Inter-Task Similarity for Lifelong Cross-Domain Reinforcement Learning

by

Sergio Arredondo Serrano

Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy in Computer Science

in the
Instituto Nacional de Astrofísica, Óptica y Electrónica
September, 2024
Tonantzintla, Puebla

Advisors:

Dr. José Martínez Carranza

Dr. Luis Enrique Sucar Succar

Acknowledgements

First and foremost, I would like to thank CONAHCyT for granting me a scholarship without which I would never have been able to carry out my doctoral studies. I would like to thank my reviewers for their professionalism in criticizing my work, contributing to bringing the final product to a better conclusion. I would like to thank my advisors, Dr. José Martínez Carranza and Dr. Luis Enrique Sucar Succar for their valuable dedication and hard work with which they guided me during the development of the research, as well as for setting an example of professional ethics and human quality.

To my family, for believing in this project and supporting me unconditionally from the beginning. To my mom and dad, for their words of encouragement, which on many occasions were the fuel that kept me going. To my brother, always being there for me when I most needed him and never letting me down.

I would also like to extend my heartfelt thanks to my friends, for going through so many amazing adventures with me, you really have made this ride that much more incredible.

A special mention goes to Jessica, Love, for your sacrifices, understanding, and for always standing by my side. Your constant love and encouragement have been momentous in helping me getting through everything.

This dissertation is a reflection of the collective support I have received, and I dedicate it to all those who have been part of this journey.

Abstract

Solving sequential decision-making problems with artificial agents, and delegating boring and dangerous chores to embodied agents, has the capacity to significantly improve the life quality and safety in the workplace of people. In Reinforcement Learning (RL), decision-making problems can be modeled and solved with weak supervision by interacting with the environment, however, learning robust behaviors for complex tasks usually takes RL agents a large amount of interactions. To mitigate data costs, Lifelong Learning agents leverage experience from previous tasks to learn better/faster, and save knowledge for potential reuse in future problems. In scenarios where the agent can significantly change (*e.g.,* robot with different morphology), being able to identify shared characteristics across tasks is crucial to avoid negative transfer and overcome the data scarcity RL suffers in robotics. Thus, in this dissertation, we propose a similarity-based approach to address the lifelong cross-domain RL problem. By estimating the *relatedness* between tasks based on their reward and transition dynamics, our system selects the most similar task (in the agent's knowledge base), which may or may not share the state-action space. Then, the policy from the selected source task is transferred (through a set of learned mapping functions) to the target task, to accelerate the exploration and learning process. The proposed lifelong learning system is evaluated in a wide variety of control tasks, showing its ability to deal with sequences of diverse problems, and autonomously make an effective use of its experiences.

Keywords: Lifelong Learning, Reinforcement Learning, Transfer Learning, Cross Domain Problem.

Resumen

Resolver problemas de toma de decisiones secuenciales con agentes artificiales, y delegar tareas aburridas y peligrosas a agentes físicos, tiene la capacidad de mejorar significativamente la calidad de vida y la seguridad en el lugar de trabajo de las personas. En el Aprendizaje por Refuerzo (RL, por sus siglas en inglés), los problemas de toma de decisiones pueden ser modelados y resueltos con supervisión débil mediante la interacción con el entorno; sin embargo, aprender comportamientos robustos para tareas complejas generalmente requiere una gran cantidad de interacciones por parte de los agentes RL. Para mitigar los costos de datos, los agentes de Aprendizaje Permanente aprovechan la experiencia de tareas anteriores para aprender mejor y más rápido, y guardar el conocimiento para una posible reutilización en problemas futuros. En escenarios donde el agente puede cambiar significativamente (por ejemplo, un robot con morfología diferente), ser capaz de identificar características compartidas entre tareas es crucial para evitar la transferencia negativa y superar la escasez de datos que sufre el RL en robótica. Por lo tanto, en esta disertación, proponemos un enfoque basado en la similitud para abordar el problema de RL continuo y entre dominios. Al estimar la relación entre tareas basándose en sus dinámicas de recompensa y transición, nuestro sistema selecciona la tarea más similar (almacenada en la base de conocimientos del agente), que puede o no compartir el espacio de estado-acción. Luego, la política de la tarea fuente seleccionada se transfiere (a través de un conjunto de funciones de mapeo aprendidas) a la tarea objetivo, para acelerar el proceso de exploración y aprendizaje. El sistema de aprendizaje permanente propuesto se evalúa en una amplia variedad de tareas de control, demostrando su capacidad para manejar secuencias de problemas diversos y utilizar de manera efectiva sus experiencias de manera autónoma.

Palabras clave: Aprendizaje Permanente, Aprendizaje por Refuerzo, Aprendizaje por Transferencia, Problema Entre Dominios.

Contents

List of Figures

List of Tables

Acronyms

AC Acrobot. 42–46, 48

AGC Adaptive Gradient Clipping. 90

APGT Adaptive Policy Gradient Transfer. 66–73

CNN Convolutional Neural Network. 14

F8 Frozen Lake 8×8. 42–46, 48

FBO Fixed-Buffer Optimization. 66, 68–74, 90

FIFO First In First Out. 67

FL Frozen Lake. 42–46, 48

IDP Inverted Double Pendulum. 66, 68–70, 72, 74, 76, 81–83

IP Inverted Pendulum. 66, 68–70, 72, 74, 75, 82, 83

LL Lifelong Learning. 18

LRL Lifelong Reinforcement Learning. 19, 30, 32, 79–84

LSS Least Similar Source. 66, 68–73

MC Mountain Car. 42–46, 48

MDP Markov Decision Process. 10–14, 18, 20, 23, 25–27, 30, 32, 38, 49

MSS Most Similar Source. 66, 68–74, 80

PCA Principal Component Analysis. 14

PE Pendulum. 42–46, 48

POMDP Partially Observable Markov Decision Process. 11

PPO Proximal Policy Optimization. 66

ReBA Reward-Based Alignment. 56, 68, 74–76, 80, 90

ReLU Rectified Linear Unit. 89, 90

RL Reinforcement Learning. 1, 3–5, 9, 10, 14–18, 23, 27, 29–33, 37, 40, 42, 43, 48, 54, 57, 61–64, 73, 74, 89

SAC Soft Actor-Critic. 66, 68–74, 79, 80, 90

TA Taxi Domain. 42–46, 48

TD Temporal Differences. 13, 14

TL Transfer Learning. 17

TRL Transfer Reinforcement Learning. 17

Chapter 1

Introduction

Thanks to the ability of computers to perform complex calculations with high efficiency, consistency and precision (in comparison to human performance), automating the ability to make decisions with artificial agents has the potential to improve people's safety in the workplace and life quality. Some examples of chores being delegated to robots include performing repetitive labor-intensive chores [Bogue, 2016], taking care of the elder [Tanioka, 2019], or exploring dangerous environments (*e.g.,* outer space [Gao and Chien, 2017], radioactive zones [Nagatani et al., 2013], deep sea [Li et al., 2023a]). Nevertheless, learning in a data-efficient way is critical requirement for robots to be a feasible solution.

By modeling the aspects of the environment the agent perceives, the things it can do, and how its actions change the environment, reinforcement learning (RL) provides a framework to solve sequential decision-making problems. With a weak supervision, in the form of a reward signal, RL agents can learn complex behaviors from their interactions with the environment [Mnih et al., 2015, Andrychowicz et al., 2020]. However, one of the main drawbacks of learning in a trial-and-error way is that it requires a large number of interactions (*i.e.,* data), rendering RL methods too data costly for most robotics applications because of the *curse of real-world samples* [Kober et al., 2013], which refers to the physical wear and tear suffered by robots as a consequence of interacting with the real world.

To extend the applicability of robust RL agents, recent progress in simulators with realistic physics engines have mitigated the data cost of training such agents [Todorov et al., 2012, Makoviychuk et al., 2021]. However, for RL problems that

have not been modeled with a high-precision simulator, transfer learning presents a feasible alternative to reduce training costs through the reuse of knowledge acquired in a different but related problem [Bone, 2008, Taylor and Stone, 2009a, Lazaric, 2012a]. Moreover, if an agent is expected to sequentially solve multiple tasks over its lifetime (*e.g.,* general purpose robots [Zucker et al., 2015]), adopting a lifelong learning approach [Parisi et al., 2019, Khetarpal et al., 2022] allows the agent to exploit previous experiences to solve a new (but related) situation faster, as people seem to do [Carbonell, 1983].

In order to learn faster/better in the current task (compared to learning from scratch), by reusing previously acquired knowledge, the source of knowledge has to be related to the current problem. Depending on how much the environment can change across tasks, being able to *select* and *adapt* related knowledge from the lifelong learning agent's growing knowledge base may become crucial to avoid negative transfer [Taylor and Stone, 2009a], as changes may range from dynamics shifts (*e.g.,* different friction factors between a walking robot and the floor) to major modifications in the agent's observation-action representation (*e.g.,* unexpected limb malfunctions). Furthermore, to perform both processes in a task-diverse setting such as a lifelong learning robot in an uncontrolled environment would require a high degree of adaptability and autonomy.

That is why, in this thesis, a lifelong learning system for a range of tasks that may not share the state-action representation is proposed. The system uses an inter-task similarity function to estimate the *relatedness* of pairs of tasks (based on their dynamics), and select the most related candidate from the set of available learned tasks. Then, a knowledge transfer method adapts the policy, from the selected source, to use it in the target task and speed the learning process up. The proposed system is evaluated in a wide variety of control tasks, showing its ability to deal with sequences of diverse problems, and autonomously make an effective use of its past experiences.

1.1 Justification

The following problems reported in the literature, which are related to transfer learning and lifelong reinforcement learning, will be addressed in this research.

1. **Large Sample Complexity**: This problem refers to the large amount of data samples reinforcement learning agents usually require to learn to solve a task [Taylor and Stone, 2009a]. The goal of transfer learning methods is to decrease the sample complexity by reusing auxiliary knowledge, acquired from a related task.

2. **Negative Transfer**: This phenomenon refers to the decrease in the performance of a learning agent in a target task, as a consequence of using knowledge transferred from a source task [Zhu et al., 2023, Glatt et al., 2016]. The goal is to endow transfer learning systems with the ability to select knowledge sources, from a set of options, that will benefit the learner, or identify that it is a better alternative to learn from scratch.

3. **Catastrophic Forgetting**: This problem refers to the situation in which a lifelong learning agent decreases its performance in old tasks (*i.e.,* forgets) as a consequence of learning to solve new tasks [Robins, 1995].

1.2 Problem Statement

To consecutively learn a finite sequence (of unknown length) of RL tasks with different representations with a lifelong learning approach is difficult for several reasons. The challenges a lifelong RL agent must face in this scenario are the following:

1. The agent must be able to compare tasks, that do not share state-action spaces, in order to identify in which of the previously solved tasks there may be pieces of reusable knowledge.

2. The system must transfer knowledge in a way that improves the performance of the learning agent, whether it is by requiring fewer interactions with the environment, or achieving a larger accumulated reward.

3. Knowledge of previous tasks should not only be used to speed up the process of learning of new tasks, but also to solve the tasks from which it was learned. Thus, the agent should avoid forgetting how to solve previous tasks after new tasks are learned.

Formally: Let a task $T_i = \langle S_i, A_i, E_i, t_i \rangle$ be defined by a pair of (finite or infinite) sets S_i and A_i that contain the states and actions the agent can adopt and perform, a function $E_i : S_i \times A_i \rightarrow S_i \times \mathbb{R}$ that models the dynamics of the environment, and a threshold performance $t_i \in \mathbb{R}$ that indicates when a task has been learned. Also, let $(T_1, ..., T_i, ..., T_N)$ be a finite sequence of cross-domain RL tasks (*i.e.*,, if $i \neq j$ then $S_i \neq S_j$ and $A_i \neq A_j$), D_L be a lifelong RL agent, D_R be a regular RL agent, and $P_i(D_L, T_j)$ be the performance of agent D_L on task T_j after D_L learned to solve task T_i, where $j < i \leq N$. Hence, this research will address the problem of a lifelong cross-domain RL agent D_L transferring knowledge between $\{T_1, ..., T_{i-1}\}$ and T_i, such that $\forall i \in [1, N]$:

- D_L learns task T_i at least as fast (requires fewer data queries of E_i to achieve a performance equal or higher than t_i) or as good (achieves a larger asymptotic performance) as D_R
- $\forall j \in [1, i-1]$, $P_i(D_L, T_j) = P_{i-1}(D_L, T_j)$

1.3 Research Questions

The main questions of this research are the following:

1. How can similarity between tasks that do not share state-action spaces be measured, in a way that helps determining if positive knowledge transfer can be performed between them?
2. What form of knowledge and how can it be transferred between tasks that do not share state-action spaces in a way that produces a positive knowledge transfer?

1.4 Hypothesis

Given a finite sequence of reinforcement learning tasks and a lifelong reinforcement learning agent, by measuring inter-task similarity between tasks that do not share state-action spaces it is possible to learn tasks faster, than learning from scratch, by transferring knowledge from previously related tasks have been learned.

1.5 General Objective

To design and develop a transfer learning methodology, for a lifelong cross-domain reinforcement learning agent, that is capable of performing positive knowledge transfer across tasks that do not share state-action spaces.

1.5.1 Specific Objectives

1. To design and develop an inter-task similarity measure that can be applied to heterogeneous RL tasks, such that selects source tasks that produce positive knowledge transfer.

2. To design and develop a transfer learning algorithm for heterogeneous RL tasks that learns *better* than a scratch learner: requires less data to achieve a threshold performance, or achieves a larger asymptotic performance.

3. To integrate the similarity measure and knowledge transfer method to develop a lifelong reinforcement learning algorithm that can be evaluated in the cross-domain setting, such that the agent: learns *better* than a scratch learner and does not *forget* how to solve previous tasks.

1.6 Scope and Limitations

- **Lifelong Setting**: The agent does not have knowledge nor control over the order in which the sequence of tasks is presented to it.

- **Unlabeled Tasks**: The agent can not identify tasks.

- **Episodic Tasks**: Tasks have a maximum number of steps that can be taken before the episode ends and the environment state resets to an initial state.

- **Fully Observable State**: Observations completely describe the environment's state, thus, $observation = state$.

- **Stationary Dynamics**: Reward and transition models do not change over time.

- **Continuous and Discrete Spaces**: State and action spaces can consist of the cross product of a set of continuous variables or finite countable sets.
- **Bounded Actions and Observations**: Each action, state, and reward variable is bounded, thus their lower and upper limits are known *a priori* for all tasks.
- **Dense/Rich Rewards**: tasks with reward models that are a dense function of their state-action space, where a dense reward function provides feedback at every step of the agent's training process [Zhong Hong, 2024].
- **Non-image Representations**: Reinforcement learning tasks whose observations are images are out of the scope of this research.

1.7 Contributions

The main contributions made in this research are following:

1. A systematic review of knowledge transfer methods focused on the cross-domain reinforcement learning setting.
2. A model-based similarity measure for source selection purposes in cross-domain reinforcement learning.
3. A knowledge transfer method for cross-domain reinforcement learning.
4. A lifelong cross-domain reinforcement learning system.

In this document we present two similarity functions and knowledge transfer methods, for tasks with discrete and continuous state-action spaces in Chapter 4 and Chapter 5, respectively. For the discrete-space case, the inter-task similarity is defined over the action-value functions, and transfer action values to accelerate the target agent's learning, whereas in the continuous-space scenario tasks are matched via an inter-task mapping that allows comparing states and actions from different domains, as well as to transfer policies. Furthermore, for the lifelong learning problem (see Chapter 6), the transfer method for tasks with continuous spaces is adapted to address this sequential setting.

1.8 Publications

As a product of this research, two conference articles have been accepted for publication and presented, and two journal articles are currently being reviewed for publication:

- *Knowledge Transfer for Cross-Domain Reinforcement Learning: A Systematic Review* [Serrano et al., 2024]: a journal article has been accepted for publication in IEEE Access. This article covers a wide variety of knowledge transfer methods for the cross-domain setting, which is an extended revision of the one presented in Section 3.2.
- *Similarity-based Knowledge Transfer for Cross-Domain Reinforcement Learning* [Serrano et al., 2023]: a journal article submitted to Machine Learning. This article presents the similarity measure and knowledge transfer results, summarized in Chapter 5. The article was initially submitted in December, 2023 and is currently at a 2nd review stage.
- *Inter-Task Similarity Measure for Heterogeneous Tasks* [Serrano et al., 2021a]: a regular paper in the 24th RoboCup International Symposium 2021.
- *Inter-Task Similarity for Lifelong Reinforcement Learning in Heterogeneous Tasks* [Serrano, 2021]: an extended abstract in the International Joint Conference on Artificial Intelligence (IJCAI) 2021 Doctoral Consortium.

1.9 Thesis Outline

The remaining of this document is structured as follows: Chapter 2 presents the main concepts of reinforcement learning, lifelong learning and similarity functions, Chapter 3 covers the works that address problems that are closely related to lifelong cross-domain reinforcement learning. In Chapter 4 a transfer knowledge method based on a performance-based similarity measure is introduced, while Chapter 5 presents a similar method that employs a model-based similarity function [García et al., 2022], both evaluated in the cross-domain setting. Chapter 6 introduces a lifelong agent based on the developments presented in Chapter 5. Finally, Chapter

7 presents the final remarks of this research, as well as ideas of future directions to explore.

Chapter 2

Theoretical Framework

In this chapter, background theory relevant to this research is introduced. First, basic definitions of reinforcement learning and Markov decision processes are covered, followed by concepts on transfer learning, lifelong learning and similarity functions.

2.1 Reinforcement Learning

Reinforcement Learning (RL) is a subarea of machine learning that addresses sequential decision-making problems. The main objective of an RL algorithm is to learn a *satisfactory* behavior, through a trial-and-error interaction of the learning agent with the environment (see Fig. 2.1). According to [Sutton and Barto, 2018], RL is a machine learning paradigm on its own since it holds important differences with supervised learning and unsupervised learning.

- Supervised Learning: The main difference with RL lies in the source of knowledge from which the system learns. In supervised learning, an external entity (*e.g.*, a machine learning engineer) provides pairs of scenario examples (*i.e.,* input) and the correct action to perform in that situation (*i.e.,* label). Conversely, an RL agent learns from its own experience, as it interacts with the environment.

- Unsupervised Learning: Although unsupervised learning and RL have in common that labeled data is not required, the main difference between them is the

learning objective. While in unsupervised learning the system strives to find structure in a collection of unlabeled data, an RL system will try to maximize a reward signal.

Additionally, the RL problem is formalized as finding the optimal control (or policy) for a partially-known Markov Decision Process. That is, the goal is to find/learn a policy (*i.e.*, a function that maps observations to actions) whose actions maximize the expected return (see Eq. 2.1) in the case of tasks of finite duration (*i.e.*, episodic tasks), or the expected discounted return (Eq. 2.2) for tasks that go on continually without a time limit (*i.e.*, infinite-horizon problems):

$$G = \sum_{t=0}^{T} R_t \tag{2.1}$$

$$G = \sum_{t=0}^{\infty} \gamma^t R_{t+1} \tag{2.2}$$

where R_t is the reward observed by the agent at time t, T is the duration of the episode and $\gamma \in [0, 1)$ is a discount factor, that weighs the present value of future rewards (see Section 2.2 for more details) [Sutton and Barto, 2018]. Thus, any technique that is capable of solving this class of problem is considered to be an RL method.

2.2 Markov Decision Processes

A Markov Decision Process (MDP) is a mathematical framework used to model sequential decision-making problems for dynamic systems, *i.e.*, systems in which the state changes over time [Puterman, 2014]. An MDP is formally specified by a tuple $\langle S, A, \Phi, R, \gamma \rangle$, where

- S: Set of states in which the agent can be found.
- A: Set of actions the agent can perform.
- Φ: The transition function $\Phi : S \times A \times S \to [0, 1]$ specifies a probability distribution for every state-action pair (s, a), such that $\Phi(s, a, s')$ is the proba-

bility of the agent transiting to s' after executing action a from state s, where $s, s' \in S$ and $a \in A$.

- R: The reward function $R : S \times A \rightarrow \mathbb{R}$ specifies a scalar value for every state-action pair (s, a), such that $R(s, a)$ is the immediate reward signal the agent will perceive after executing action a in state s, where $s \in S$ and $a \in A$.

- γ: The discount factor $\gamma \in [0, 1)$ weights the present value of future rewards.

