_{\mathcal{T}}^{\pi_j^*} = \left(\tilde{\boldsymbol{\psi}}^{\pi_j^*}\right)^{\mathrm{T}} \tilde{\mathbf{w}}_{\mathcal{T}}$ denote the action-value function of an optimal policy learned in $\mathbf{S}_j$ and executed in the target environment $\mathcal{T}$. Let $\tilde{\boldsymbol{\psi}}^{\pi_j^*}$ denote the estimated successor feature function from the combined source and target observations from $\mathbf{S}_j$ and $\mathcal{T}$ as defined in Eq. (9), and $\tilde{\mathbf{w}}_{\mathcal{T}}$ is the estimated reward mapper for environment $\mathcal{T}$ by using loss function in Eq. (6). It can be shown that the difference of the true action-value function and the estimated one through successor feature functions and reward mapper, is bounded as:

$$
\Pr\left(\left| Q_{\mathcal{T}}^{\pi_j^*}(s,a) - \tilde{Q}_{\mathcal{T}}^{\pi_j^*}(s,a) \right| \leq \varepsilon(m)\ \forall s, a \right) \geq 1 - \delta,
$$

where $\varepsilon(m) = \sqrt{2\log(|\mathbb{X}|u_m/\delta)}\sigma_{m,d}(\mathbf{x})$, $\mathbf{x} \in \mathbb{X}$ $\delta \in (0,1)$, $u_m = \frac{\pi^2 m^2}{6}$, m being the number of observations in environment $\mathcal{T}$, and $\mathbf{x} = (s,a)$. $\sigma_{m,d}(\mathbf{x})$ is the square root of posterior variance as defined in Eq. (10).

Proof: Proof is available in the supplementary material.

Lemma 1 ensures that reconstructing the action-value function on a new target environment $\mathcal{T}$ by using $\tilde{\boldsymbol{\psi}}^{\pi_j^*}$ and $\tilde{\mathbf{w}}_{\mathcal{T}}$ can be achieved with a bounded error with high probability. Essentially, the key term in $\varepsilon(m)$ is $\sigma^2_{m,d}(\mathbf{x})$ that is computed by using $\mathbf{K}_*$ (see Eq. (8)), which itself incorporates the modeling noise variance $\sigma^2_{\mathbf{S}}$. Hence, by increasing $\sigma^2_{\mathbf{S}}$, $\sigma^2_{m,d}(\mathbf{x})$ will be higher and accordingly the upper bound will become looser. We note that due to the consistency of GPs, as $m \to \infty$, the uncertainty of predictions tends to 0 ($\sigma^2_{m,d}(\mathbf{x}) \to 0$) and thus $\varepsilon(m) \to 0$.

We now present our final result that bounds the difference of the optimal action-value function in the target environment and our GPI-derived action value function.

Theorem 2

Let $\mathbf{S}_{i=1\ldots N}$ be N different source environments with dissimilar transition functions $p_{i=1\ldots N}$. Let us denote the optimal policy π that is defined based on the GPI as:

$$\pi(s) \in \operatorname*{argmax}_{\mathrm{a}\in\mathcal{A}} \max_{\mathrm{j}\in\{1\ldots\mathrm{N}\}} \tilde{Q}_{\mathcal{T}}^{\pi_j^*}(s,a), \tag{13}$$

where $\tilde{Q}_{\mathcal{T}}^{\pi_j^*} = \left(\tilde{\boldsymbol{\psi}}^{\pi_j^*}\right)^{\mathrm{T}} \tilde{\mathbf{w}}_{\mathcal{T}}$ being the action-value function of an optimal policy learned in $\mathbf{S}_j$ and executed in target environment $\mathcal{T}$, $\tilde{\boldsymbol{\psi}}^{\pi_j^*}$ is the estimated successor feature from the combined source and target observations from $\mathbf{S}_j$ and $\mathcal{T}$ as defined in Eq. (9), and $\tilde{\boldsymbol{w}}_{\mathcal{T}}$ is the estimated reward mapper for target environment from Eq. (6). Considering Lemma 1 and Eq. (13), the difference of optimal action-value function in the target environment and our GPI-derived action value function is upper bounded as:

$$\begin{aligned} Q_{\mathcal{T}}^*(s,a) - \tilde{Q}_{\mathcal{T}}^{\pi\in\pi_j^*}(s,a) \leq{}& \frac{2\phi_{\max}}{1-\gamma}\left|\left|\tilde{\mathbf{w}}_{\mathcal{T}} - \mathbf{w}_j\right|\right| \\ &+ \frac{\gamma\left|\left|\mathbf{P}_{\mathcal{T}}(s,a) - \mathbf{P}_j(s,a)\right|\right|}{(1-\gamma)} \\ &\times \frac{\left(\left|\left|\mathbf{Q}_{\mathcal{T}}^* - \mathbf{Q}_j^j\right|\right| + \left|\left|\mathbf{Q}_j^j - \mathbf{Q}_{\mathcal{T}}^j\right|\right|\right)}{(1-\gamma)} \\ &+ \frac{2\varepsilon(m)}{(1-\gamma)}. \end{aligned} \tag{14}$$

where $\phi_{\max} = \max_{s,a}||\phi(s,a)||$. We also define $\mathbf{P}_{\mathcal{T}} = [p_{\mathcal{T}}(s'|s,a),...]_{\forall s'\in\mathcal{S}}$, $\mathbf{P}_j = [p_j(s'|s,a),...]_{\forall s'\in\mathcal{S}}$, $\mathbf{Q}_{\mathcal{T}}^* = [\max_{b\in\mathcal{A}} Q_{\mathcal{T}}^*(s',b),...]_{\forall s'\in\mathcal{S}}$, $\mathbf{Q}_j^j = [\max_{b\in\mathcal{A}} \tilde{Q}_j^{\pi_j^*}(s',b),...]_{\forall s'\in\mathcal{S}}$, $\mathbf{Q}_{\mathcal{T}}^j = [\max_{b\in\mathcal{A}} \tilde{Q}_{\mathcal{T}}^{\pi_j^*}(s',b),...]_{\forall s'\in\mathcal{S}}$, and γ as the discount factor.

Proof: Proof is available in supplementary material.

In inequality (14), $\left|\left|\tilde{\mathbf{w}}_{\mathcal{T}} - \mathbf{w}_j\right|\right|$ encodes the dissimilarity of reward functions in target environment $\mathcal{T}$ and $\mathbf{S}_j$, and $\varepsilon(m)$ holds the error of action-value reconstruction by GP-modelled successor features (Lemma 1). Additionally, $\gamma||\mathbf{P}_{\mathcal{T}} - \mathbf{P}_j|| \times (||\mathbf{Q}_{\mathcal{T}}^* - \mathbf{Q}_j^j|| + ||\mathbf{Q}_j^j - \mathbf{Q}_{\mathcal{T}}^j||)/(1-\gamma)$ is a term related to both the dissimilarity of dynamics in environment $j \in \{1,\ldots.N\}$ and the target environment $\mathcal{T}$, as well as the expected future reward in these two environments. We note that Eq. (13) enforces the selection of the best policy among $j \in \{1,\ldots.N\}$ policies, hence the derived upper bound is only based on the best selected policy among N source environments.