The purpose of specifying an MDP is to formally describe the setting in which an agent can interact with the environment. The transition function (Φ) represents the stochastic effects of the actions in the state of the system, whereas the reward function (R) represents (in an implicit form) the desired behavior for the agent. That is, in the reward function one should assign positive large values to pairs (s, a) such that executing a from s is desirable. On the other hand, one should penalize undesirable state-actions pairs by assigning large negative values.

Moreover, the MDP is a particular instance of the more general partially observable Markov decision process (POMDP) setting, in which the agent updates a state belief, based on actions and observations (see Fig. 2.1). Defined by a tuple $\langle S, A, \Phi, R, \gamma, O, \Omega, B_0 \rangle$ where $o \in O$ is the set of observations the agent can perceive, $\Omega : S \times A \times O \rightarrow [0, 1]$ the observation function that models the probability of perceiving an observation given an action and the current state, and $B_0 : S \rightarrow [0, 1]$ the initial state belief [Kaelbling et al., 1998, Serrano et al., 2021b].

2.2.1 Policy

In the context of MDPs, a policy $\pi : S \rightarrow A$ is a function that returns an action a for every state s. In order to solve a decision-making problem, actions must be carefully selected, in such way that the criterion of what is considered a desirable behavior is fulfilled, as shown in Fig. 2.1.

Thus, an optimal policy π^* (Eq. 2.5) will select, for every state, the action that maximizes the expected reward, *i.e.*, the best possible action. The expected reward (weighted by discount factor γ, where $0 \leq \gamma < 1$) of an agent that follows actions drawn by a policy π is represented by the state value function (Eq. 2.3),

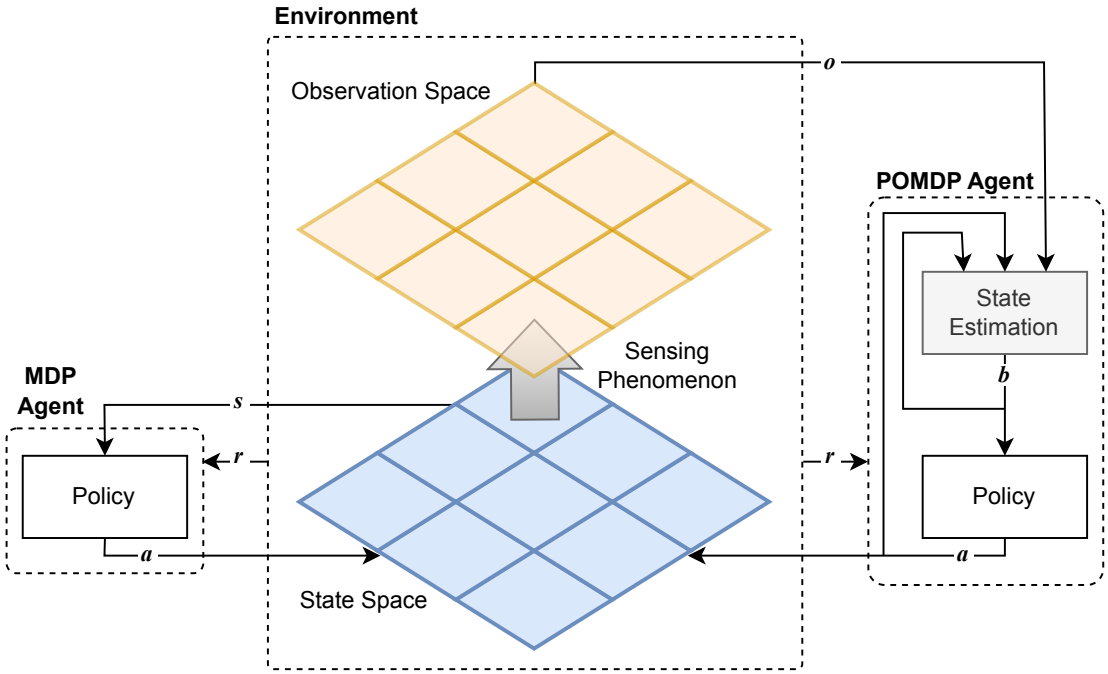


Figure 2.1: In an MDP setting, every time the agent executes an action (***a***) it will receive in return a reward signal (***r***) associated to the performed action and the current state (***s***) of the world (also called system or environment). Then, the agent consults its policy to perform the best action given the new state of the world.

which can be optimized (see Eq. 2.4) to learn the optimal policy π^* [Watkins and Dayan, 1992, Bellman and Dreyfus, 2015].

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} \Phi(s, \pi(s), s') V^{\pi}(s') \tag{2.3}$$

$$V^{\pi^*}(s) = max_a \left\{ R(s, a) + \gamma \sum_{s' \in S} \Phi(s, a, s') V^{\pi^*}(s') \right\} \tag{2.4}$$

$$\pi^*(s) = argmax_a \left\{ R(s, a) + \gamma \sum_{s \in S} \Phi(s, a, s') V^{\pi^*}(s') \right\} \tag{2.5}$$

In order to compute π^* for a fully specified MDP, an iterative algorithm is capable of finding such policy. For instance, value iteration is an algorithm that starts by assigning every state with a value of 0, and through an iterative process the value of every state is updated using Eq. 2.6. The algorithm stops once $|V_t(s) - V_{t-1}(s)| < \epsilon$

is satisfied for every state in S, given a margin of error ϵ [Sucar, 2015].

$$V_t(s) = max_a \left\{ R(s,a) + \gamma \sum_{s' \in S} \Phi(s,a,s') V_{t-1}(s') \right\} \tag{2.6}$$

Alternatively, some algorithms employ the action value function $Q(s,a)$ (see Eq. 2.7), which represents the expected discounted reward for taking action a in state s and then follow a policy π [Watkins and Dayan, 1992]:

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} \Phi(s,\pi(s),s') V^{\pi}(s') \tag{2.7}$$

2.2.2 Learning Approaches

The way an MDP policy can be computed greatly depends on how much information about the environment is available. That is, the more information the agent has, the easier it will be to compute a good policy. According to [Taylor and Stone, 2009a], there are many ways in which a policy can be obtained, from which we present three of the most common approaches:

1. **Dynamic programming**: Value iteration and policy iteration are examples of algorithms of this type of approach, in which the algorithm assumes that a fully specified MDP is available (*i.e.*, $\langle S, A, \Phi, R \rangle$). Therefore, as the system has a full model of the environment (which is also assumed to be correct), no interaction is required.

2. **Model-based**: These methods relax the assumption dynamic programming approaches make on the availability of the full MDP, as they interact with the environment in order to estimate an approximated model of it, *i.e.*, Φ and R. This approach is mostly used in environments where the model is stationary but unknown, hence, once the model is estimated it can be reused.

3. **Temporal difference (TD)**: Methods that learn an action-value function $Q : S \times A \rightarrow \mathbb{R}$ by backing up all the rewards that have been perceived through time. For a pair (s,a), $Q(s,a)$ represents the expected return when a is executed from s. At any time, the best current policy is equivalent to select the

highest valued action form the current state, *i.e.*, $\underset{a}{\text{argmax}}Q(s,a)$. Q-learning [Watkins and Dayan, 1992] and Sarsa [Rummery and Niranjan, 1994] are examples of TD algorithms.

4. **Monte Carlo methods**: Contrary to dynamic programming methods, that learn a value function or policy from a completely described MDP, Monte Carlo methods learn from experiences. Monte Carlo methods use sequences of states, actions and rewards from real or simulated interactions. After sampling multiple sequences of interactions by a policy π_t, the cumulative discounted rewards are averaged over the sequences to estimate the expected return, and approximate the state value function $V_t(s)$ (*i.e.*, Policy Evaluation). Then, the policy is updated to π_{t+1} by greedily selecting actions from $V_t(s)$ (*i.e.*, Policy Improvement). The policy evaluation and improvement steps are alternated until the policy reaches an optimal performance [Sutton and Barto, 2018].

Additionally, according to [Li, 2017], deep RL results from using a deep neural network (*i.e.*, a neural network with two or more layers) to approximate an RL component, such as a value function, a policy, a transition model or a reward model. The parameters that describe these components consist of the weight and bias values in the neural network. Deep Q-Network [Mnih et al., 2015] and A3C [Mnih et al., 2016] are some examples of deep RL algorithms.

2.3 Latent Spaces

A latent space (also known as hidden variable space) is a multidimensional space defined by a set of variables that do not (necessarily) represent real world phenomena, and are used to model important features of input data for some particular process. Latent spaces can be used to model the probability density of some phenomenon with less independent parameters compared to using observable variables only, which requires fewer data [Bishop, 1998]. Similarly, whether it is by finding a more discriminant space with principal component analysis (PCA), or with convolutional neural networks ((CNN)) [Guo et al., 2017], latent spaces can be a powerful tool for enhancing pattern recognition or classification in high-dimensional data spaces.

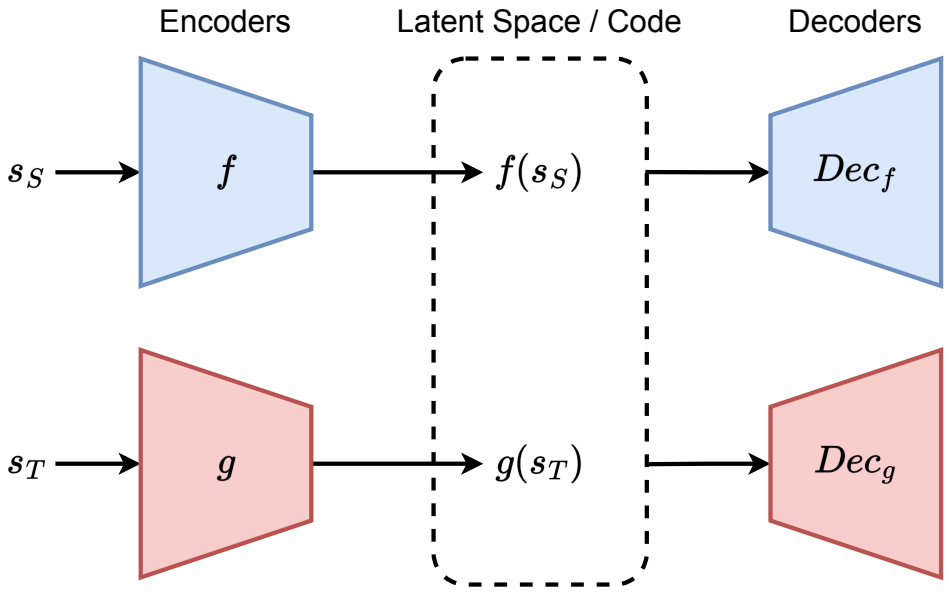


Figure 2.2: Pair of autoencoders setting used in [Gupta et al., 2017]. The encoders f, g map states s_S, s_T from a source and target state space, respectively, to a latent space in which the image of states can be compared to each other. The decoders map any data point from the latent space to their respective state space. That is, with the following function compositions states can be mapped across state spaces: $Dec_g \circ f(s_S)$ and $Dec_f \circ g(s_T)$.

On the other hand, latent spaces have been recently used in RL for knowledge transfer applications, particularly for cross-domain settings (*i.e.*, where the source and target task have different state/action representations). In this scenario, latent spaces are learned (usually with autoencoder settings [Tschannen et al., 2018]) to bridge the state and/or action spaces of the two tasks, while satisfying some criterion. For instance, [Gupta et al., 2017] train a pair of autoencoders so that the pairs states from different state spaces (see Section 2.2 for a definition of state space) that are known to be similar to each other are mapped near to each other in the latent space.

2.4 Transfer Learning

The purpose of transfer learning (TL) is to improve the performance of machine learning methods by means of transferring knowledge between tasks [Lazaric, 2012b]. Furthermore, according to [Taylor and Stone, 2009a], the main insight that motivates the use of TL methods is that knowledge generalization might occur across tasks, and not only within them. Although several taxonomies have been proposed for TL

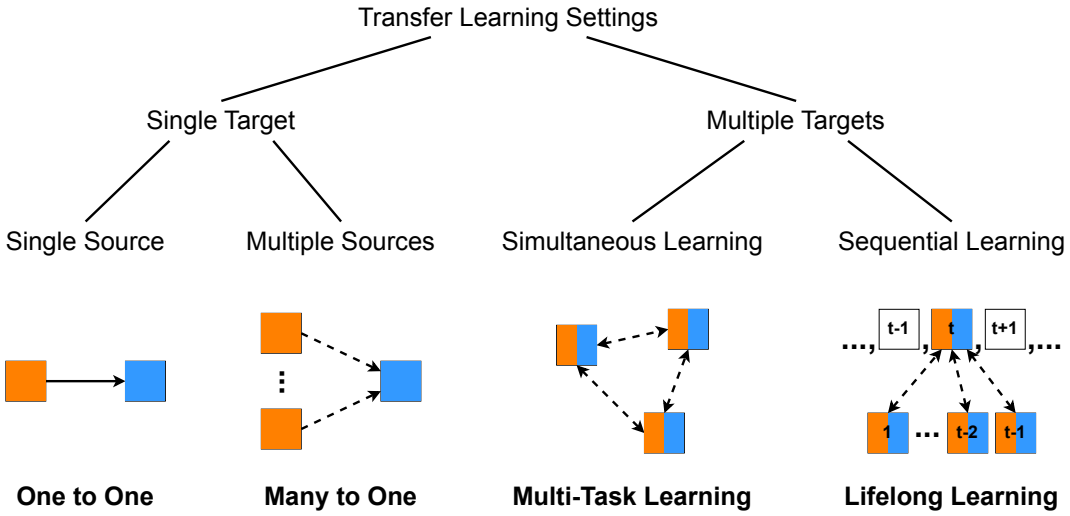


Figure 2.3: Classification of transfer learning settings based on the availability of source (orange) and target (blue) tasks at the moment knowledge needs to be transferred, as well on the order in which tasks must be learned. Solid arrows indicate the direction in which knowledge is transferred, whereas dashed arrows indicate that it is up to the TL system to decide whether to transfer knowledge in that direction. In the multiple target setting, tasks can behave both as a source and target task.

methods (*e.g.,* based on the availability of labeled data [Pan and Yang, 2009] or the goal in a multi-agent setting [Da Silva and Costa, 2019]), we present a taxonomy based on the availability of tasks and the order in which they must be learned (see Fig. 2.3). Additionally, the performance of a TL method applied to RL tasks can be measured in multiple ways (see Fig. 2.4).

- Jumpstart: The performance of a learning agent at the beginning of the training process. This metric can potentially be improved if knowledge is transferred from a source task before the training begins in the target task.

- Asymptotic Performance: The final performance of a learning agent.

- Total Reward: The total reward accumulated by the agent. Visually, the area under the learning curve represents this metric.

- Transfer Ratio: This metric is equivalent to the ratio of the total reward accumulated by the transfer learning method and the total reward accumulated by the non-transfer learner.

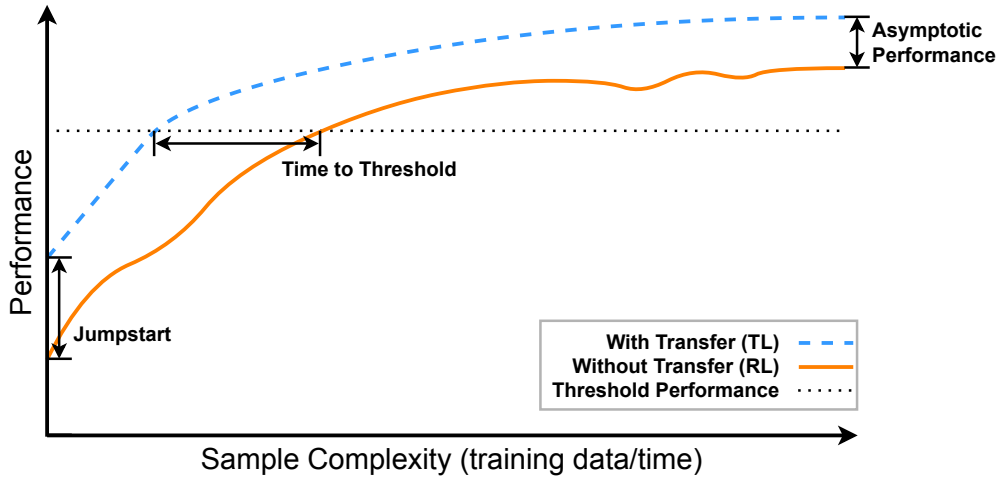


Figure 2.4: Example of improvement in the jumpstart, asymptotic performance, total reward (*i.e.,* the area under the performance curve) and time to threshold as a result of transferring knowledge.

- Time to Threshold: The time required by a learning agent to achieve a pre-specified performance level.

Hence, TL methods can help an RL agent to reduce the time to the threshold (*learn faster*) and to raise its total reward and asymptotic performance (*learn better*). In RL, the learning time is known as sample complexity, which is the amount of data required by an algorithm to learn. When a TL method harms the performance of a learning agent in a target task (*i.e.,* increases sample complexity or decreases the jumpstart, asymptotic performance, or total reward) is called negative transfer, whereas if an improvement is achieved then it is called positive transfer [Taylor and Stone, 2009a].

2.4.1 Cross-Domain Transfer Reinforcement Learning

In RL, there is a particular form of the TL problem in which state and/or action spaces may be different, called cross-domain transfer reinforcement learning [Ammar et al., 2015b]. One of the main challenges of cross-domain TRL is that states/actions from different spaces are not directly comparable. Thus, there are certain strategies that cross-domain TRL methods adopt to overcome the representation mismatch challenge, which include using inter-task mappings, latent spaces (see Section 2.3)

and similarity functions.

- **Inter-Task Mapping**: An inter-task mapping is a function that maps states/actions from one task's state/action space to the other [Taylor and Stone, 2009b]. This mapping allows transferring knowledge in the form of (*state, action, next state*) tuples [Ammar et al., 2015b], or to compare the transition/reward functions of tasks defined over different spaces [Serrano et al., 2023].
- **Latent Space**: Latent spaces are often used as a common representational space to which states/actions from different tasks are mapped. Although latent spaces can be used as midpoint in the process of mapping elements across tasks [Serrano et al., 2023], they are also employed to compare how similar/different states/actions are to bias the behavior of a learning agent through reward shaping [Gupta et al., 2017, Hu and Montana, 2019].
- **Similarity Function**: Instead of explicitly learning inter-task mappings, or latent spaces, to compare states/actions from different tasks, some works employ functions that map task-dependent elements to task-invariant features. For instance, comparing the action value functions [Serrano et al., 2021a], or the state-visitation distributions of two tasks [Fickinger et al., 2021].

2.5 Lifelong Reinforcement Learning

Lifelong Learning (LL) is a continual learning process in which, at any time, the learning system has learned to perform a sequence of N tasks (tasks can be from different domains). When the $(N+1)$-th task is encountered, the learning model can exploit past experiences (stored in its knowledge base) to aid the learning process in the current task [Chen and Liu, 2018] (see Fig. 2.5).

In the context of RL, tasks are represented by MDPs, while the learning system can be modeled in a variety of ways. According to [Mendez and Eaton, 2020], lifelong RL systems can be classified in two categories, depending on how parameters are shared among the learned tasks:

- Single-model: In this approach, a single set of parameters is responsible for storing the knowledge acquired by the entire sequence of tasks. Such methods

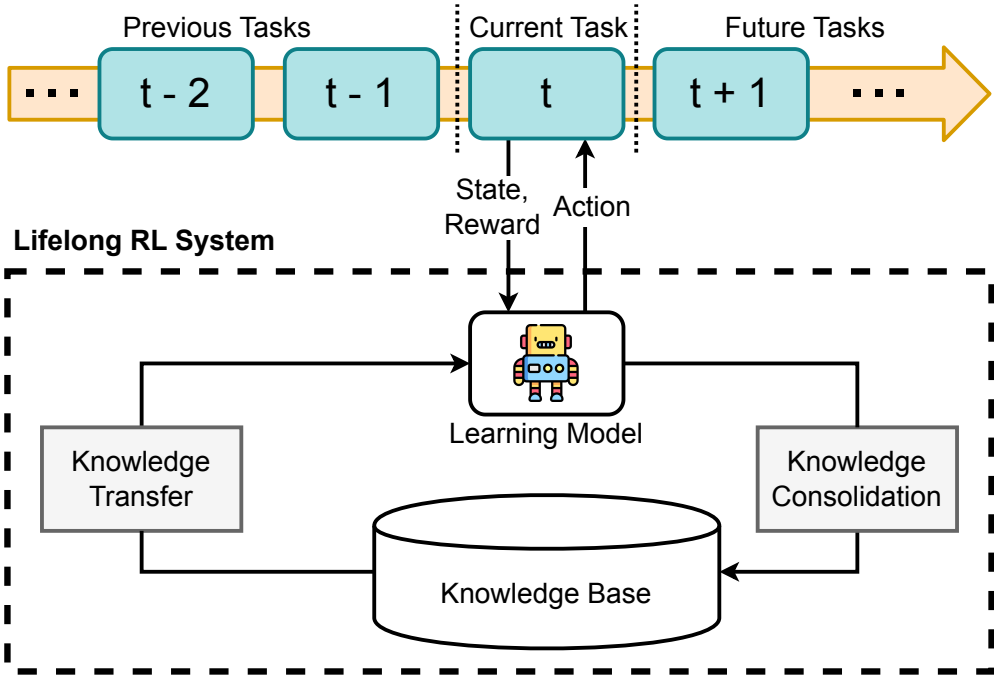


Figure 2.5: Lifelong reinforcement learning (LRL) system. The system learns to solve tasks sequentially. When it encounters with a new task, the learning model learns by interacting with the environment and using auxiliary knowledge acquired in previous tasks (*i.e.,* Knowledge Transfer). After a task is learned, the recently acquired information is stored in the knowledge base (*i.e.*, Knowledge Consolidation), so that the agent can reuse it to learn future tasks in a sample efficient way.

are usually employed when on of the following assumption is true: the sequence of tasks are similar to each other, or the model is over-parameterized which allows it to learn the diversity present among the sequence of tasks.

- Multi-model: The multi-model approach consists of a learning system constituted by a set of parameters that are shared among all tasks, and a collection of task-specific parameters. The set of shared parameters enables transferring knowledge across tasks, while the task-specific parameters are responsible for capturing the particularities tasks may have.

2.6 Distance and Similarity Functions

According to [Tversky, 1977], the concepts of similarity and distance are important for artificial intelligence in general as they provide a way to organize, classify, and

generalize over some class of objects. Despite distance and similarity functions being widely explored for propositional representations (*i.e.,* feature vectors), they can also be employed to assess the similarity of objects from a graph-based representation (*e.g.,* an MDP).

Distance and similarity functions can be seen as a complement to each other. Distance functions assign larger values to pairs of objects that are *more dissimilar*, whereas similarity functions associate larger values to pairs of objects that are *more similar*, or *closer*, with respect to some criterion. Definitions for distance metric and similarity function [Ontañón, 2020] are provided below.

Definition 1 (distance metric) A *distance metric* d over objects in a set X is a function $d : X \times X \rightarrow [0, \infty)$ such that, for each $x, y, z \in X$ the following properties are satisfied:

- $d(x, y) \geq 0$ (Non-negativity)
- $d(x, y) = 0 \iff x = y$ (Identity)
- $d(x, y) = d(y, x)$ (Symmetry)
- $d(x, z) \leq d(x, y) + d(y, z)$ (Triangle inequality)

Definition 2 (similarity function) A *similarity function* s over objects in a set X is a function $s : X \times X \rightarrow [0, u]$, where u is an upper bound, and where for each $x, y \in X$ the following properties are satisfied:

- $s(x, y) \geq 0$ (Non-negativity)
- $s(x, y) \leq u$ (Boundedness)
- $s(x, y) = u \iff x = y$ (Identity)
- $s(x, y) = s(y, x)$ (Symmetry)

2.6.1 Pseudometric and Semi-metric

A pseudometric space and semi-metric space are generalizations of a metric space (*i.e.*, a set and metric distance tuple) in which not every property of a metric space is satisfied. In a pseudometric space, the distance between different elements can be zero (*i.e.*, the identity property is not satisfied as elements are not necessarily distinguishable) [Howes, 2012]. On the other hand, in a semi-metric space, the triangle inequality property is not satisfied [Wilson, 1931]. Formally, pseudometrics and semi-metrics are defined in Definition 3 and Definition 4, respectively.

Definition 3 (Pseudometric) A *distance pseudometric* d over objects in a set X is a function $d : X \times X \to [0, \infty)$ such that, for each $x, y, z \in X$ the following properties are satisfied:

- $d(x, y) \geq 0$ (Non-negativity)
- $d(x, y) = d(y, x)$ (Symmetry)
- $d(x, z) \leq d(x, y) + d(y, z)$ (Triangle inequality)

Definition 4 (Semi-metric) A *distance semi-metric* d over objects in a set X is a function $d : X \times X \to [0, \infty)$ such that, for each $x, y \in X$ the following properties are satisfied:

- $d(x, y) \geq 0$ (Non-negativity)
- $d(x, y) = 0 \iff x = y$ (Identity)
- $d(x, y) = d(y, x)$ (Symmetry)

2.6.2 Kantorovich Metric

The Kantorovich metric allows measuring the distance between two distributions, given that there exists a *ground distance* $d : S \times S \to \mathbb{R}$ between the elements that constitute the support S of the distributions. For instance, if $S = \{0, 5, 10\}$ is the set of possible grades a robot chef can receive in a pie contest, then $d(x, y) =$

$|x - y| \forall x, y \in S$ could be used to measure the distance between two pie-grade probability distributions, allowing us to evaluate how far two robots are in terms of their culinary abilities [Deng and Du, 2009].

Roughly speaking, for two discrete probability distributions $\mathbb{P}$ and $\mathbb{Q}$ (with finite supports $\{x_0, ..., x_m\}$ and $\{y_0, ..., y_n\}$ respectively) minimizing the total cost in the discrete transportation problem (see Eq. 2.8) is equivalent to finding the distance between $\mathbb{P}$ and $\mathbb{Q}$:

$$\begin{aligned} \text{Minimize } & \sum_{i=0}^{m}\sum_{j=0}^{n} \mu(x_i, y_j) d(x_i, y_j) \\ \text{Subject to } & \forall 0 \leq i \leq m : \sum_{j=0}^{n} \mu(x_i, y_j) = \mathbb{P}(x_i) \\ & \forall 0 \leq j \leq n : \sum_{i=0}^{m} \mu(x_i, y_j) = \mathbb{Q}(y_j) \\ & \forall 0 \leq i \leq m, 0 \leq j \leq n : \mu(x_i, y_j) \geq 0 \end{aligned} \tag{2.8}$$

where $d(x_i, y_j)$ is the ground distance between x_i and y_j, $\mathbb{P}(x_i), \mathbb{Q}(y_j)$ are the probabilities of x_i and y_j (respectively), and $\sum_{i=0}^{m}\sum_{j=0}^{n} \mu(x_i, y_j)$ is the distance between probability distributions $\mathbb{P}$ and $\mathbb{Q}$ (after the optimization problem in Eq. 2.8 is solved).

2.6.3 Hausdorff Distance

Given that (X, d) is a metric space, the Hausdorff Distance provides a way to measure the distance between every compact subset of X (*i.e.*, subsets that are closed and bounded) [Rockafellar and Wets, 2009]. To measure the distance between a pair of finite sets of points $X = \{x_0, ..., x_m\}, Y = \{y_0, ..., y_n\}$, the Hausdorff distance is defined by Eq. 2.9:

$$H(X, Y) = max(h(X, Y),\ h(Y, X)) \tag{2.9}$$

$$h(X, Y) = \max_{x \in X} \min_{y \in Y} \|x - y\| \tag{2.10}$$

where $\|\cdot\|$ is a norm for the points in X and Y (*e.g.*, the Euclidean norm), and $h(X, Y)$ (Eq. 2.10) is called the directed Hausdorff distance from X to Y [Huttenlocher et al., 1993]. Since $h(X, Y)$ returns the largest distance between a point in X and its nearest neighbor in Y, if $h(X, Y) = d$ then every point in X has a point from Y within a radius of d. Additionally, given that the Hausdorff distance returns the largest of the directed Hausdorff distances (computed in both directions), then the Hausdorff provides the radius within which every point has a neighbor point from the other set.

2.7 Chapter Summary

In this chapter, the basic theory on reinforcement learning, transfer learning, lifelong learning, distance/similarity functions and latent spaces has been presented. MDPs provide a basic framework to model sequential decision-making problems, upon which transfer methods (*e.g.,* lifelong learning) can built on to learn near optimal policies with a lower sample complexity. Additionally, basic definitions for distance and similarity functions and latent spaces have been presented, because a similarity-based approach is proposed to avoid negative transfer in the lifelong RL setting, through the alignment of different state-action spaces in a latent representation.

Chapter 3

Related Work

In order to reuse knowledge from previously learned tasks, the proposed lifelong learning method employs a cross-domain similarity measure to select the source task from which knowledge will be transferred. This chapter presents a literature review of works concerned with measuring similarity in MDPs (Section 3.1), transferring knowledge across different domains (Section 3.2) and learning a sequence of tasks (Section 3.3).

3.1 Similarity Measures

The ability to measure how similar two situations are is crucial for the generalization of knowledge. In the context of sequential decision-making problems (modeled as an MDP), similarity measures can be categorized in two classes: model-based and performance-based methods [García et al., 2022]. Model-based similarity measure compare elements related to the description of the MDP, *i.e.,* the reward and/or transition functions. For instance, [Ammar et al., 2014b] train a Restricted Boltzmann Machine [Sutskever et al., 2009] to approximate the transition function of a task, and then use it to see how well it predicts state transitions sampled from another MDP. The smaller the error, the more similar a pair of tasks are. Similarly, [Narayan and Leong, 2019] assess similarity between state-action pairs by comparing their state-transition distributions with the Jensen-Shannon distance [Grosse et al., 2002, Nielsen, 2019]. Similarly, [Tao et al., 2021] and [Gleave et al., 2020] propose

to evaluate the inter-task similarity based on the reward function. While [Tao et al., 2021] assume reward functions are modeled as a linear combination of shared features and estimate similarity as the cosine distance between the weight vectors that define each reward function, [Gleave et al., 2020] define a pseudometric over reward functions, which they use to bound the regret of optimal policies over changes in the initial state distribution and state transition distribution. On the other hand, [Carroll and Seppi, 2005] define the inter-task similarity based on the immediate reward of state-action pairs.

In light that both the transition and reward functions hold a great amount of information about a task, some works include both aspects in their MDP similarity definition. In [Ferns et al., 2004, Ferns et al., 2012] an equivalence class, called bisimulation, is defined over the state space to group states for abstraction purposes, based on two semimetrics that measure the transition distributions with the Kantorovich metric [Deng and Du, 2009] and the total variation distance [Gibbs and Su, 2002], respectively. In similar fashion, [Wang et al., 2019] propose to measure state and action similarity, within an MDP, using a graph representation that captures the behavior of states and actions based on their reward and transition functions.

In [Song et al., 2016], the bisimulation class is extended to composite the state-similarity of two MDPs, with the Kantorovich and Hausdorff metrics [Henrikson, 1999], to measure MDP similarity. Similarly, [Sorg and Singh, 2009] expand on the equivalence class of MDP homomorphisms [Ravindran, 2004] by defining a soft-version of the equivalence class that can be approximated with a data set, instead of requiring complete knowledge of the MDP. The soft MDP homomorphism allows learning an inter-MDP mapping that ideally preserves the algebraic structure defined by the transition and reward functions. Additionally, [Castro et al., 2021] extend the application of state-similarity methods to large-scale domains (*e.g.*, deep RL) by providing a distance that is significantly less complex to compute than bisimulation [Ferns et al., 2004, Ferns et al., 2012], which aids learning faster in complex tasks such as the Arcade Learning Environment [Bellemare et al., 2013].

On the other hand, performance-based methods assess the inter-similarity by comparing the performance of policies/value functions in their respective tasks (*i.e.,* policy similarity), as well as the effect of transferring knowledge across tasks (*i.e.,* transfer gain). For instance, in both [Carroll and Seppi, 2005] and [Zhang et al., 2024] compare partially learned value functions to determine the similarity between

a pair of tasks, which provides a simple comparison method that can be directly applied in any value-based RL setting. Other works, such as [Talvitie and Singh, 2007] and [Heng et al., 2022] compute similarity by testing a policy in the other task, and use the total reward as a similarity score, where the more similar two tasks are, the better the expected performance of a policy in the other task is.

The way similarity measures assess the overlap between MDPs address different settings. However, similarity measures can be sorted based on the aspects considered to estimate the inter-task similarity. Reward-based similarity measures [Carroll and Seppi, 2005, Gleave et al., 2020, Tao et al., 2021] compare tasks grounded over environments with identical dynamics but different criteria of what an optimal behavior consists of. On the other hand, similarity measures that only compared state-transition functions [Ammar et al., 2014b, Narayan and Leong, 2019] can be more flexible, as they measure the differences between two environment dynamics with identical reward functions. This type of works are helpful to evaluate the effect of environmental factors that are out of the control of the MDP designer (*e.g.*, a mobile robot learning to navigate in different terrains). However, works that take into account both the reward and transition functions in the similarity assessment offer the greatest flexibility, whether it is by comparing trained policies [Talvitie and Singh, 2007, Heng et al., 2022], value functions [Carroll and Seppi, 2005, Zhang et al., 2024] or compositions of behavior-based state similarity scores [Sorg and Singh, 2009, Song et al., 2016, Castro et al., 2021].

In a setting with multiple sources of knowledge available, such as lifelong learning, being able to effectively select/combine the knowledge that will be transferred is essential to producing positive transfer. This section covered works that perform this selection in a variety of ways, from which only a few can be used in the cross-domain setting with some considerations [Carroll and Seppi, 2005, Talvitie and Singh, 2007, Heng et al., 2022, Zhang et al., 2024]. In [Carroll and Seppi, 2005] compare the immediate reward of state-action pairs, while differences in the transition functions are not measured. In the case of [Talvitie and Singh, 2007] and [Zhang et al., 2024], an inter-task mapping across state spaces is assumed to be available, whereas [Heng et al., 2022] assume some high-level commonalities between tasks (*e.g.,* robots with similar morphology). In contrast, both of the similarity measures presented in this document, in Chapter 4 and Chapter 5, learn the inter-task mappings that match the state-action spaces, and are evaluated in a diverse set tasks.

3.2 Cross-Domain Knowledge Transfer

In contrast to transferring knowledge among task from the same domain, in the cross-domain setting it is necessary to overcome the representation mismatch, either by transferring knowledge that is independent to domains (*e.g.,* weights of a neural network), or by mapping domain-dependent information to the state-action space of the target task (*e.g.,* state-transition data sets, policies, value functions). Regarding the origin of the transferred knowledge, some works focus on exploiting the availability of a task expert to learn by imitation in a different domain. For instance, [Kim et al., 2020] learn an inter-task mapping to transfer policies and state-action demonstrations to the target task, while [Fickinger et al., 2021] use a single state-action demonstration to guide the learning process of a target policy through reward shaping [Ng et al., 1999]. In contrast, other works have developed methods to learn from expert demonstrations, from another domain, that contain only states [Raychaudhuri et al., 2021, Franzmeyer et al., 2022, Zakka et al., 2022, Salhotra et al., 2023, Li et al., 2023b, Li et al., 2023c]. Although a more challenging setting, the state-only methods offer greater flexibility, as specifying what actions caused the state transitions is no longer necessary.

In addition to expert demonstrations, valuable information can also be found in knowledge acquired by other learning algorithms. Such knowledge can take the form of demonstrations generated by an expert policy [Ammar et al., 2012, Ammar et al., 2015b, Shankar et al., 2022, Aktas et al., 2023, Watahiki et al., 2023], or of the policy itself which can be used to decrease the exploration in the target task [Soni and Singh, 2006, Cheng et al., 2018, Joshi and Chowdhary, 2018, Zhang et al., 2021a, Wang et al., 2022, Yang et al., 2023, Gui et al., 2023, Chen et al., 2024].

While another way to reduce the exploration stage of a learning agent is to guide their behavior through an auxiliary reward [Brys et al., 2015, Gupta et al., 2017, Hu and Montana, 2019, Hejna et al., 2020], transferring parameters can be an effective way to transfer knowledge when tasks use the same architecture to model their solution, whether it is in the form of a value function [Torrey et al., 2006, Taylor and Stone, 2007a, Taylor et al., 2007a, Taylor and Stone, 2007b, Banerjee and Stone, 2007, Kuhlmann and Stone, 2007, Torrey et al., 2008, Taylor et al., 2008b], or a function approximator [Taylor et al., 2005, Taylor and Stone, 2005, Taylor et al., 2007b, Devin et al., 2017, Chen et al., 2019, Zhang et al., 2021b]. In other cases, the

inductive bias can be in the form of *if-then* rules that advise actions in the target task [Torrey et al., 2005], a criterion to select training data [Ammar and Taylor, 2012], adding the pre-activation output of a neural network to another network [Wan et al., 2020], complementary data to improve the approximation of the target dynamics [Taylor et al., 2008a], or supervision scores for imperfect data sets [Cao et al., 2022].

On the other hand, some works develop methods that are able to handle multiple sources of knowledge. For instance, in [Ammar et al., 2015a, Qian et al., 2020] policies are represented as a linear combination of a task-specific vector, a domain-specific basis and an inter-domain shared basis. By following a hierarchical structure, knowledge is combined and shared among tasks from the same domain through the domain-specific basis, and across all tasks through the inter-domain basis. In [Liu et al., 2023] an image encoder, trained with data sampled from different tasks, is used to provide an alternative representation for the RL to use as the observation space (instead of the original image space), which significantly speeds up the learning process. Other works evaluate source policies in the target domain to either weight the influence of multiple source policies in the target learner [Heng et al., 2022], or select the best source candidate [Talvitie and Singh, 2007], [Zhang et al., 2024].

The cross-domain knowledge transfer methods reviewed so far cover a wide range of settings, each of them exploiting the data and knowledge available. For instance, while the source selection methods [Talvitie and Singh, 2007, Heng et al., 2022, Zhang et al., 2024] are great for scenarios in which multiple sources of knowledge are available but it is not clear how/when they should be used, cross-domain imitation learning methods [Kim et al., 2020, Fickinger et al., 2021, Raychaudhuri et al., 2021, Franzmeyer et al., 2022, Zakka et al., 2022] take advantage of demonstrations of the target task but in a different state-action space. Thus, given the variety of approaches and objectives these methods pursue, there is no clear-cut criterion to sort from best to worst, as the impact of a method will depend on the requirements of the problem.

Considering that the knowledge transfer methods proposed in this document (see Chapter 4 and Chapter 5) select the most fit source task to transfer knowledge from, the most similar works to ours are [Ammar et al., 2015a, Qian et al., 2020, Heng et al., 2022, Talvitie and Singh, 2007, Zhang et al., 2024]. However, the main difference in our approach is that no external supervision is required in any step of

the transfer process, nor assumptions are made about the source and target tasks being related in any form.

3.3 Lifelong Reinforcement Learning

Beyond solving a sequence of independent knowledge transfer problems, in Lifelong RL the agent is constantly making decisions about how past experiences can aid learning the current task, as well as how the knowledge acquired in the present may affect its performance in the future. Two of the core problems in the lifelong learning setting are avoiding negative transfer and catastrophic forgetting, which have been dealt with in a variety of ways. For instance, [Lecarpentier et al., 2020] and [Xie and Finn, 2022] create a separate instance of a policy to solve each task and focus on performing positive transfer. While [Lecarpentier et al., 2020] exploit the fact that the optimal action-value function is Lipschitz continuous in the MDP space to provide a good heuristic to select a source task, [Xie and Finn, 2022] retain experiences from previous tasks (relabeled with the current task reward function) to pre-train an initial policy that will later be fine tuned with data from the current task.

Similarly, [Rusu et al., 2016] and [Liu et al., 2019] instance a neural network, to act as the policy in each task, and connect the output of middle layers from previous tasks to transfer knowledge to the most recent network (*i.e.,* lateral connections), while the weights of previous networks remain frozen. In [Schwarz et al., 2018] the lateral connection approach is also adopted, however, they restrict the architecture to only have two neural networks: the first one that is trained to solve the current task, and the second one to retain the knowledge of every task (as a knowledge base) and transfer knowledge to the first network, through policy distillation [Rusu et al., 2015]. Moreover, [Fernando et al., 2017] also present a framework suitable for lifelong RL that finds an optimal configuration of multiple neural networks (with genetic algorithms [Lambora et al., 2019]) to transfer knowledge through lateral connections to another network being trained to solve a different task than the first one. On the other hand, [Muppidi et al., 2024] address the loss of plasticity over time (which can cause negative transfer) by treating the LRL problem as a sequence of convex problems, which motivates the use of online convex optimization methods, while [Dick et al., 2024] focus on identifying the change in tasks, by comparing

recently gathered data sets with older ones in a latent space using the Wasserstein distance [Panaretos and Zemel, 2019].