Theorem 2 is the core of our theoretical analysis that shows by using N different action-value functions that are obtained by their corresponding successor feature functions and reward mapper weights, one can still hold an upper bound on the difference of the optimal policy $Q_{\mathcal{T}}^*(s,a)$, and $\tilde{Q}_{\mathcal{T}}^{\pi}(s,a)$, if $\pi(s)$ is selected by GPI as defined in Eq. (13). As expected, this upper bound is looser when the norm distance between $\tilde{\mathbf{w}}_{\mathcal{T}}$ and $\mathbf{w}_j$ are higher - i.e. the reward functions are significantly different. As explained in Lemma 1, by increasing the amount of noise when modeling the source successor feature functions, $\sigma_{m,d}(\mathbf{x})$ will be higher and accordingly the upper bound will become looser since $\varepsilon(m)$ increases. This is reasonable as increasing $\sigma_{\mathbf{S}}$ indicates that source and target environment are significantly dissimilar. Note that if the environments are exactly similar, $\gamma||\mathbf{P}_{\mathcal{T}} - \mathbf{P}_j|| \times (||\mathbf{Q}_{\mathcal{T}}^* - \mathbf{Q}_j^j|| + ||\mathbf{Q}_j^j - \mathbf{Q}_{\mathcal{T}}^j||)/(1-\gamma) = 0$, this leads to a similar upper bound in (Barreto et al., 2018). Further analysis on the obtained upper bound can be found in supplementary materials.

4. Experiments

We evaluate the performance of our method on 3 benchmarks: (1) A toy navigation problem, (2) Classic CartPole control, and (3) The environment introduced by Barreto et al. (Barreto et al., 2020; 2019). We compare our approach, Successor Features for Dissimilar Environments (*SFDE*) with two related studies: Fast Successor Features (*FSF*) (Barreto et al., 2020) and Linear Projection of Successor Features (*LPSF*) (Zhang et al., 2017). Additional experiments are available in the supplementary materials. All the algorithms, including SFDE, are used in two phases: (1) The first phase is adaptation where we fine-tune the source successor feature functions using the first 1000 target observations. This step is method specific. (2) A testing phase in which we only use the learnt policy and collect reward without updating the models. For the adaptation phase, ϵ-greedy based exploration is used, afterwards we set the exploration rate $\varepsilon_e = 0$ in the testing phase to demonstrate the effects of transfer from previous environments. The baseline FSF is not equipped to handle environments with dissimilar dynamics. We adapt this approach by fine-tuning the successor feature functions of the source environments

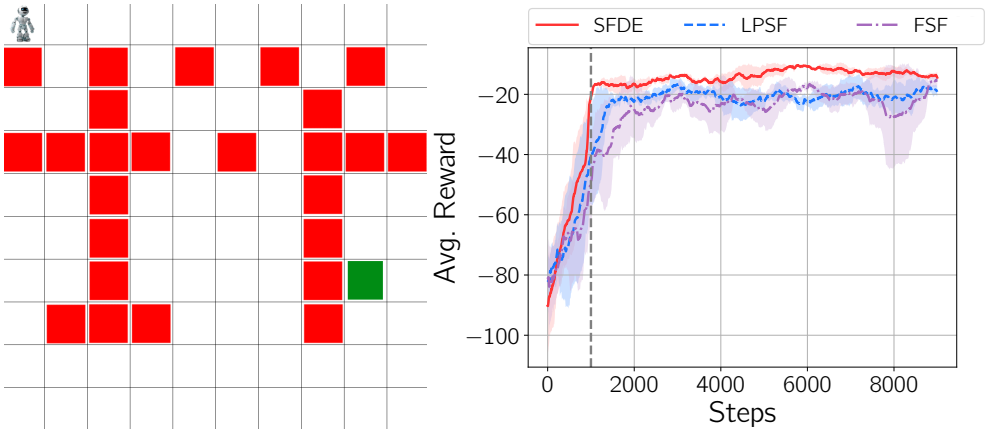

Figure 2. (left) Sample of proposed maze environment with red squares as obstacles with -50 reward and green square as goal with +100 reward. (Right) Obtained results of SFDE and two baselines. The results are averaged over 50 runs. The dashed vertical line demarcates the adaptation phase.

using the first stored 1000 observations of the target environment in the adaptation phase. A batch size of 64 is used at every step $64 \leq t \leq 1000$ to feed the new target observations to the previously learned source successor features. Once the testing phase starts ($t > 1000$), we stop fine-tuning of the successor models for the rest of the experiment. For the LPSF baseline, we follow the idea of (Zhang et al., 2017) by defining a linear relation between the source and target successor feature - that is there exists a mapping $\boldsymbol{\beta}$=$\boldsymbol{\beta}_1 ... \boldsymbol{\beta}_N$ such that the following loss function is minimised: $\mathcal{L}_{\boldsymbol{\beta}}(\boldsymbol{\theta}_\beta) = \sum_{i=1...N} \left|\left|\tilde{\psi}^{\mathcal{T}}(\mathbf{x}) - \boldsymbol{\beta}_i \tilde{\psi}^{\mathbf{S}_i}(\mathbf{x})\right|\right|, \forall \mathbf{x} \in \mathcal{D}^{\mathcal{T}}$. Intuitively, the best linear projection is found for each source successor feature function to minimise its distance to the target successor. We follow the same approach explained for FSF to feed the target observations in adaptation phase to N neural networks that each represent the model of $i-$th source successor features $i = \{1, \ldots, N\}$ with $\mathcal{L}_{\boldsymbol{\beta}}$ loss function. Each of these N neural networks are MLPs with no hidden layers that is an equivalent of linear regression in which the weights of the neural networks are $\boldsymbol{\beta}_i$. Likewise, the best obtained value of $\boldsymbol{\beta}$ in the adaptation phase is used in the testing phase. We used the same batch size of 64 to minimise $\mathcal{L}_{\boldsymbol{\beta}}$ loss function for LPSF. The linearly projected successor features then used in the GPI framework to obtain the optimal policy. In our approach, we construct the GP based on the combination of source and target observations. To this end, 500 randomly sampled observations from source $\mathcal{D}^{\mathbf{S}}$, $\forall \mathbf{S} \in \{\mathbf{S}, \ldots, \mathbf{S}_N\}$ and all the observations in $\mathcal{D}^{\mathcal{T}}$ at every step of the adaptation phase is used. However, once the testing phase is initiated, the new target observations are disregarded and previously seen observations from the target are reused. Further details of implementations are available in the supplementary material.

4.1. Toy Navigation Problem

The proposed maze problem consists of a 10×10 grid and the agent needs to find the goal in the maze. The agent can pass through the obstacles but it receives a reward of -50. It also receives -1 reward for each step and $+100$ reward for reaching the goal. The action set is defined as $\mathcal{A} = \{\text{left}, \text{right}, \text{up}, \text{down}\}$. Figure 2 (left) shows an example of this environment with red squares indicating obstacles, and the green square as the goal.

A set of 12 policies $\Pi^{\mathbf{S}} = \left\{\pi^{\mathbf{S}_1}, \ldots, \pi^{\mathbf{S}_{12}}\right\}$ is learnt using generic Q-Learning on randomly generated maze source environments $\left\{\mathcal{M}(\mathcal{S}, \mathcal{A}, p_1, R_1), \ldots, \mathcal{M}(\mathcal{S}, \mathcal{A}, p_{12}, R_{12})\right\}$ as $\mathbf{S}_1, \ldots, \mathbf{S}_{12}$ in which the location of 25 obstacles and goal are changed. After obtaining the corresponding policies, successor feature functions of these source environments are learnt following Algorithm 1. Given $\tilde{\boldsymbol{\psi}}^{\pi_1}, \ldots, \tilde{\boldsymbol{\psi}}^{\pi_{12}}$, we generate a random target environment and use the first 1000 observations of adaptation phase as explained. Following Algorithm 2, $\mathcal{GP}^{1...12}$ is constructed as each step by fitting both source and target observations. We set $\sigma_{\mathbf{S}}^2 = 0.1$ and $\sigma^2 = 0.01$ as the modelling noise and the measurement noise, respectively. Having modelled the successor features, we then perform GPI by using the predicted mean as shown in Eq. (9) and (10). To calculate $\tilde{\boldsymbol{w}}_{\mathcal{T}}$, the reward mapper of target environment is calculated by minimising loss function introduced in Eq. (6). Figure 2 (right) shows the performance of our approach and other baselines. As expected, our method incorporating the dissimilarity of environments performs better than the other two related approaches. Figure 2 shows that LPSF adjusts the source successor features but it seems to be slower than SFDE in updating the successor feature functions as a linear projection may not always be found. It can be seen that FSF can update the values of successor features to some