Another way to avoid catastrophic forgetting is to build/learn policies from a set of reusable skills. For example, [Tessler et al., 2017] propose a hierarchical architecture in which, after pre-training a set of policies in different skills, learn a policy to use such skills to solve a variety of tasks. By acting with temporally extended actions, learning takes significantly fewer interactions in comparison to a standard RL policy. Similarly, [Tasse et al., 2021] and [Mendez et al., 2022] propose composing policies from a set of solutions to problems other then the current task. While [Tasse et al., 2021] use binary vectors to describe relevant objects in each task and grab knowledge from previous action-value function to compose a policy for the current task, in similar fashion to [Fernando et al., 2017], in [Mendez et al., 2022] a search for the optimal configuration is performed in which a set of modules can solve the current task in zero-shot (if a descriptor of the task is available), or few-shot learning after interacting with the environment.

Concerned with the memory efficiency of lifelong learning systems, some works propose sharing some the parameters across policies. For instance, [Ammar et al., 2014a, Ammar et al., 2015a, Mendez et al., 2018, Mendez and Eaton, 2020] propose representing the parameters to each task solution as a linear combination of task-specific and shared factors. In [Ammar et al., 2014a, Mendez and Eaton, 2020], policy gradient methods are used to update a shared latent basis and a task-specific sparse vector, whose product models the task-specific policy weights. This linear factorization is extended in [Ammar et al., 2015a] to include an additional factor between the shared basis and task-specific vectors, which is shared across tasks from the same domain. Additionally, [Mendez et al., 2018] showcase how the linear factorization approach can be used to model reward functions for the lifelong inverse RL setting.

On the other hand, some works devise methods to learn as many tasks as possible with a fixed set of parameters. For instance, [Kirkpatrick et al., 2017] mitigate catastrophic forgetting in a neural network by slowing down the changes in weights that are relevant for previous tasks (according to their Fisher information matrix). Instead, less relevant weights are the ones responsible for storing newer knowledge, such that the performance in older tasks is only harmed due to blackout catastrophe (*i.e.,* when the neural network's capacity is saturated). Furthermore,

[Isele and Cosgun, 2018] also exploit the storing capacity of a single neural network by adopting a strategy that replays experiences, from the replay buffer [Mnih et al., 2015], in order to not forget previously learned skills.

In lifelong learning, the learning agent is deployed to perform on a variety of tasks, during which there is no external supervision (excluding the reward signal) in the loop. To ensure that the agent will perform well despite the lack of supervision, it is necessary to endow it with methods that prevent negative transfer, as settings with more task diversity are addressed. Each of the methods reviewed in this section address a different set of challenges present in the LRL setting (*e.g.*, storing knowledge in a memory-efficient way, avoiding negative transfer, catastrophic forgetting), which makes sorting them from best to worst difficult. However, [Ammar et al., 2015a] are the only ones that address the cross-domain setting. Thus, in order to study the generalization capabilities of lifelong learning systems, the architecture proposed in this document (see Chapter 6) considers the problem of negative transfer in a cross-domain lifelong learning setting, which employs a model-based similarity function [García et al., 2022] to select the best source task from a progressively growing library of options.

3.4 Chapter Summary

In this chapter the most relevant works in the areas of MDP similarity, cross-domain knowledge transfer and lifelong reinforcement learning were covered. Although there are few similarity measures and lifelong learning systems that address the cross-domain RL setting, there is a growing interest in transferring knowledge across domains. Some of the works presented in Section 3.2 that transfer knowledge from multiple source can be extended to lifelong learning agents, as the selection of source of knowledge is a core step in positive transfer, thus, potentially providing a starting point to tackle lifelong cross-domain RL. However, the main difference between our proposal and the most related works is that the our method not only transfers knowledge across tasks from different domains, but it selects from which task it should transfer knowledge, without any special data requirements or supervision.

Chapter 4

Similarity for Knowledge Transfer in Discrete Spaces

In order to accelerate learning via transfer learning, the source and target tasks must be similar. Moreover, in scenarios when multiple sources of knowledge are available (*e.g.,* lifelong learning), being able to select knowledge related to the target task becomes a critical skill. In this chapter, a similarity function for RL tasks with discrete, but different, state-action spaces (based on their action value functions) is presented in Section 4.1. Additionally, a knowledge transfer method based on the similarity function is introduced in Section 4.2, whereas the experimental evaluation of both methods and discussion are detailed in Section 4.3.

4.1 Value-based Inter-Task Similarity Measure

To measure the similarity between two tasks with discrete state-action spaces, the proposed method compares their action-value functions, also known as Q functions. The main idea is to use a partially learned Q function (from the target task) to find the source task with a completely learned Q function that is the closest to the target counterpart. Given that the similarity measure presented in this section considers tasks with discrete spaces, each task is represented by a matrix that models its Q function. That is, a matrix containing the expected discounted return after executing an action a from a state s, and then follow a policy π (see Eq. 4.1) [Watkins and

Dayan, 1992], for every state-action pair.

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} \left\{ \Phi(s, \pi(s), s') V^{\pi}(s') \right\} \tag{4.1}$$

where $R(s,a)$ is the immediate reward observed after executing action a in state s, $\Phi(s, \pi(s), s')$ is the probability for the state of the agent to transit from s to s' after executing the action selected by the policy π in state s, and $V^{\pi}(s')$ is the value function that represents the expected discounted return if policy π is followed from state s' (see Sections 2.2 and 2.2.1 for more details).

Computing the similarity between two tasks, based on their Q functions, requires carrying out four main steps:

1. **Clustering Q-values** (Section 4.1.1): Group q-values row wise and column wise to build a domain-agnostic representation so that Q functions can be compared despite belonging to tasks with different state-action spaces.
2. **Determining Cluster Distributions** (Section 4.1.2): Model the size of each cluster' size as histograms.
3. **Comparing Cluster Distributions** (Section 4.1.3): Measure the similarity of cluster size distributions with the intersection between their respective histograms (see Eq. 4.6).
4. **Compute the Inter-task Similarity** (Section 4.1.4): After computing the intersection scores between multiple histograms, they are stored into a matrix, from which inter-task similarity is computed in as the mean value of such matrix.

The overall process of computing the intersection matrices is shown in Fig. 4.1, whereas the details of each of the steps above are presented in the following sections.

4.1.1 Clustering Q-values

Let $Q \in \mathbb{R}^{|S| \times |A|}$ be a real-valued matrix, where $Q_{i,j}$ represents the q-value $Q(s_i, a_j)$ (see Eq. 4.1) of state $s_i \in S$ and action $a \in A$. After applying a clustering process within each row of matrix Q, matrix $L \in \mathbb{R}^{|S| \times |A|}$ will contain the cluster ID/label assigned to each q-value by the clustering algorithm (see Fig. 4.2):

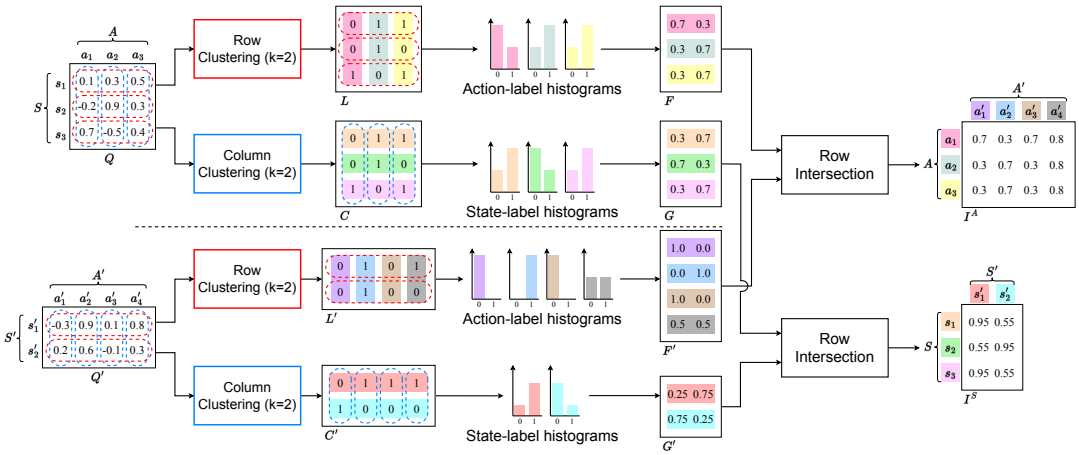


Figure 4.1: Computation of the intersection matrices I^A and I^S for a pair of action-value function matrices Q and Q'. Matrices L and C contain the labels assigned to the elements of the Q matrix if they were clustered row-wise (red dotted ovals) and column-wise (blue dotted ovals), respectively. Each row in matrix F is a histogram for the label frequency of each column in L, whereas in matrix G rows count the label frequency of each row in C. The elements of matrices I^A and I^S represent the intersection value between the rows of matrices F, F' and G, G', respectively. (Best seen in color.)

$$L_{i,j} = \underset{k}{\text{argmin}} |Centroid(Q_{i,*}, k) - Q_{i,j}| \tag{4.2}$$

where $Centroid(L_{i,*}, k) \in \mathbb{R}$ is the centroid of the k-th cluster in the i-th row $Q_{i,*} \in \mathbb{R}^{|A|}$ of Q. Similarly, we define the matrix $C \in \mathbb{R}^{|S| \times |A|}$ that will contain the labels assigned by a clustering algorithm to the elements of Q within the same column, that is:

$$C_{i,j} = \underset{k}{\text{argmin}} |Centroid(Q_{*,j}, k) - Q_{i,j}| \tag{4.3}$$

In other words, the labels in L represent how the q-values of every action are distributed with respect each state, whereas C models the q-value distribution of every state over each action. Additionally, before assigning values to matrices L and C, the centroid IDs are sorted in ascending order. That is, the following expression is true for any pair of centroids:

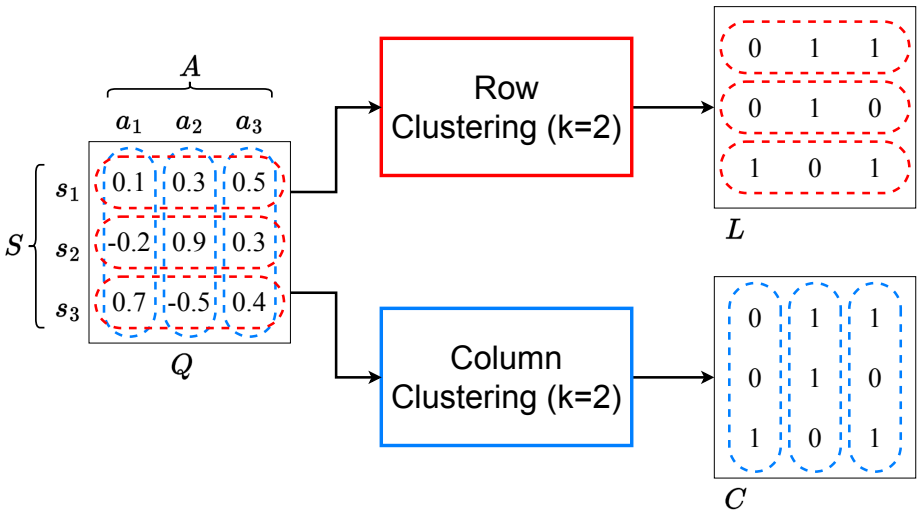


Figure 4.2: Clustering process of Q values. The Q values of a row are clustered into k groups, where the label assigned to the Q values in each group are shown in the same row-column position in matrix L (this process is applied to each row of matrix Q). Similarly, the same process is performed with every column of matrix Q to produce matrix C.

$$Centroid(Q_{i,*}, k) \leq Centroid(Q_{i,*}, l)\ \forall i$$
$$Centroid(Q_{*,j}, k) \leq Centroid(Q_{*,j}, l)\ \forall j$$
$$\text{such that } k < l$$

where by sorting the centroid IDs it is possible to interpret them as clusters of higher expected returns the higher the ID is. Sorting cluster labels so that this interpretation is true is what enables the comparison of state-action pairs from different spaces possible, as their similarity is defined on how their Q value compares to those of other state-action pairs from their same space.

Moreover, Section 4.1.2 describes how the cluster distribution is computed and modeled as histograms.

4.1.2 Determining Cluster Distributions

After clustering states and actions, according to their value with respect actions and states, their label occurrences are counted from matrices L and C (see Fig. 4.3). Let $F \in \mathbb{R}^{|A| \times k}$ and $G \in \mathbb{R}^{|S| \times k}$ be a pair of matrices that in each row will contain the

label-count histogram of each action and state (respectively), where k is the number of clusters used in the row/column clustering process (see Eq. 4.2 and Eq. 4.3). The i-th row of F, $F_{i,*} \in \mathbb{R}^k$, contains a normalized histogram of the labels in the i-th column of L, as described by Eq. 4.4:

$$F_{i,l} = \frac{1}{|S|} \sum_{j=0}^{|S|-1} [l = L_{j,i}] \tag{4.4}$$

where $[\cdot]$ is the Iverson bracket, which returns 1 if the condition within the brackets is satisfied and 0 otherwise [Gehr et al., 2016]. Similarly, Eq. 4.5 describes how the normalized histograms of state labels occurrences are computed:

$$G_{i,l} = \frac{1}{|A|} \sum_{j=0}^{|A|-1} [l = C_{i,j}] \tag{4.5}$$

Each row in F describes the distribution of labels an action received in the row clustering processes, while the rows in G model the label assignment distribution of each state across the action clustering processes. In other words, given that the centroid IDs are sorted in ascending order (with respect to their centroids), each row in F can be interpreted as a distribution of how preferred its respective action is from the set of states. Similarly, each row in G can be interpreted to model the distribution of how preferable they are to take actions from.

Furthermore, in Section 4.1.3 the process to estimate the similarity between states and actions, through the comparison of their label occurrence histograms, is explained.

4.1.3 Comparing Cluster Distributions

The action and state label histograms of a Q function model the role actions and states have (within the task partially/completely learned) in terms of how they contribute to achieving long-term cumulative rewards (*i.e.,* the goal of RL agents). Thus, to determine the similarity between states/actions from different tasks, we propose a domain-agnostic method to compare their roles within their respective tasks that consists of computing the intersection of their label occurrences histograms

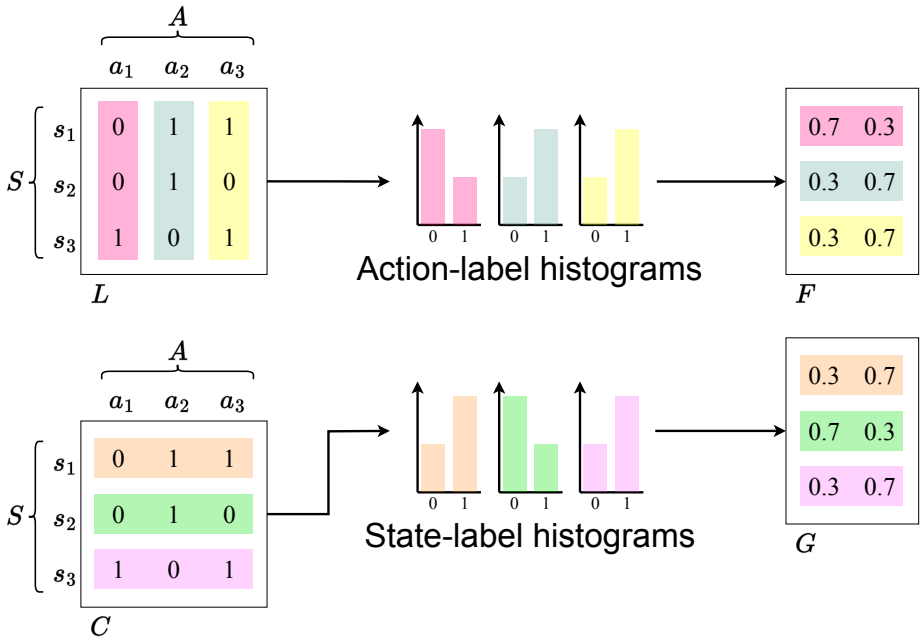


Figure 4.3: An action-label histogram is built by counting the number of occurrences of each label in a column of matrix L (this process is repeated for each column). Similarly, a state-label histogram is built counting the label occurrences in each row of matrix C. Action-label and state-label histograms are stored as rows in matrices F and G, respectively.

(see Fig. 4.4).

Let $Q \in \mathbb{R}^{|S|\times|A|}$ and $Q' \in \mathbb{R}^{|S'|\times|A'|}$ be two matrices that represent the Q function of two MDPs with state-action spaces of different dimensions (*i.e.*, $|S\times A| \neq |S'\times A'|$), and $F \in \mathbb{R}^{|A|\times k}, G \in \mathbb{R}^{|S|\times k}$ and $F' \in \mathbb{R}^{|A'|\times k}, G' \in \mathbb{R}^{|S'|\times k}$ be the matrices containing the label occurrences histograms for Q and Q', respectively. Then, we define matrices $I^A \in \mathbb{R}^{|A|\times|A'|}$ and $I^S \in \mathbb{R}^{|S|\times|S'|}$ to hold the intersection values (see Eq. 4.6) between rows from F, G and F', G', respectively.

$$Intersection(u, v) = \sum_i min(u_i, v_i) \quad \text{where } u, v \in \mathbb{R}^k \tag{4.6}$$

Specifically, the computation of values in I^A and I^S is described by Eq. 4.7 and Eq. 4.8:

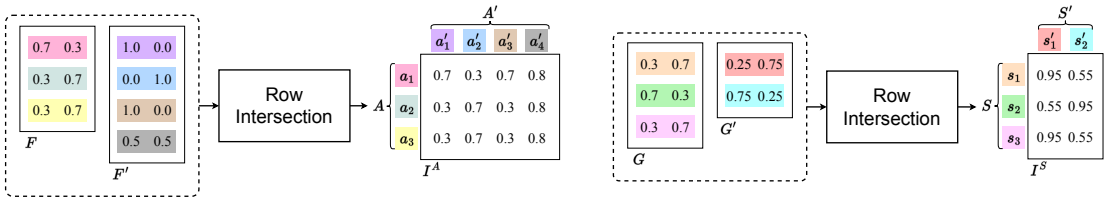


Figure 4.4: After the action-label and state-label histograms have been built for two Q tables, their similarity is computed as the intersection between histograms (see Eq. 4.6) and stored in matrices I^A and I^S. Then, the mean values of I^A and I^S are computed to yield the action-based and state-based similarity scores, respectively.

$$I^A_{i,j} = Intersection(F_{i,*}, F'_{j,*}) \tag{4.7}$$

$$I^S_{i,j} = Intersection(G_{i,*}, G'_{j,*}) \tag{4.8}$$

That is, each element of I^A describes how much the role of a pair of actions within their respective tasks overlaps, where the higher the score the more similar they are, and an intersection value of 1 represents a perfect match. Analogously, the elements in I^S model the similarity between states of the pair of tasks.

Additionally, in Section 4.1.4 the process to compute the overall similarity across tasks is described.

4.1.4 Computing Inter-Task Similarity

Considering that each element in matrices I^A, I^S represents the similarity between individual elements (either between states or actions), in order to assess the similarity between two tasks, we propose to use the mean operator (Eq. 4.9) [Golub and Van Loan, 2013] on both matrices.

$$\bar{M} = \frac{1}{m \cdot n} \sum_{i}^{m} \sum_{j}^{n} M_{i,j} \tag{4.9}$$

Although the mean operator satisfies the four properties necessary to be considered a similarity function (see Section 2.6), we propose to evaluate this operation based on its ability to work as a source-selection heuristic.

In Section 4.2, the proposed method to transfer knowledge, based on the similarity definitions described in this section, across tasks with different state-action spaces is presented.