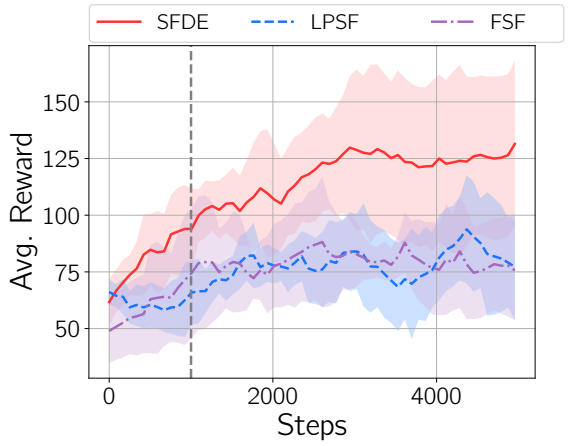

Figure 3. Experiments with CartPole-v0 with $Pole\_Length = 0.5m$ in source environment and $Pole\_Length = 3m$ in target environment. The results are averaged over 10 runs. The dashed vertical line demarcates the adaptation phase.

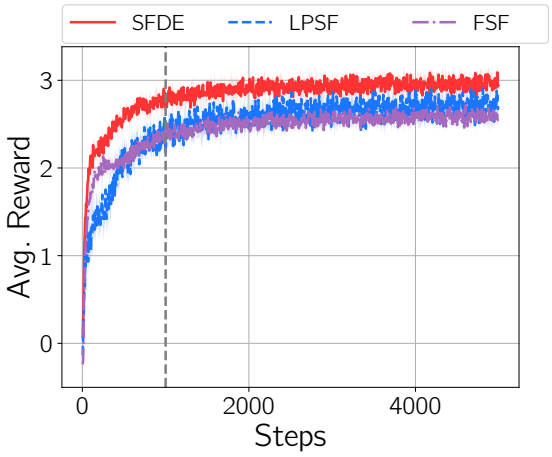

Figure 4. Results on the environment introduced by FSF. The results are averaged over 10 runs. The dashed vertical line demarcates the adaptation phase.

extent, however, it seems to be less effective than the other 2 approaches. This can be the result of fine-tuning the successor feature models with significantly different observations that can be an issue in such scenarios. Note that we have not used any visual information in this experiment and the location of agent is translated to (x, y) in the 10×10 grid.

4.2. CartPole-v0

In the CartPole problem, we define a source environment $\mathbf{S}_1$ with a learnt policy $\pi^{\mathbf{S}_1}$ and a target environment $\mathcal{T}$. We incorporate the dissimilarities of dynamics in source and target by changing the length of the pole from $Pole\_Length = 0.5m$ in source, to $Pole\_Length = 3.0m$ in the target environment. This change impacts the transition probabilities of the target environment, hence, it can be considered as a change in dynamics. Similar to the previous experiment, we use the first 1000 observations in all three methods in a same manner. At each step, we fit $\mathcal{GP}^1$ by combination of source and target observations. We set $\sigma^2_{\mathbf{S}} = 0.1$, $\sigma^2 = 0.01$, and the maximum number of steps in the CartPole is set to 200 in each episode. To translate the image data into states that can be used in our framework, we used the flattened output of the last convolution layer as the state of the CartPole environment. The detailed structure of the proposed network is available in supplementary materials. Figure 3 shows the results of this experiment and it can be seen that our proposed method outperforms both FSF and LPSF by using the GP-based modelled successor features.