4.2 Knowledge Transfer

After selecting the most similar source task Q function Q to the partially learned Q function Q', using one of the following scores $\bar{I}^A, \bar{I}^S$, knowledge is transferred in the form of q-values from Q to Q' (see Fig. 4.5). To transfer q-values from Q to Q', the following steps are performed for every state-action pair in the target task, that is, $\forall(s'_i, a'_j) \in S' \times A'$:

1. Find the state from the source task $s \in S$ that is the most similar to the target task state $s'_i \in S'$:
$$s = \underset{s_k \in S}{\text{argmax}} I^S_{k,i}$$

2. Find the action from the source task $a \in A$ that is the most similar to the target task action $a'_j \in A'$:
$$a = \underset{a_k \in A}{\text{argmax}} I^A_{k,j}$$

3. Transfer the q-value of the most similar source state-action pair $Q(s, a)$ to be the q-value of the target state-action pair (s'_i, a'_j) in Q', that is:
$$Q'(s'_i, a'_j) \leftarrow Q(s, a)$$

After performing steps 1-3 for every state-action pair in the target task, we let the RL algorithm run its course. The main motivation to transfer knowledge in this way is to accelerate learning by assigning the completely learned q-value to its partially learned counter part in the target task Q function, under the assumption that partially learned Q function is directed towards a solution that is similar to the selected source Q function.

In Section 4.3 a series of experiments that evaluate the ability of the similarity measure presented in Section 4.1 to select useful sources of knowledge, and the knowledge transfer method from this section to improve learning are detailed and

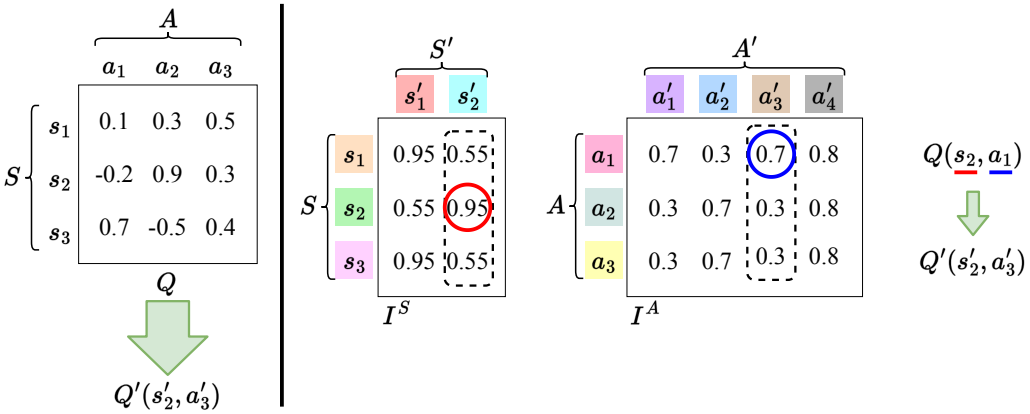


Figure 4.5: Example of an action-value transfer using the action-value function and intersection matrices form Fig. 4.1. To transfer from Q to $Q'(s'_2, a'_3)$ the state $s \in S$ and action $a \in A$ with the largest intersection value to $s'_2 \in S'$ and $a'_3 \in A'$ are selected. In this example, $a_1 \in A$ is the most similar action to $a'_3 \in A'$, whereas state $s_2 \in S$ is the most similar to $s'_2 \in S'$. Therefore, the action value $Q(s_2, a_1)$ is transferred to $Q'(s'_2, a'_3)$.

discussed.

4.3 Experiments

4.3.1 Experimental Setting

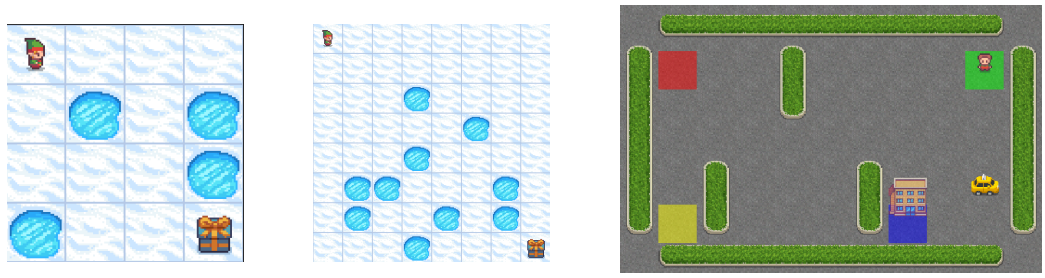
To evaluate the proposed similarity measure and knowledge transfer method, six environments from the OpenAI Gym suite [Brockman et al., 2016] (now Gymnasium [Towers et al., 2023]) were used: Frozen Lake (FL), Frozen Lake 8×8 (F8), Taxi Domain (TA) [Dietterich, 2000], Acrobot (AC) [Sutton, 1995], Mountain Car (MC) [Moore, 1990] and Pendulum (PE). Considering the proposed methods are applicable to RL tasks with discrete spaces, the state and action spaces of AC, MC and PE were discretized (see Table 4.1 for the dimensions of each task's space). In the set of evaluation tasks, FL and F8 are significantly more similar than to any other tasks, and we would like to evaluate if the proposed knowledge transfer method is able to exploit such structural similarities if no state and action mappings are available. Additionally, the other tasks were included to provide a larger scope of how the similarity measure behaves when non-related tasks are available for knowledge transfer purposes.

The objective of the present experiments is to evaluate:

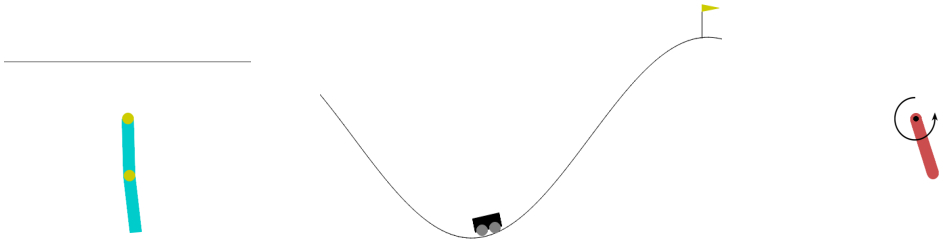
1. The order in which the similarity measure ranks source tasks for each target task,
2. The benefit of transferring knowledge, and
3. The data (sampled from the target task) required to estimate a low-error similarity score

In order to learn a Q function in every environment, the Q-learning algorithm was employed as the base learning method. To completely learn a Q function in every environment, the agent was trained for 300,000 episodes, interleaved with 10 evaluation episodes (which are reported in Fig. 4.7) every 100 training episodes. Additionally, the following parameters were used:

- Learning from scratch: both the learning rate and exploration probability were initialized with 0.95 and linearly decreased to a final value of 0.01.



(a) FL (b) F8 (c) TA



(d) AC (e) MC (f) PE

Figure 4.6: Set of evaluation tasks. In Frozen Lake (FL) and Frozen Lake 8×8 (F8) the goal is achieved when the elf navigates (with stochastic actions) to the treasure location (without falling into the fishing holes). The Taxi Domain (TA) presents a navigation problem in which the taxi-driver agent must pick up the passenger and drop it in a specific location. In Acrobot (AC), Mountain Car (MC) and Pendulum (PE) the RL agent must learn to apply small forces to balance the poles and car (in the case of MC) to reach certain position that is out of reach if force is constantly being applied.

- Learning with transferred knowledge: After initializing the target task Q function (using the method from Section 4.2 and a Q function completely learned from scratch for the target task), the learning rate linearly decreased from 0.5 to 0.01, while the exploration probability decreased from 0.25 to 0.01.

- Clustering q-values: In order to cluster q-values, to compute matrices L, C as part of the similarity measurement (see Section 4.1.1), the K-means algorithm [Jain and Dubes, 1988] was used, with a number of clusters of $k = 3$, as it is the largest number of clusters that can be used in every state/action space of the the task presented in Table 4.1.

Table 4.1: Collection of test environments. Columns present (left to right): the name of the environment, the size of the state space, and the size of the action space.

Environment	$\|S\|$	$\|A\|$
Frozen Lake (FL)	16	4
Frozen Lake 8x8 (F8)	64	4
Taxi (TA)	500	6
Acrobot (AC)	6400	3
Mt. Car (MC)	400	3
Pendulum (PE)	640	20

4.3.2 Results

In Table 4.2 the similarity scores $\bar{I}^S$ and $\bar{I}^A$ between each pair of tasks are shown. It is worth noting that among the complete set of task pairs, most tasks obtained the highest similarity score when compared to another copy of them, for both similarity functions. Additionally, In the case of the action-based similarity function (*i.e.,* $\bar{I}^A$), when a task obtained the highest score with a copy of itself, the similarity score was above 0.84. On the other hand, Table 4.3 shows the order in which the set of tasks would be ranked as option to transfer knowledge from for every target task, based on the similarity scores reported in Table 4.2.

In terms of learning improvement, as a consequence of using the knowledge transfer method described in Section 4.2, Table 4.4 shows the transfer learning performance to learning from scratch performance ratio (*i.e.,* a measure of how positive was the transferred knowledge in the target agent in comparison to learning from scratch) given knowledge was transferred to every (target) task from every tasks that is different. Additionally, Fig. 4.7 shows:

- The performance of the learning agent (*i.e.,* green graph),
- The similarity score between two completely learned Q functions of the same task (*i.e.,* solid lines), and
- The similarity score between the Q function being learned (by the learning agent) and a completely learned Q function of the same task (*i.e.,* dashed lines).

In other words, Fig. 4.7 shows how good the similarity estimation between a par-

Table 4.2: Inter-task similarity scores based on the mean value of matrices I^S and I^A. The larger the value, the more similar the tasks. Values in the main diagonal that are the largest value in its row and column are in bold. The similarity scores in the lower triangle of each similarity table are omitted since similarity matrices are symmetrical.

	I^S						I^A					
	FL	F8	TA	AC	MC	PE	FL	F8	TA	AC	MC	PE
FL	**0.49**	0.46	0.48	0.13	0.35	0.24	0.78	0.76	0.62	0.58	0.79	0.76
F8	–	**0.51**	0.48	0.13	0.42	0.25	–	**0.85**	0.64	0.50	0.80	0.71
TA	–	–	**0.51**	0.16	0.40	0.27	–	–	0.62	0.50	0.62	0.57
AC	–	–	–	**0.81**	0.37	0.56	–	–	–	**0.98**	0.61	0.77
MC	–	–	–	–	**0.46**	0.38	–	–	–	–	**0.89**	0.80
PE	–	–	–	–	–	0.45	–	–	–	–	–	**0.88**

Table 4.3: Ranking of source tasks for a target task based on the inter-task similarity measures $\bar{I}^S$ and $\bar{I}^A$. Each row shows, from left to right, the acronyms of the target task, the source tasks ranked from most to least similar using the $\bar{I}^S$ score, followed by the source tasks ranked from most to least similar using the $\bar{I}^A$ score.

Target task	Source task ranking											
	$\bar{I}^S$						$\bar{I}^A$					
	1st	2nd	3rd	4th	5th	6th	1st	2nd	3rd	4th	5th	6th
FL	FL	TA	F8	MC	PE	AC	MC	FL	F8	PE	TA	AC
F8	F8	TA	FL	MC	PE	AC	F8	MC	FL	PE	TA	AC
TA	TA	F8	FL	MC	PE	AC	F8	MC	TA	FL	PE	AC
AC	AC	PE	MC	TA	FL	F8	AC	PE	MC	FL	F8	TA
MC	MC	F8	TA	PE	AC	FL	MC	F8	PE	FL	TA	AC
PE	AC	PE	MC	TA	F8	FL	PE	MC	AC	FL	F8	TA

tially and completely learned Q functions is with respect on the number of training episodes.

4.3.3 Discussion

Regarding the first evaluation objective of the experiments (*i.e.,* the similarity-based rankings of source tasks), Table 4.3 shows that in most of the cases, a target task would select its source twin as the most similar among the set of source tasks, with both similarity functions (*i.e.,* $\bar{I}^S$ and $\bar{I}^A$). In the case of the state-based similarity function (*i.e.,* $\bar{I}^S$), the correct source task was ranked among the top-2 most similar tasks. These results provide evidence that the similarity functions

Table 4.4: Transfer learning performance to learning from scratch performance ratio. Columns present the task from which knowledge was transferred, while rows present the task in which the agent trained with the additional knowledge. Values above 1 represent a performance improvement in comparison to learning from scratch, while those below 1 represent a performance deterioration. The values from the main diagonal are omitted because knowledge was transferred across different tasks only.

Target task	Source task					
	FL	F8	TA	AC	MC	PE
FL	–	1.67	0.35	0.001	0	0
F8	1.27	–	0.05	0.0002	0.0006	0
TA	0.99	0.99	–	1.01	1.01	-23.96
AC	1.15	1.16	1.17	–	1.17	1.18
MC	1.08	1.08	1.08	1.08	–	1.08
PE	1.32	1.31	1.32	1.31	1.31	–

have the ability to autonomously select (in most cases) a source task that would benefit the target learner. Moreover, being able to identify task-relevant similarity with out labels is a crucial requirement for the development of autonomous agents capable of transferring knowledge. By identifying that a very similar task has already been learned, the agent could learn a near-optimal policy with the help of auxiliary knowledge transferred from the selected source task.

Regarding the second evaluation objective (*i.e.,* the benefit of transferring knowledge), Table 4.4 shows the transfer learning performance to learning from scratch performance ratios obtained by transferring to each target task from every source task but itself. It is worth noting that the knowledge transfer method is able to exploit the structural similarity between Frozen Lake (FL) and Frozen Lake 8 (F8), given that the state space of F8 is a super set of FL, which is shown as both tasks benefit from transferring knowledge from each other. Additionally, the significant differences between FL, F8 and the control-based tasks (*i.e.,* AC, MC, PE) also affect negatively transferring from AC, MC and PE to Fl, and F8. Considering that the state spaces of the control-based tasks (see Table 4.1) are significantly larger than FL and F8's, there are more state-action pairs form which to choose, thus, increasing the opportunities for the knowledge transfer method to transfer an action value that does not actually relate to the target state-action pair. However, if a system were to use the state-based similarity function (*i.e.,* $\bar{I}^S$), the knowledge transfer method would ensure a positive transfer for every target task, even for the Pendulum (PE), which ranked Acrobot as the most similar task (see Table 4.3) and

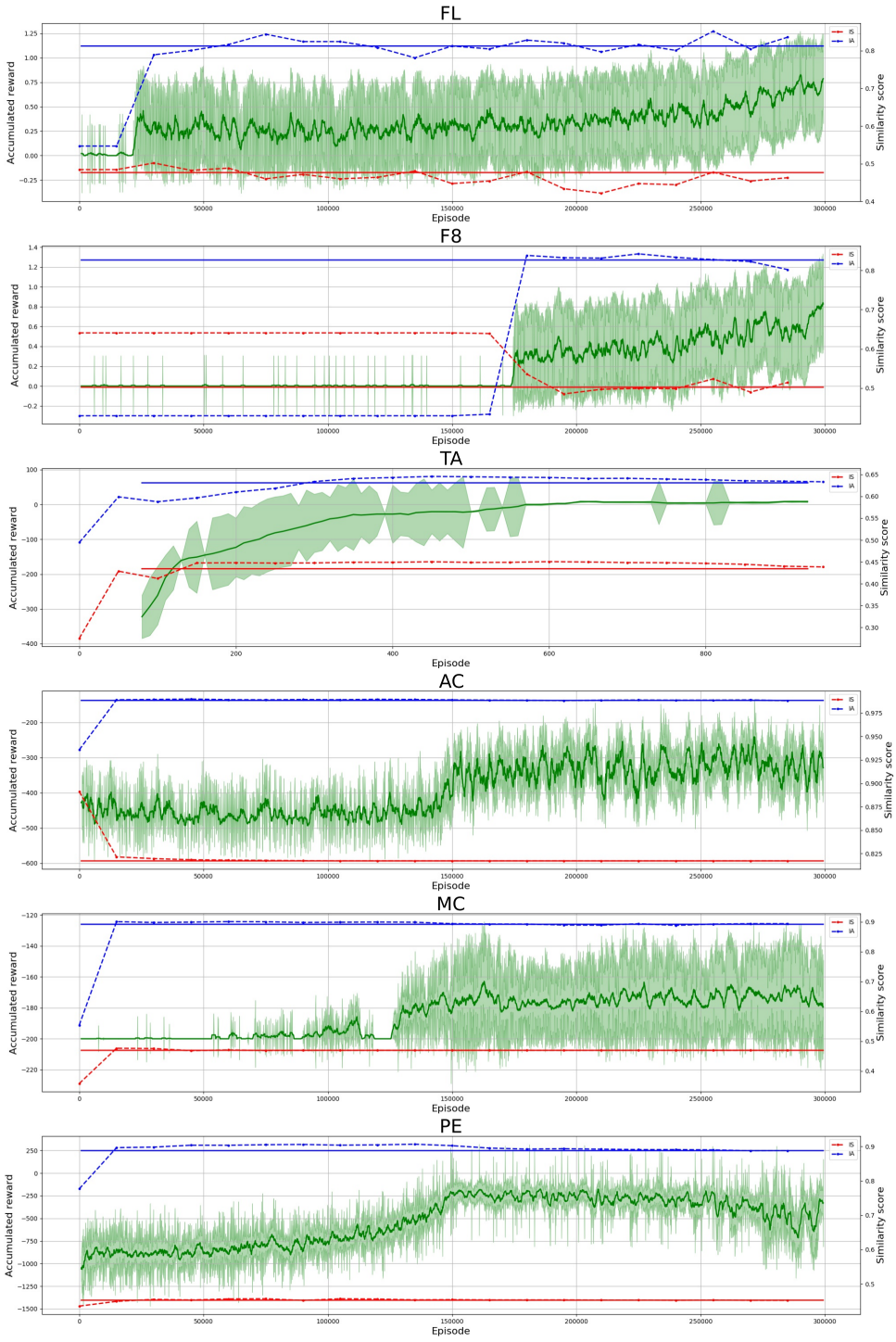


Figure 4.7: Progression of similarity scores between a partially learned Q function and completely learned Q function of the same task (*i.e.,* red and blue dashed lines for $\bar{I}^S$ and $\bar{I}^A$ respectively) with respect to the learning agent's performance (*i.e.,* green graph). The horizontal axis shows the training episodes while the left and right vertical axes show the agents accumulated reward and similarity scores, respectively. The solid red and blue lines show the best possible similarity scores for reference.

obtained a transfer learning performance to learning from scratch performance ratio of 1.31 (see Table 4.4).

Concerning the third (and last) evaluation objective (*i.e.,* the data cost of computing good estimations of the inter-task similarity), Fig. 4.7 shows the evolution of the similarity scores between the partially learned and completely learned Q functions of the same task through the learning process of an RL agent (*i.e.,* dashed lines). In the case of Frozen Lake (FL), Frozen Lake 8×8 (F8) and Taxi (TA), the similarity scores get near to the best possible similarity score (*i.e.,* solid lines) only until the agent increases its performance, whereas in Acrobot (AC), Mountain Car (MC) and Pendulum (PE), a good estimation of the similarity score is obtained at an early stage of the training process. In order to exploit the knowledge transfer method, it is necessary to develop similarity measures that require fewer data to produce good estimations, regardless of the task at hand.

Ideally, a similarity measuring function should be able to select sources of knowledge that a knowledge transfer method can exploit to accelerate the learning process of a learning agent. Additionally, both of these process (*i.e.,* knowledge *selection* and *adaptation*) must require an amount of interactions with the target task environment (*i.e.,* data) that is lower than what learning from scratch. The ranking order of the source tasks in Table 4.3 and transfer learning performance to learning from scratch performance ratios in Table 4.4 suggest that the proposed state-based similarity function (*i.e.,* $\bar{I}^S$) and knowledge transfer method (see Section 4.2) can produce a positive transfer (*i.e.,* transfer learning performance to learning from scratch performance ratio greater than 1.0). However, the data cost of obtaining a reliable similarity score showed to be largely variable, shown in Fig. 4.7, as for half of the tasks the similarity function obtained a good estimation when the learning agent showed an improvement in its performance. In other words, although the similarity functions shows an ability to select sources of knowledge that provide a positive transfer, it still needs a lower sample complexity (*i.e.,* data cost) for it to be employed in online reinforcement learning problems. Future work on this end could focus on estimating the commonalities between the reward and transition models separately, which may be less data costly to approximate, in comparison to learning the value function.