4.3. FSF Environment

We use the environment introduced in Barreto et al. (Barreto et al., 2020) as our final experiment. The proposed environment is a 10×10 grid cells with $\mathcal{A} = \{\text{left}, \text{right}, \text{up}, \text{down}\}$. There are 10 objects spread across the grid at all time that the agent can pick up. Once an object is picked up, another random object will appear in the grid randomly. Each object belongs to one of two types (red or blue). At each time step, the agent receives an image showing its position, the position of objects, and type of each object (Barreto et al., 2020). Then the agent selects the proper action to move in a direction. The object is assumed picked up, if the agent occupies the cell in which the object presents. In that case, it gets a reward based on the type of the object and a new object will appear randomly in the grid.

Following the setting of FSF experiments, we trained the agent in 2 different source environments with different reward functions and dynamics of the environment. We incorporated the dissimilarity of environments by adding a 5% random transition noise to the target environment and creating a single random terminal state with a negative reward of -1. Figure 4 shows the obtained results on this problem. Similar to other experiments, our approach seems to outperform the other two baselines. Interestingly, FSF outperforms LPSF in adaptation phase, but both methods approximately converge to the same value of the average reward.

5. Conclusion

In this paper we proposed a novel transfer learning approach based on successor features in RL. Our approach is for the scenarios wherein the source and the target environments have dissimilar reward functions as well as dissimilar environment dynamics. We propose the use of Gaussian

Processes to model the source successor features functions as noisy measurements of the target successor functions. We provide a theoretical analysis on the convergence of our method by proving an upper bound on the error of the optimal policy. We evaluate our method on 3 benchmark problems and showed that our method outperform existing methods.

Acknowledgements

This research was partially funded by the Australian Government through the Australian Research Council (ARC). Prof Venkatesh is the recipient of an ARC Australian Laureate Fellowship (FL170100006).

References

Ammar, H. B. and Taylor, M. E. Reinforcement learning transfer via common subspaces. In *International Workshop on Adaptive and Learning Agents*, pp. 21–36. Springer, 2011.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pp. 4055–4065, 2017.

Barreto, A., Borsa, D., Quan, J., Schaul, T., Silver, D., Hessel, M., Mankowitz, D., Zidek, A., and Munos, R. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pp. 501–510. PMLR, 2018.

Barreto, A., Borsa, D., Hou, S., Comanici, G., Aygün, E., Hamel, P., Toyama, D., Mourad, S., Silver, D., Precup, D., et al. The option keyboard: Combining skills in reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 13052–13062, 2019.

Barreto, A., Hou, S., Borsa, D., Silver, D., and Precup, D. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48):30079–30087, 2020.

Dayan, P. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

Gupta, A., Devin, C., Liu, Y., Abbeel, P., and Levine, S. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

Joy, T. T., Rana, S., Gupta, S., and Venkatesh, S. A flexible transfer learning framework for bayesian optimization with convergence guarantee. *Expert Systems with Applications*, 115:656–672, 2019.

Konidaris, G. and Barto, A. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 489–496, 2006.

Liang, X., Wang, T., Yang, L., and Xing, E. Cirl: Controllable imitative reinforcement learning for vision-based self-driving. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 584–599, 2018.

Ma, C., Ashley, D. R., Wen, J., and Bengio, Y. Universal successor features for transfer reinforcement learning. *arXiv preprint arXiv:2001.04025*, 2020.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Rasmussen, C. E. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pp. 63–71. Springer, 2003.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Shilton, A., Gupta, S., Rana, S., and Venkatesh, S. Regret bounds for transfer learning in bayesian optimisation. In *Artificial Intelligence and Statistics*, pp. 307–315. PMLR, 2017.

Siriwardhana, S., Weerasakera, R., Matthies, D. J., and Nanayakkara, S. Vusfa: Variational universal successor features approximator to improve transfer drl for target driven visual navigation. *arXiv preprint arXiv:1908.06376*, 2019.

Yin, H. and Pan, S. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Zhang, J., Springenberg, J. T., Boedecker, J., and Burgard, W. Deep reinforcement learning with successor features for navigation across similar environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2371–2378. IEEE, 2017.

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