4.4 Chapter Summary

An inter-task similarity function and a knowledge transfer method for tasks with discrete state-action spaces were presented this chapter. The similarity function compares the action-value function of two MDPs to determine their compatibility for transfer purposes (*i.e.,* a performance-based similarity measure [García et al., 2022]). Then, the knowledge transfer method transfer q values, among the most similar state-action pairs across tasks, to accelerate the learning process in the target task. Although the similarity function and transfer method showed an ability to produce positive transfer in variety of the target tasks, it is too data expensive for online reinforcement learning.

Chapter 5

Model-based Similarity for Cross-Domain Knowledge Transfer

Every time a lifelong learning agent encounters a new task, it has the option to reuse knowledge acquired by any of the previously learned tasks. To avoid negative transfer, it is important that the agent transfers knowledge only from sources that are related to the target task. That is, when multiple sources of knowledge are available, the processes of selection and adaptation of knowledge are equally important to produce a performance improvement in the target task. However, if the source and target tasks are defined over different state-action spaces (*i.e.,* cross domain), measuring their similarity by directly comparing their transition/reward function (*i.e.,* model-based similarity [García et al., 2022]) is not possible due to the space mismatch.

Therefore, in order to transfer from multiple source tasks to a task with different state-action space, in this chapter a model-based similarity measure and knowledge transfer method are presented. As shown in Fig. 5.1, the main steps performed to transfer knowledge in this setting are the following:

1. State-Action Spaces Alignment (Section 5.1): In order to compare the reward and transition functions of two tasks from different domains, a reward-based alignment of their state-action spaces is performed. This process results in learning a set of functions (modeled as neural networks) that map states and actions across domains.

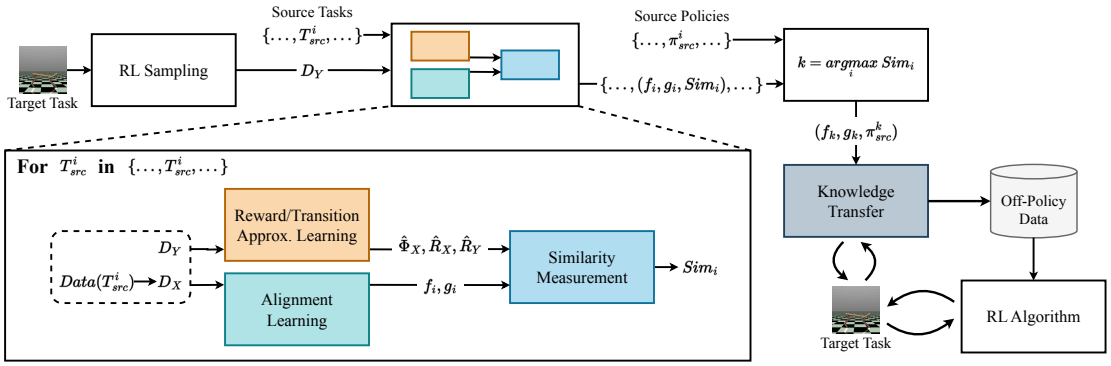


Figure 5.1: Overview of the similarity-based knowledge transfer method. Given a data set sampled from the target task (D_Y), and a set of source data sets (D_X), a set of alignment mappings is learned (f_i, g_i) for every source-target pair (see Section 5.1), followed by an evaluation of their similarity to the target task (Sim_i), with the help of approximated transition ($\hat{\Phi}_X$) and reward models ($\hat{R_X}, \hat{R_Y}$) (see Section 5.2). Then, the mappings from the most similar source task are used to transfer its policy (π^k_{src}) to the target task to accelerate the learning process (see Section 5.3).

2. Dynamics-based Similarity Measurement (Section 5.2): With the mapping functions learned in the previous step, the similarity between two tasks is computed by comparing how similar their transition and reward function (*i.e.*, approximated from data) predictions are.

3. Similarity-based Knowledge Transfer (Section 5.3): After measuring the similarity between the target task and every source task, the policy learned in the most similar source task is transferred (through the mappings learned in step 1) to take actions in the target task and alleviate the exploration required to learn.

5.1 Reward-based Alignment

In order to align the state-action spaces of two different domains, we propose an immediate reward alignment criterion, where the main idea is that matching state-action pairs that produce similar rewards can help transferring optimal behaviors. To learn the state and action mapping functions, we take inspiration from the unsupervised alignment method proposed in [Wang and Mahadevan, 2009]. In the method proposed by [Wang and Mahadevan, 2009], given data sets $X = \{x_i | x_i \in \mathbb{R}^p\}$ and

$Y = \{y_j | y_j \in \mathbb{R}^q\}$, a pair of spaces are aligned in an m-dimensional latent space (see Section 2.3 for a definition of latent space), by minimizing the cost function described in Eq. 5.1:

$$C(\alpha, \beta) = \mu \sum_{x_i \in X, y_j \in Y} (\alpha^T x_i - \beta^T y_j)^2 W_{i,j} + 0.5 \sum_{x_i, x_j \in X} (\alpha^T x_i - \alpha^T x_j)^2 W^X_{i,j} + 0.5 \sum_{y_i, y_j \in Y} (\beta^T y_i - \beta^T y_j)^2 W^Y_{i,j} \quad (5.1)$$

where $\mu \in \mathbb{R}$ is coefficient that weighs the importance of the inter-domain alignment term, $\alpha \in \mathbb{R}^{p \times m}$ and $\beta \in \mathbb{R}^{q \times m}$ are matrices that are use to linearly map data points from X and Y to the latent space, and $W \in \mathbb{R}^{|X| \times |Y|}, W^X \in \mathbb{R}^{|X| \times |X|}, W^Y \in \mathbb{R}^{|Y| \times |Y|}$ are matrices that hold the similarity between points from different spaces (*i.e.*, inter-domain similarity in W) and between points from the same space (*i.e.*, intra-domain similarity in W^X, W^Y). Thus, to minimize Eq. 5.1, α, β must map near in the latent space the data points that have high similarity values in W, W^X, W^Y.

To compute the similarity between points, [Wang and Mahadevan, 2009] compare how the nearest neighbors of each point are distributed around it. A local neighborhood of a data point x_i is represented by a square matrix of dimensions $R^{x_i} \in \mathbb{R}^{(K+1) \times (K+1)}$, where the first row and column of R^{x_i} correspond to x_i, while the other K rows and columns correspond to the K-nearest neighbors of x_i (in ascending order with respect to their distance to x_i). Considering that $R^{x_i}_{j,k}$[1] represents the Euclidean distance between the data points that correspond to the j-th row and k-th column (respectively), R^{x_i} describes the local structure of a space surrounding data point x_i. Then, the local patterns R^{x_i}, R^{y_j} of two points x_i, y_j are compared by a distance function $dist : \mathbb{R}^{(K+1) \times (K+1)} \times \mathbb{R}^{(K+1) \times (K+1)} \rightarrow \mathbb{R}$ (see Section 3.2 in [Wang and Mahadevan, 2009]) that evaluates how well the patterns of neighbors around x_i and y_j match each other (the better the match, the smaller $dist(R^{x_i}, R^{y_j})$ is). Finally, the similarity values are computed from the distance between patterns with a kernel: $W_{i,j} = e^{-dist(R^{x_i}, R^{y_j})}, W^X_{i,j} = e^{-dist(R^{x_i}, R^{x_j})}, W^Y_{i,j} = e^{-dist(R^{y_i}, R^{y_j})}$.

In Eq. 5.1, matrices W, W_x, W_y are computed *a priori*, and describe how similar inter-domain (*i.e.*, x_i, y_j) and intra-domain (*i.e.*, x_i, x_j and y_i, y_j) pairs are. In other

[1] Single elements of a matrix A are indicated by sub indexes (*e.g.*, $A_{i,j}$), while single elements of a vector a are accessed with brackets (*e.g.*, $a[i]$).

words, the objective of the cost function is to map data points near to each other in the latent space if they are similar in their original spaces (*i.e.,* X, Y) or to a data point from the other domain. However, given that the purpose of aligning RL state-action spaces is to transfer policies that maintain optimal behaviors, the cost function from Eq. 5.1 does not meet our needs in the following aspects:

- Equation 5.1 was devised to train linear mappings. If non-linear function approximators (*e.g.,* neural networks) were used to align tasks with arbitrarily complex structures, training such models with Eq. 5.1 may result in degenerate solutions where every data point is mapped to a narrow neighborhood of the latent space.
- The inter-domain similarity (*i.e.,* the similarity between x_i and y_j) is based on how similar the local neighborhood structures of x_i and y_j. Since we can not ensure RL tasks will have spaces with similar structures, we propose using the immediate rewards as alignment criterion, which have an identical meaning across all RL tasks (*i.e.,* the how preferable actions are within a 1-step horizon).

Therefore, to overcome the restrictions imposed by the alignment method in a cross-domain RL setting, we present a reward-based alignment loss function that encourages mapping states-action pairs close to each other if they produce similar rewards, while preserving the geometrical structure present in the original state-action spaces (*e.g.,* if two states are far from each other in their original space, then they should be mapped far away in the latent space, and vice versa).

Let $f_\theta : \mathbb{R}^p \to \mathbb{R}^m$ and $g_\omega : \mathbb{R}^q \to \mathbb{R}^m$ be a pair of functions parametrized by trainable parameters θ, ω respectively (*e.g.*, neural networks), and D_x, D_y a pair of training data sets made of normalized data, defined as follows:

$$D_X = \{(x_i, r^{x_i}) | x_i \in \mathbb{R}^p,\ r^{x_i} \in [0,1],\ x_i[k] \in [0,1] \text{ for } k = 1, ..., p\}$$
$$D_Y = \{(y_j, r^{y_j}) | y_j \in \mathbb{R}^q,\ r^{y_j} \in [0,1],\ y_j[k] \in [0,1] \text{ for } k = 1, ..., q\}$$

where x_i, y_j are normalized vectors from different spaces, and r^{y_j}, r^{y_j} are the normalized rewards associated to x_i, y_j, respectively. Then, the alignment loss is computed by Eq. 5.2:

$$L_A = \sum_{(x_i, r^{x_i}) \in D_X, (y_j, r^{y_j}) \in D_Y} -Cos(f_\theta(x_i), g_\omega(y_j))\, W(r^{x_i}, r^{y_j}) \tag{5.2}$$

$$W(r^{x_i}, r^{y_j}) = exp\left(\frac{-|r^{x_i} - r^{y_j}|}{\delta^2}\right) \tag{5.3}$$

where $Cos(\cdot,\cdot)$ is the cosine similarity which measures the angle between $f_\theta(x_i)$ and $g_\omega(y_j)$, $W(\cdot,\cdot)$ a function that computes the similarity coefficient given a pair of rewards, and δ a constant that controls how steep the similarity is with respect to the reward difference. Considering that non-linear function approximators (*e.g.*, neural networks) are capable of mapping a wide variety of input vectors to a narrow region of the latent space (contrary to linear mappings that are restricted to their linear nature), instead of using the Euclidean norm such as in Eq. 5.1, Eq. 5.2 employs the cosine similarity to measure differences between latent vectors since it can not be minimized with arbitrary vectors of small magnitude.

To preserve the local neighborhood relations (that are present in the original spaces) in the latent space, we use the geometry preserving loss, described in Eq. 5.4:

$$\begin{aligned} L_G = & \sum_{(x_i,r^{x_i}),(x_j,r^{x_j})\in D_X} \frac{[Cos_d(x_i,x_j) - Cos(f_\theta(x_i), f_\theta(x_j))]^2}{|D_X|^2 - |D_X|} + \\ & \sum_{(y_i,r^{y_i}),(y_j,r^{y_j})\in D_Y} \frac{[Cos_d(y_i,y_j) - Cos(g_\omega(y_i), g_\omega(y_j))]^2}{|D_Y|^2 - |D_Y|} \end{aligned} \tag{5.4}$$

$$Cos_d(a,b) = -2 \cdot d(a,b) + 1 \tag{5.5}$$

$$d(a,b) = \frac{\sqrt{\sum_i^N (a[i] - b[i])^2}}{\sqrt{N}} \tag{5.6}$$

where N is the dimensionality of vectors $a, b \in \mathbb{R}^N$, $d(\cdot,\cdot)$ is the normalized Euclidean distance between two vectors, and $Cos_d(\cdot,\cdot) : [0,1] \rightarrow [-1,1]$ is a linear mapping from the normalized Euclidean distance to the range of the Cosine similarity. Given that Eq. 5.5 provides a direct mapping from the Euclidean distance between normalized vectors to the Cosine similarity, it is employed in Eq. 5.4 as a reference of how far two vectors mapped to the latent space should be, depending on how far they are from each other in their original space.

Additionally, to measure similarity and transfer policies across tasks, it is necessary to map states and actions across domains. Therefore, we train functions f_θ, g_ω

along with functions $f_\phi^{-1} : \mathbb{R}^m \to \mathbb{R}^p, g_\psi^{-1} : \mathbb{R}^m \to \mathbb{R}^q$, which are trained to decode the latent vectors, that the encoders f_θ, g_ω yield, back to the domains of f_θ, g_ω. To maintain the consistency between the latent space and each original space (*i.e.*, X, Y) we minimize the reconstruction loss (see Eq. 5.7), while the cycle-consistency loss (see Eq. 5.8) encourages consistency across spaces X and Y, as described below:

$$\begin{aligned} L_{R_X} &= \left(x - f_\phi^{-1} \circ f_\theta(x)\right)^2 \\ L_{R_Y} &= \left(y - g_\psi^{-1} \circ g_\omega(y)\right)^2 \end{aligned} \tag{5.7}$$

$$\begin{aligned} L_{C_X} &= \left(x - f_\phi^{-1} \circ g_\omega \circ g_\psi^{-1} \circ f_\theta(x)\right)^2 \\ L_{C_Y} &= \left(y - g_\psi^{-1} \circ f_\theta \circ f_\phi^{-1} \circ g_\omega(y)\right)^2 \end{aligned} \tag{5.8}$$

where $\circ$ is the function composition operator. In other words, Eq. 5.7 trains the encoder functions that map data points to the latent space (*i.e.*, f_θ, g_ω) and their respective decoders (*i.e.* f_ϕ^{-1}, g_ψ^{-1}) to reconstruct vectors back from their latent mapping. On the other hand, Eq. 5.8 trains both encoders and decoders (*i.e.*, $f_\theta, g_\omega, f_\phi^{-1}, g_\psi^{-1}$) to reconstruct vectors back from their image to the other domain. Thus, the Reward-Based Alignment (ReBA) loss is described by Eq. 5.9:

$$\begin{aligned} L_{ReBA} = \lambda_1 \cdot L_A + \lambda_2 \cdot L_G + \lambda_3 \cdot (L_{R_X} + L_{R_Y}) \\ + \lambda_4 \cdot (L_{C_X} + L_{C_Y}) + \lambda_5 \cdot (\|f_\theta(x)\|_2 + \|g_\omega(y)\|_2) \end{aligned} \tag{5.9}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$ are constant coefficients that weight how important each loss term is during training, while $\|f_\theta(x)\|_2 + \|g_\omega(y)\|_2$ is the L2-norm of the latent vectors, used to promote small-magnitude latent vectors and enhance the stability of the training process. From Eq. 5.9, the alignment loss (L_A) and geometry-preserving loss (L_G) are original contributions, whereas the reconstruction loss (L_{R_X}, L_{R_Y}) and cycle-consistency loss (L_{C_X}, L_{C_Y}) have been widely used before in the auto-encoders literature before [Zhao et al., 2017, Hoffman et al., 2018, Dwibedi et al., 2019, Bank et al., 2023].

5.1.1 Alignment Learning

To align the state-action spaces of two RL domains, the alignment of the pair of state spaces is performed independently of the alignment of the actions spaces. In each of these processes, two encoders and two decoders are trained to minimize Eq. 5.9 (see in Fig. 5.2). Regarding the training data, let $D_X = \{(s_t^x, a_t^x, r_t^x)|(s_t^x, a_t^x, r_t^x, s_{t+1}^x) \sim MDP_X\}$, $D_Y = \{(s_t^y, a_t^y, r_t^y)|(s_t^y, a_t^y, r_t^y, s_{t+1}^y) \sim MDP_Y\}$ be a pair of normalized data sets sampled from MDP_X and MDP_Y, where $r_t^x \in \mathbb{R}$ is the immediate reward observed after executing action $a_t^x \in \mathbb{R}^p$ from state $s_t^x \in \mathbb{R}^q$ in MDP_X, and the same meaning applies for $s_t^y \in \mathbb{R}^m, a_t^y \in \mathbb{R}^n, r_t^y \in \mathbb{R}$ in MDP_Y. Then, the data sets used to train alignment of the state and action spaces (D_{Sx}, D_{Sy} and D_{Ax}, D_{Ay} respectively) are defined as follows:

$$
\begin{aligned}
D_{Sx} &= \{(x_i, r^{x_i}) \mid x_i = s_i^x, r^{x_i} = r_i^x, (s_i^x, a_i^x, r_i^x) \in D_X, i = 1, ..., |D_X|\} \\
D_{Sy} &= \{(y_j, r^{y_j}) \mid y_j = s_j^y, r^{y_j} = r_j^y, (s_j^y, a_j^y, r_j^y) \in D_Y, j = 1, ..., |D_Y|\} \\
D_{Ax} &= \{(x_i, r^{x_i}) \mid x_i = a_i^x, r^{x_i} = r_i^x, (s_i^x, a_i^x, r_i^x) \in D_X, i = 1, ..., |D_X|\} \\
D_{Ay} &= \{(y_j, r^{y_j}) \mid y_j = a_j^y, r^{y_j} = r_j^y, (s_j^y, a_j^y, r_j^y) \in D_Y, j = 1, ..., |D_Y|\}
\end{aligned}
$$

That is, although the mapping functions that align the state and action spaces are trained separately, the reward associated with each state-action pair is the same in both processes.

5.1.2 Similarity-based Data Pair Matching

Computing the alignment loss function, as described in Eq. 5.9, requires evaluating $|D_x| \cdot |D_y|$ instances for each learning update of the encoder models. This can become a practical limitation when using large data sets. Thus, to make the learning method more flexible in terms of computational processing time, we present an approximation of Eq. 5.2 in Eq. 5.10:

$$
L_A \approxeq \tilde{L_A} = \sum_{(x,r^x,y,r^y) \in D(D_x, D_y)} -Cos(f_\theta(x), g_\omega(y))\, W(r^x, r^y) \tag{5.10}
$$

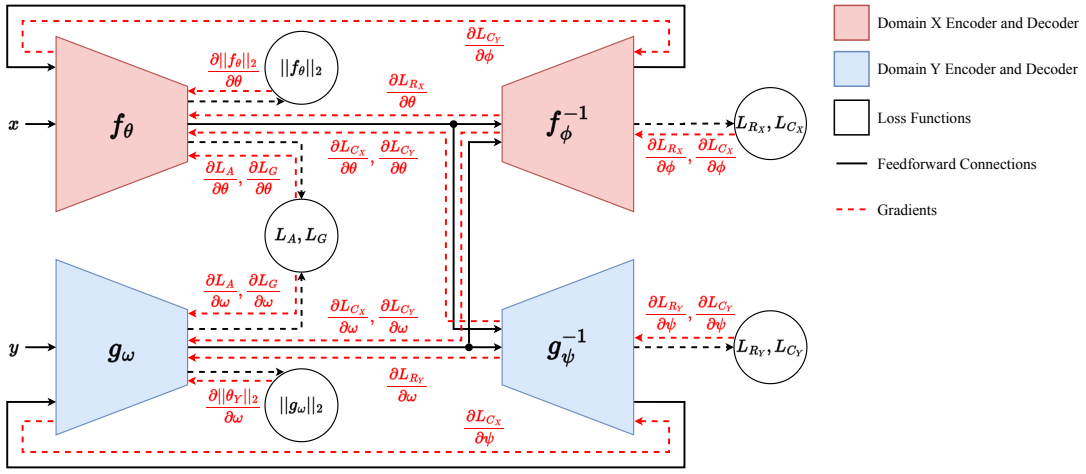


Figure 5.2: The pair of encoder neural networks (f_θ, g_ω) are trained to align the input spaces (X, Y) with the minimization of Eq. 5.9, as well as the regularization terms. On the other hand, the reconstruction and cycle-consistency losses encourage both pairs of encoder-decoders $(f_\theta, f_\phi^{-1}$ and $g_\omega, g_\psi^{-1})$ to retrieve the inputs (x, y) back from the latent space (*i.e.,* Eq. 5.7) and from the other domain (*i.e.,* Eq. 5.8), respectively.

$$D(D_x, D_y) = \{(x, r^x, y, r^y) \mid \forall (x, r^x) \in D_x, (y, r^y) = \underset{(y_i, r^{y_i}) \in D_y}{\arg\max}\ W(r^x, r^{y_i})\} \cup$$
$$\{(x, r^x, y, r^y) \mid \forall (y, r^y) \in D_y, (x, r^x) = \underset{(x_i, r^{x_i}) \in D_x}{\arg\max}\ W(r^{x_i}, r^y)\}$$

where $W(r^x, r^y)$ is the similarity coefficient (Eq. 5.3) and $D(\cdot, \cdot)$ a function that returns a subset of the training data set $D_x \times D_y$, in which every data point $x \in D_x$ is paired with the data point $y \in D_y$ that has the most similar reward r^y to that of x (*i.e.*, r^x). And vice versa, every data point $y \in D_y$ is paired to the data point $x \in D_x$ with the most similar reward. By pairing points by checking their most similar match (based on their rewards), the inclusion of every data point is ensured.

That is, since the alignment loss function returns the largest values when *very similar* data pairs are mapped near to each other in the latent space, the approximation in Eq. 5.10 prioritizes evaluating only the closest pair for every data point in the original data sets.

5.1.3 Geometry Preserving Neighborhood Selection

In addition to the approximation introduced in Section 5.1.2, the geometry preserving loss function is also approximated to alleviate the computational burden of

computing it in its original form (see Eq. 5.4). Instead of evaluating the distance in the latent space of every data point mapped from the same domain, each instance is evaluated against its k-nearest neighbors, as described in Eq. 5.11:

$$L_G \approxeq \tilde{L_G} = \sum_{(x_i,r^{x_i})\in D_x} \sum_{(x_j,r^{x_j})\in K(x_i,D_x,k)} \frac{[Cos_d(x_i,x_j) - Cos(f_\theta(x_i), f_\theta(x_j))]^2}{|D_x|\cdot k} + \sum_{(y_i,r^{y_i})\in D_y} \sum_{(y_j,r^{y_j})\in K(y_i,D_y,k)} \frac{[Cos_d(y_i,y_j) - Cos(g_\omega(y_i), g_\omega(y_j))]^2}{|D_y|\cdot k} \tag{5.11}$$

where $Cos_d(\cdot,\cdot)$ is defined in Eq. 5.5, k the number of neighbors to include in each data point's geometry preserving computation, and $K(x, D, k)$ a function that returns the k nearest neighbors [Fix, 1985] (using the Euclidean distance) of a vector x from data set D.

5.2 Dynamics-based Inter-Task Similarity Measure

Once the mapping functions that align the state and action spaces of a pair of tasks have been learned, they are used to evaluate the similarity between those tasks (under the assumption that their reward functions are dense, rich and normalized). Let D_X the data set recollected while learning a policy π_X for the source task $T_X = \langle S_X, A_X, D_X, \pi_X\rangle$, D_Y be a data set sampled by interacting with the environment E_Y from the target task $T_Y = \langle S_Y, A_Y, E_Y\rangle$, and (f_S, f_A, g_S, g_A), $(f_S^{-1}, f_A^{-1}, g_S^{-1}, g_A^{-1})$ the encoders and decoders that align their state and action spaces, respectively. The following approximate models are trained (with data sets D_X, D_Y) to predict the immediate reward in the source and target task:

$$\hat{R_X} : S_X \times A_X \to \mathbb{R}$$
$$\hat{R_Y} : S_Y \times A_Y \to \mathbb{R}$$

whereas $\hat{\Phi}_X : S_X \times A_X \to S_X$ is trained with D_X to predict the next state in the source task. Then, the inter-task similarity between tasks T_X and T_Y is given by

Eq. 5.12:

$$
\begin{aligned}
Sim(T_X, T_Y) = 1 - & \sum_{s_y, a_y, s'_y, r_y \in D_Y} \sum_{a'_x \in A_x, a'_y \in A_y} \frac{|\hat{R}_X(s'_x, a'_x) - \hat{R}_Y(s'_y, a'_y)|}{|D_Y| \cdot |A_x|} \\
& s'_x = \hat{\Phi}_X(f_S^{-1} \circ g_S(s_y), f_A^{-1} \circ g_A(a_y)) \\
& A_y \sim U(A_Y) \\
& A_x = f_A^{-1} \circ g_A(A_y)
\end{aligned}
\tag{5.12}
$$

where s'_x is the next-state prediction according to domain X's transition model, given a state and action mapped from domain Y, A_y a set of domain Y actions sampled from a uniform distribution $U(\cdot)$, and A_x their mapping to domain X's action space. In other words, Eq. 5.12 evaluates how similar the reward distributions are between the states reached from equivalent state-action pairs, according to the learned mapping functions (see Fig. 5.3).

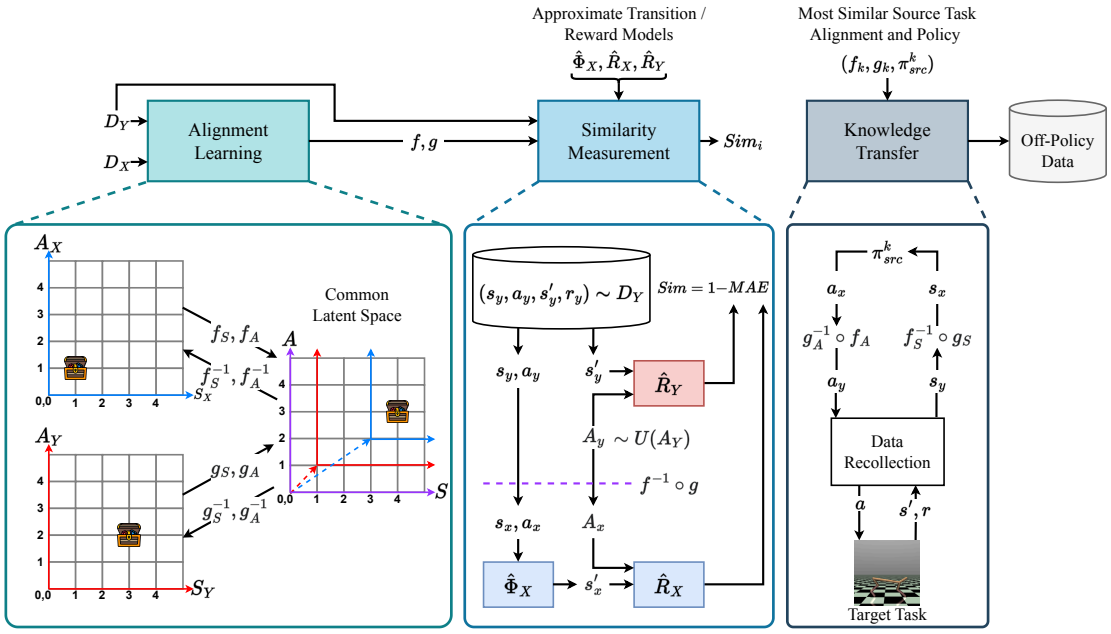


Figure 5.3: Interactions between the knowledge transfer method main stages. Once learned, the alignment mapping functions (f, g), target task data set (D_Y) and reward/transition approximation models $(\hat{\Phi}_X, \hat{R}_X, \hat{R}_Y)$ are used to compute the inter-task similarity. Finally, an off-policy data set (*i.e.,* data that will be used for learning purposes by a policy different than the one that generated the data) is sampled from the target domain with the policy from the most similar source task, by mapping target states (to the source domain) so they are fed to the source policy π^k_{src} and mapping the selected action back to the target domain.

5.3 Similarity-based Knowledge Transfer

To transfer knoweldge between tasks from different domains, the **Sim**ilarity-based **Kno**wledge **T**ransfer (SimKnoT) method is presented in Algorithm 1.

Algorithm 1 starts by performing standard RL exploration in the target task for a fixed number of steps and stores the sampled data in a buffer B (lines 1-5). Then, the sampled data is used to learn the alignment mapping functions, and measure similarity, between the target task and each source task, with their respective data sets (lines 6-11). A data set is sampled (line 12), in the target domain, with the policy from the most similar source task with Algorithm 2 and stored in buffer B. In order to assimilate the knowledge transferred by the source policy, the target policy updates its parameters for a fixed number of steps, without adding new data to the buffer B. Finally, the target policy finishes its training process following a standard

RL setting.

Algorithm 1 $SimKnoT$: Similarity-based Knowledge Transfer

Input: $T_{src} = \{T_{src}^1, ..., T_{src}^N\}, T_{tgt} = \langle S_{tgt}, A_{tgt}, E_{tgt} \rangle$
Output: π_{tgt}

▷ Train in RL setting and sample data
$D \leftarrow \{\}$, $B \leftarrow \{\}$, $\pi_{tgt} \leftarrow InitPolicy()$
for $i \leftarrow 1$ to I_{preTL} **do**
    Sample transition (s, a, s', r) with (π_{tgt}, E_{tgt}) and add it to B and D
    Optimize RL objective of π_{tgt} with respect to B
end for

▷ Learn alignment and measure similarity
$Sim \leftarrow [\,]$, $Models \leftarrow [\,]$
for $i \leftarrow 1$ to N **do**
    Optimize Eq. 5.9 with $(D, GetData(T_{src}^i))$ and add the alignment models to $Models$
    Compute similarity (Eq. 5.12) with $(D, GetData(T_{src}^i))$ and add the similarity score to Sim
end for
Get index of most similar task from Sim and store it in $simID$

▷ Sample data with transferred policy and update target policy
π_{tgt}, $B \leftarrow TransferPolicy(T_{src}, Models, simID, \pi_{tgt}, B, E_{tgt})$

▷ Update π_{tgt} without adding data to the replay buffer
for $i \leftarrow 1$ to E **do**
    Optimize RL objective of π_{tgt} with respect to B
end for

▷ Train in RL setting
for $i \leftarrow 1$ to I_{posTL} **do**
    Sample transitions (s, a, s', r) with (π_{tgt}, E_{tgt}) and add it to B
    Optimize RL objective of π_{tgt} with respect to B
end for
return π_{tgt}

Algorithm 2 Source Policy Transfer to Target Domain

Input: $T_{src}, Models, simID, \pi_{tgt}, B, E_{tgt}$
Output: π_{tgt}, B
function TRANSFERPOLICY($T_{src}, Models, simID, \pi_{tgt}, B, E_{tgt}$)
▷ Source policy and alignment models
$\pi_{src} \leftarrow GetPolicy(T_{src}^{simID})$
$\phi_{src}^{S}, \theta_{tgt}^{S}, \phi_{tgt}^{A}, \theta_{src}^{A} \leftarrow GetAlignmentModels(Models[simID])$

▷ Perform off-policy RL with source policy
for $i \leftarrow 1$ to I_{TL} **do**
Sample (s, a_{tgt}, s', r) from E_{tgt}, where $a_{src} = \pi_{src}(\phi_{src}^{S} \circ \theta_{tgt}^{S}(s_{tgt}))$ $a_{tgt} = \phi_{tgt}^{A} \circ \theta_{src}^{A}(a_{src})$, and add it to B
Optimize RL objective of π_{tgt} with respect to B
end for
return π_{tgt}, B
end function

5.4 Experiments

5.4.1 Experimental Setting

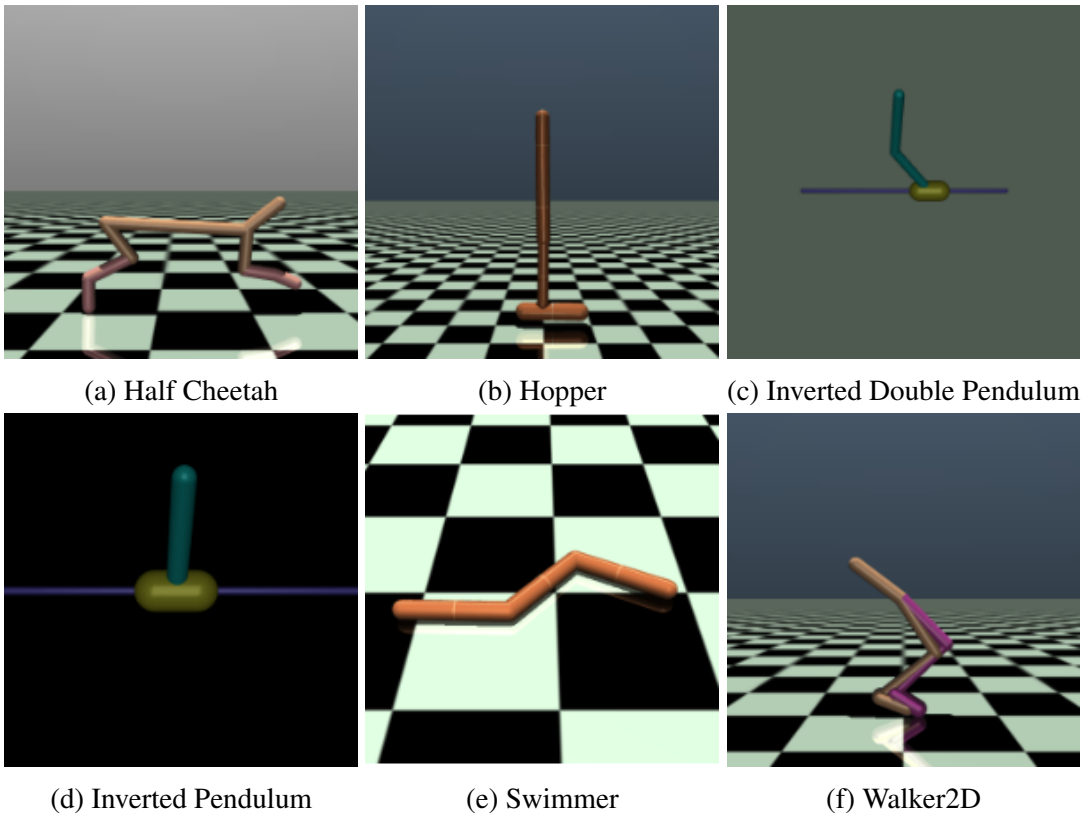
The objective of the present experiments is to evaluate the SimKnoT algorithm regarding the following questions:

1. Can the similarity measure identify inter-task similarity in the cross-domain setting?
2. Can SimKnoT perform positive transfer among tasks with no semantic relation?
3. Does the transferred knowledge impact the target learner's performance?

To evaluate SimKnoT, six control-based tasks with different state-action spaces, and significant morphological differences among some tasks, were selected from the Gymnasium Suite [Todorov et al., 2012, Towers et al., 2023]: Half cheetah, hopper, inverted double pendulum, inverted pendulum, swimmer and walker-2D (see Fig. 5.4). The purpose of using a set of evaluation tasks with considerably different morphology is to test the ability of SimKnoT to autonomously select a source of knowledge that will benefit the target learner, despite some of the available options may hurt the target agent if selected (*i.e.*, avoid negative transfer). Moreover, given that the problem setting addressed by this research is focused on tasks with rich/dense reward functions (see Section 1.6), the original reward function in the inverted pendulum task (which is sparse), was replaced by the function described by Eq. 5.13:

$$R_{IP\,Dense}(\theta, \dot{\theta}) = C_{Alive} - 0.01 \cdot sin(\theta)^2 - (cos(\theta) - 1)^2 - 0.005 \cdot \dot{\theta}^2 \tag{5.13}$$

where C_{Alive} is a bonus for staying alive, the sin, cos terms penalize the angular distance between the pole and the perfect vertical position, whereas the last term penalizes the angular velocity of the pole. The value of the C_{Alive} constant is the same as in the inverted double pendulum reward.



(a) Half Cheetah (b) Hopper (c) Inverted Double Pendulum

(d) Inverted Pendulum (e) Swimmer (f) Walker2D

Figure 5.4: Set of evaluation tasks: Half Cheetah, Hopper, Inverted Double Pendulum (IDP), Inverted Pendulum (IP), Swimmer and Walker2D.

To contextualize the proposed knowledge transfer method, we use the Soft-Actor Critic (SAC) algorithm [Haarnoja et al., 2018] (implemented by MushroomRL [D'Eramo et al., 2021]), and a modification of SAC in which the agent optimizes with respect the current data in the replay buffer, without adding new data, for multiple iterations (as SimKnoT does in Algorithm 1 in lines 13-15), which we refer as SAC with fixed-buffer optimization (SAC+FBO). To evaluate the impact of the selected source task, we evaluate two variations of SimKnoT: SimKnoT MSS and SimKnoT LSS, which select the most similar source and the least similar source, respectively.

Additionally, we compare SimKnoT to our implementation of the Adaptive Policy Gradient Transfer (APGT) method proposed in [Zhang et al., 2024]. The APGT method, which is based on the proximal policy optimization (PPO) algorithm [Schulman et al., 2017], uses target-to-source state mappings (provided by the user) to feed states from the target task to the critic (*i.e.*, state value estimator) of each source agent to measure similarity and transfer knowledge to the target agent (see

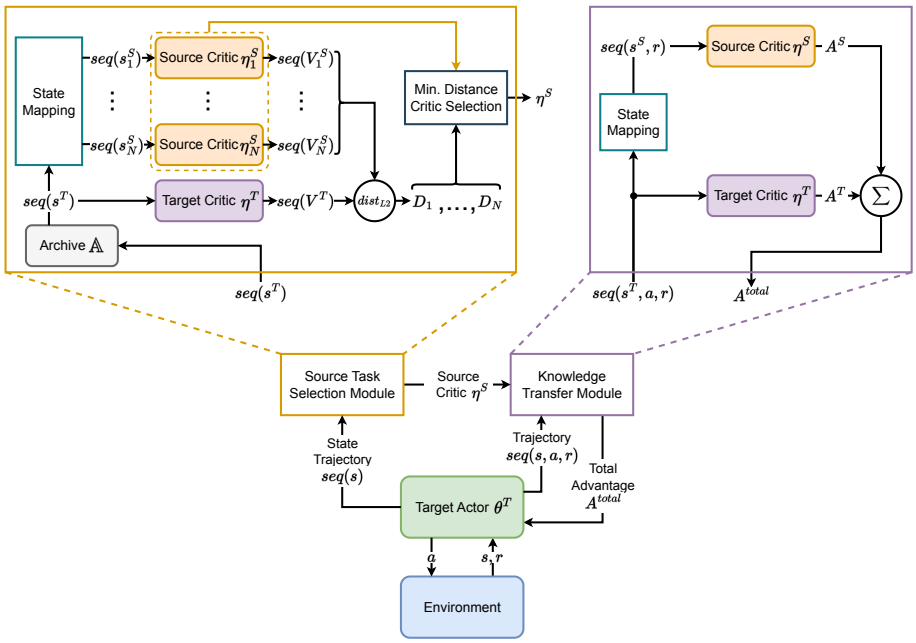


Figure 5.5: The APGT method proposed in [Zhang et al., 2024] employs source task selection module to identify the most similar source task based on the value of a sequence of states $seq(s^T)$, contained the archive $\mathbb{A}$ (a FIFO list), which are mapped to the state space of each source task state space. The source critic that produced the sequence of state-values with the lowest Euclidean distance to the sequence of state values produced by the target critic $seq(V^T)$ is selected as the most similar source task. Then, in the knowledge transfer module, the advantage yielded by the source critic A^S is added to the one produced by the target critic A^T to bias the target actor's θ^T training process. Throughout the training process the archive $\mathbb{A}$ is updated with the latest trajectories observed by the target policy, and the similarity measurement and knowledge transfer processes are repeated.

Fig. 5.5 for an overview of APGT). We used the same parameters as in the original publication [Zhang et al., 2024]. For the state mappings between each task, we hand coded mappings based on the physical similarity between variables (*e.g.*, mapping the knee angle of the leg in Hopper to both knee angles in Walker2D). Also, in each knowledge transfer experiment, APGT has access to all source tasks, including the same task as in the target domain, whose state variables are mapped by the identity function.

Also, the average performance of a random policy and a policy trained from scratch for 10 million environment steps, which serve as bottom and top performance references.

In each experiment, the source task policies have been trained for 1 million

(hopper, inverted pendulum, inverted double pendulum) and 3 million (half cheetah, swimmer, walker-2D) environment steps, resulting in a trained policy and a data set of $(state, action, nextstate, reward)$ observations, which is used to train the transition and reward approximation models (*i.e.,* $\hat{\phi}, \hat{R_X}$ from Section 5.2). On the other hand, in the target task SAC, SAC+FBO, SimKnoT LSS/MSS (SAC as base algorithm) and APGT are trained for 950,000 environment steps. Within the training process, SAC+FBO and SimKnoT perform certain processes that SAC does not:

- **Initial Sampling Period**: SimKnoT uses the observations gathered during the first environment steps (10,000 in IP, IDP and 50,000 in te other tasks) to train the approximated reward model ($\hat{R_Y}$ in Section 5.2), learn the alignment functions and measure the similarity between the target domain and each source domain.
- **Knowledge Transfer Period**: SimKnoT transfers actions drawn from the most similar source task policy to interact with the target environment for the next 10,000 environment steps in IP, IDP, 100,000 environment steps in H. Cheetah and 25,000 environment steps in the other tasks.
- **Fixed-Buffer Optimization**: At the end of the knowledge transfer period, SAC+FBO and SimKnoT perform 4,000 learning updates with respect the current data in the replay buffer.

To reduce the time required to train the alignment models, the approximations presented in Sections 5.1.2 and 5.1.3 are incorporated in the original loss to produce the approximated ReBA loss in Eq. 5.14:

$$\begin{aligned}\tilde{L}_{ReBA} = \lambda_1 \cdot \tilde{L_A} + \lambda_2 \cdot \tilde{L_G} + \lambda_3 \cdot (L_{R_X} + L_{R_Y}) \\ + \lambda_4 \cdot (L_{C_X} + L_{C_Y}) + \lambda_5 \cdot (\|f_\theta(x)\|_2 + \|g_\omega(y)\|_2)\end{aligned} \tag{5.14}$$

where $\tilde{L_A}$ and $\tilde{L_G}$ represent the approximations presented in Eq. 5.10 and Eq. 5.11, respectively. Thus, for every experiment, Eq. 5.9 is replaced by Eq. 5.14 in line 8 of Algorithm 1. For more details on the architectures and parameters used to train the alignment models, reward/transition approximated models and policies, see appendix A.

5.4.2 Results

Figure 5.6 shows the similarity scores obtained after computing Eq. 5.12 between each pair of source (column) and target (row) task where two tasks (*i.e.,* inverted pendulum, swimmer) out of six were successfully identified as the most similar with their source twin. Moreover, although the remaining target tasks (*i.e.,* half cheetah, hopper, walker-2D, inverted double pendulum) were matched with different task, their source counterpart was ranked as the second most similar, as shown in the main diagonal in Fig. 5.6. However, it is worth noting that given that SimKnoT does not have access to the task IDs/labels (see the research scope and restrictions in Section 1.6) the matching source and target tasks is done based on the limited amount of data sampled while training.

In relation to the knowledge transfer results, Fig. 5.7 shows the performance of SimKnoT MSS, SimKnoT LSS, SAC, SAC+FBO and APGT in each target task, averaged over 5 independent trials and smoothed with a uniform and symmetrical centered window of length 15. Similarly, Fig. 5.9 shows the results of transferring knowledge with SimKnoT MSS to IP and IDP with 50,000 and 100,000 environment steps in the initial sampling period and the knowledge transfer period, respectively. Table 5.1 shows the periods (within the training process) during which a method had the top performance in the target task and also had a statistically significant higher performance than the other methods, with a confidence of 0.95 (*i.e.*, $p\text{-}value = 0.05$), using the t-test due to the small samples.

5.4.3 Discussion

Regarding the first question (*i.e., Can the similarity measure identify inter-task similarity in the cross-domain setting?*), the ability to pair *similar* tasks together is assessed in terms of how these pairings promote positive transfer. Even though the similarity measure described in Section 5.2 did not pair every target task with its source copy, using the most similar task for each target task allowed SimKnoT MSS to outperform SAC (*i.e.,* learning from scratch) in five out of six transfer experiments. That is, given most of the transfer experiments resulted in positive transfer, it is safe to say that the similarity measure can detect inter-task similarity for knowledge transfer purposes. Moreover, by comparing the performance of SimKnoT MSS

Table 5.1: Uninterrupted periods of highest performance and statistically significant superiority (with a $p\text{-}value = 0.05$) of a method over the others. The score on the left side of the forward slash is the average performance in the period specified by the interval on the right side of the slash. For methods that were never the top performer on a task (with statistical significance) a hyphen is shown (*i.e.*, $-$). The period with the highest performance in each target task is in bold font.

Method	Cumulative Reward $\uparrow^+$ / Start Env. Steps $\times 10^3$ $\downarrow^+$, End Env. Steps $\times 10^3$					
	H. Cheetah	Hopper	IP	Swimmer	Walker2D	IDP
SAC	–	–	–	–	–	9351.5/ 780, 781
SAC+FBO	35.2/ 90, 91	182.9/ 45, 46	373.6/ 16, 21	–	235.2/ 50, 53	90.2/ 10, 21
APGT	–	229.3/ 60, 63	–	–	–	–
SimKnoT LSS	-234.8/ 1, 2	–	–	–	–	–
SimKnoT MSS	**6531.8/ 948, 949**	**2014.9/ 944, 945**	**9998.6/ 302, 303**	–	**1194.3/ 933, 937**	**9357.1/ 229, 230**

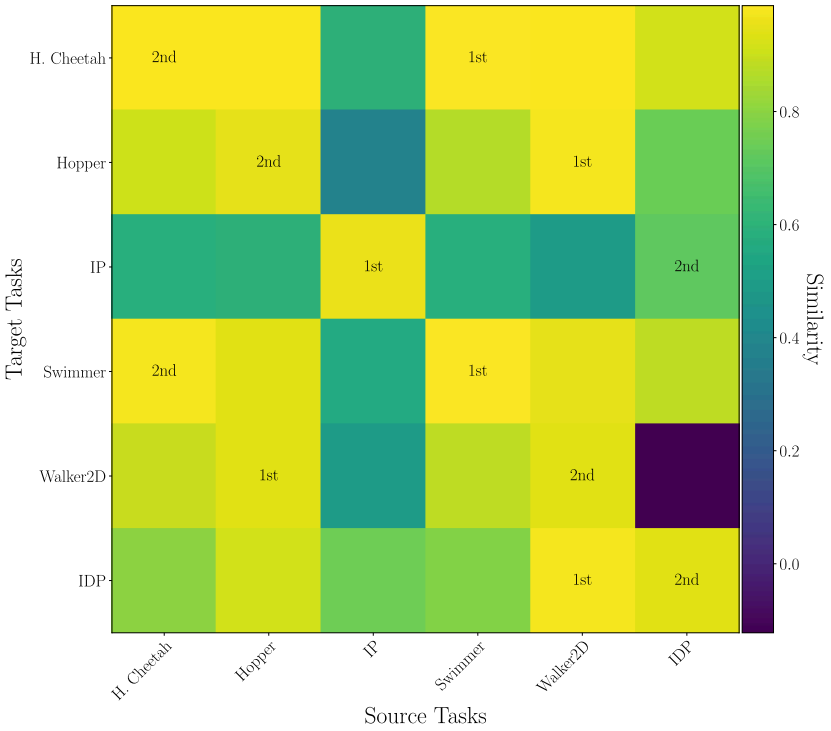


Figure 5.6: Similarity matrix computed with Eq. 5.12, where the higher the value, the more similar two tasks are. Rows and columns represent target and source tasks, respectively. The first and second most similar source task to each target task are specified with a label.

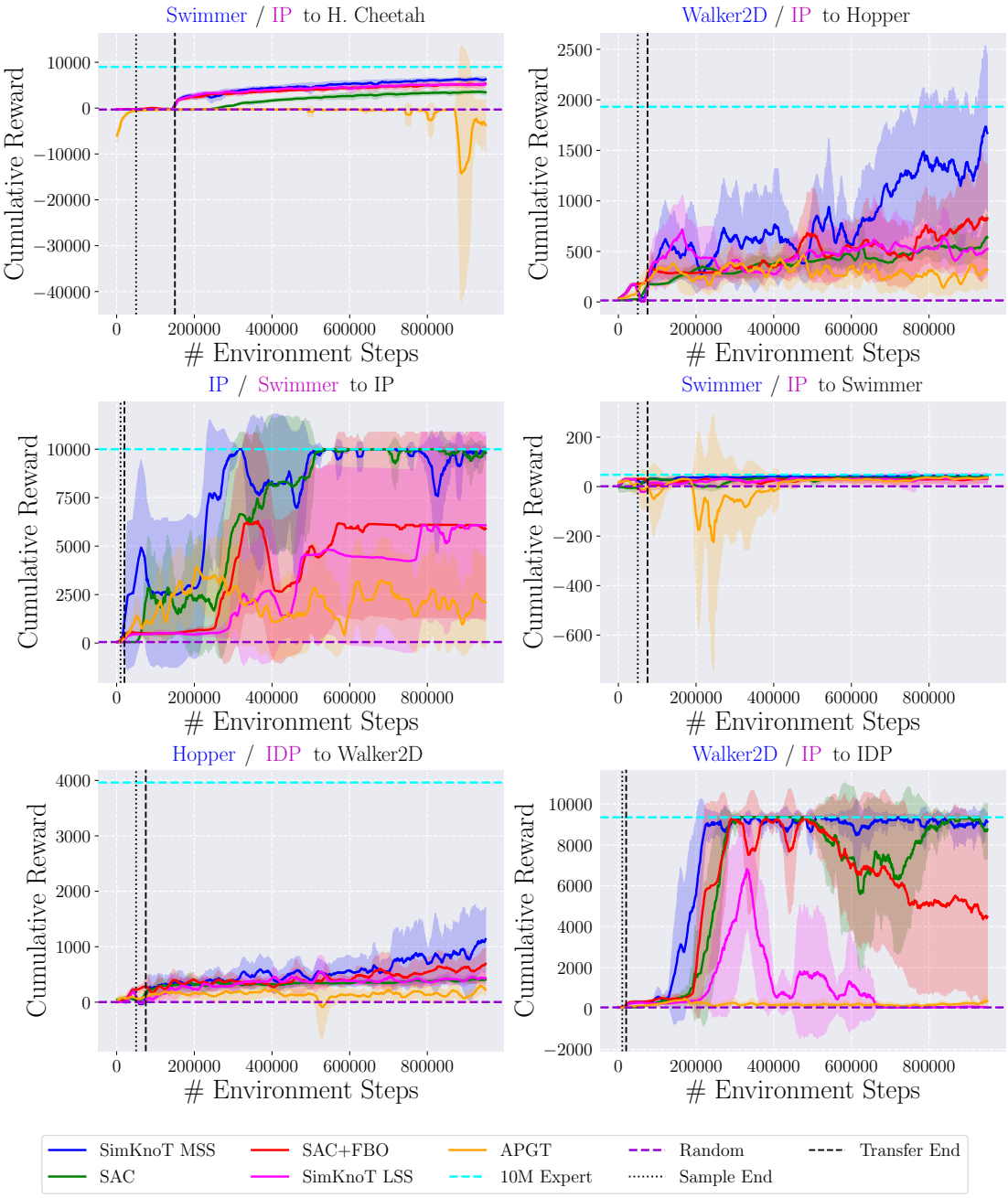


Figure 5.7: Episodic cumulative reward of SimKnoT MSS/LSS, SAC, SAC+FBO and APGT in each target task (averaged over 5 trials). The source task selected by SimKnoT MSS and SimKnoT LSS are shown in blue and magenta in the title of each graph, respectively. The vertical dotted line shows the timestamp at which the initial sampling period ended and the knowledge transfer period started, while the vertical dashed line shows when the knowledge transfer period ended and the fixed-buffer optimization was performed. Top and bottom horizontal lines show the average performance of a policy trained with 10 million observations and a random policy.

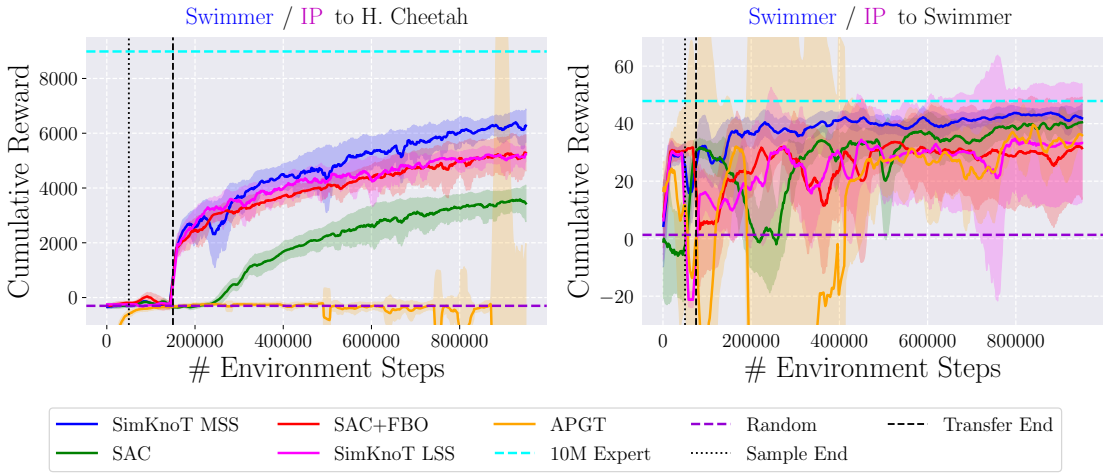


Figure 5.8: Episodic cumulative reward of SimKnoT MSS/LSS, SAC, SAC+FBO and APGT in H. Cheetah and Swimmer (averaged over 5 trials). The source task selected by SimKnoT MSS and SimKnoT LSS are shown in blue and magenta in the title of each graph, respectively. The vertical dotted line shows the timestamp at which the initial sampling period ended and the knowledge transfer period started, while the vertical dashed line shows when the knowledge transfer period ended and the fixed-buffer optimization was performed. Top and bottom horizontal lines show the average performance of a policy trained with 10 million observations and a random policy.

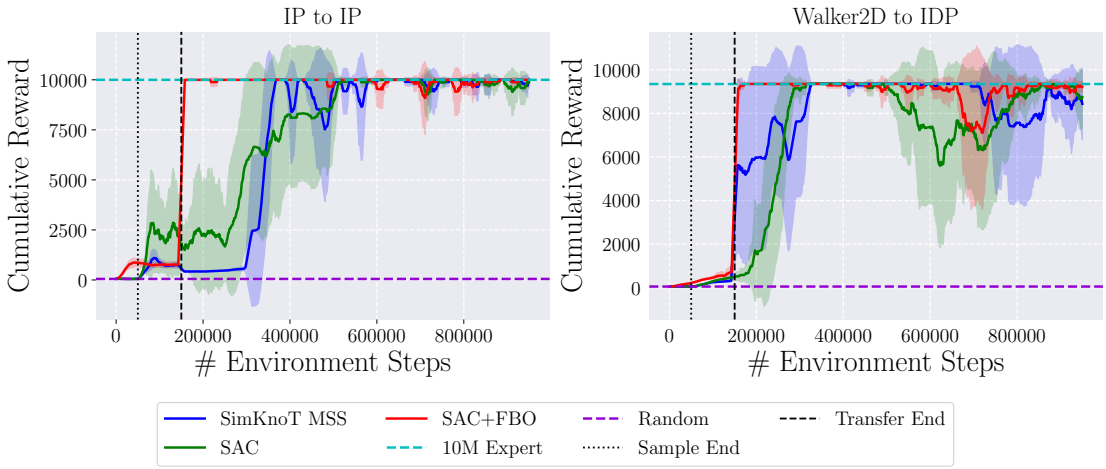


Figure 5.9: Episodic cumulative reward of SimKnoT MSS, SAC and SAC+FBO in IP and IDP (averaged over 5 trials). The vertical dotted line shows the timestamp at which the initial sampling period ended and the knowledge transfer period started, while the vertical dashed line shows when the knowledge transfer period ended and the fixed-buffer optimization was performed. Top and bottom horizontal lines show the average performance of a policy trained with 10 million observations and a random policy.

and SimKnoT LSS, it is safe to say that the selecting the most similar task (according to Eq. 5.12) affects the outcome of transferring knowledge, as by transferring knowledge from the least similar task caused a negative transfer in most target tasks.

Additionally, SimKnoT was able to identify a source task and safely transfer knowledge from it to the target agent, despite the variety of options available. On the other hand, the APGT method [Zhang et al., 2024] was not able to achieve a similar result. Even though APGT had at its disposition a trained agent in the same task as in the target domain, with the identity function mapping state variables from the target task to the source task, the presence of other source tasks affected its ability to ignore sources of negative transfer in every target task.

In relation to the second question (*i.e., Can SimKnoT perform positive transfer among tasks with no semantic relation?*), in addition to transferring knowledge across tasks with shared high-level features (*e.g.,* Hopper and Walker2D), by transferring knowledge from Swimmer (a 3-limb crawling robot) to H. Cheetah (a 9-limb dog like robot) SimKnoT shows its ability to improve the performance of a target learner, despite the lack of discernible commonalities as H. Cheetah not only deals with inertial forces but also with the constant effect of gravity (see the upper-left graph in Fig. 5.7 and left graph in Fig. 5.8).

Although SimKnoT is not the first work to report positive transfer across tasks with significantly different structures (*e.g.,* [Raychaudhuri et al., 2021] successfully transferred knowledge from the Mujoco Ant task to H. Cheetah), to the best of our knowledge, SimKnoT is the first method to achieve it by autonomously choosing the source of knowledge from a widely varied set of sources. With such ability, SimKnoT presents an option to automatically select and transfer knowledge in scenarios where an expert is not available due to the sheer number of tasks that need to be solved, *e.g.,* decreasing the sample complexity of solving every instance in a database of RL tasks [Ramos et al., 2021] by reusing useful data when possible.

Concerning the third question (*i.e., Does the transferred knowledge impact the target learner's performance?*), by comparing SimKnoT MSS to the SAC+FBO baseline we can assess the impact of using the transferred policy to explore the target environment. Considering that Deep RL policies and value functions are modeled by neural networks, one way to improve the model's performance is by simply performing more updates on the model's parameters (*i.e.,* weights in a neural network) with respect to the available data. Therefore, to assess the impact of the data rec-

ollected by SimKnoT MSS in the target task, we compare its performance against SAC+FBO, which is a SAC agent that performs the same number of parameter updates as SimKnoT MSS, at the same instant during the training process.

As shown in Fig. 5.7 and Table 5.1, SimKnoT MSS achieves a larger cumulative reward in fewer environment steps than SAC+FBO in five out of the six knowledge transfer experiments (*i.e.,* H. Cheetah, Hopper, IP, IDP and Walker2D). This result suggests that the increased number of learning updates is not the only process responsible for outperforming the SAC baseline, but also the data used in the additional learning updates.

On the other hand, the duration of the initial sampling period and knowledge transfer period play a critical role in the success of SimKnoT. As shown in Fig. 5.9, by sampling data and transferring knowledge at a later stage in the training process, the SAC+FBO outperforms SimKnoT MSS in the IP and IDP tasks. We suspect that in tasks with simpler dynamics, small sampling and transferring periods provide a better way to improve th RL base agent, while in more complex tasks longer periods will be necessary as more data will be likely required to train a good approximation of the reward and transition models (which are used to measure inter-task similarity).

Moreover, another critical aspect in the success of SimKnoT is that tasks have a dense and rich reward function. To show the dependence of SimKnoT on rich reward models we created a copy of each evaluation task with a sparse reward function. In the case of H. Cheetah, Hopper, Swimmer and Walker2D a +1 is observed if the agents moves forward, 0 otherwise. For IP and IDP, a +1 is observed if the (upper) pole is standing upright (within a certain angle limit), 0 otherwise. Figure 5.10 shows the similarity scores between IP and every task (including itself), under the same setting described in Section 5.4.1 (except the reward function). These results show how, in the sparse setting, IP no longer is matched with itself as the most similar, but IDP is ranked as the least similar task by a wide margin, despite it is the second most similar source task in the dense reward setting.

Additionally, to evaluate the contribution made by each one of the proposed loss terms (*i.e.,* $\tilde{L}_A$ in Eq. 5.10 and $\tilde{L}_G$ in Eq. 5.11), in Table 5.2 the similarity scores between two instances of the Inverted Double Pendulum task are reported for the proposed ReBA loss (see Eq. 5.14) and multiple variations. In each experiment, the source and target data sets had a size of 1 million and 50,000 observations, respectively. It is worth noting that the similarity score slightly declines if the alignment or

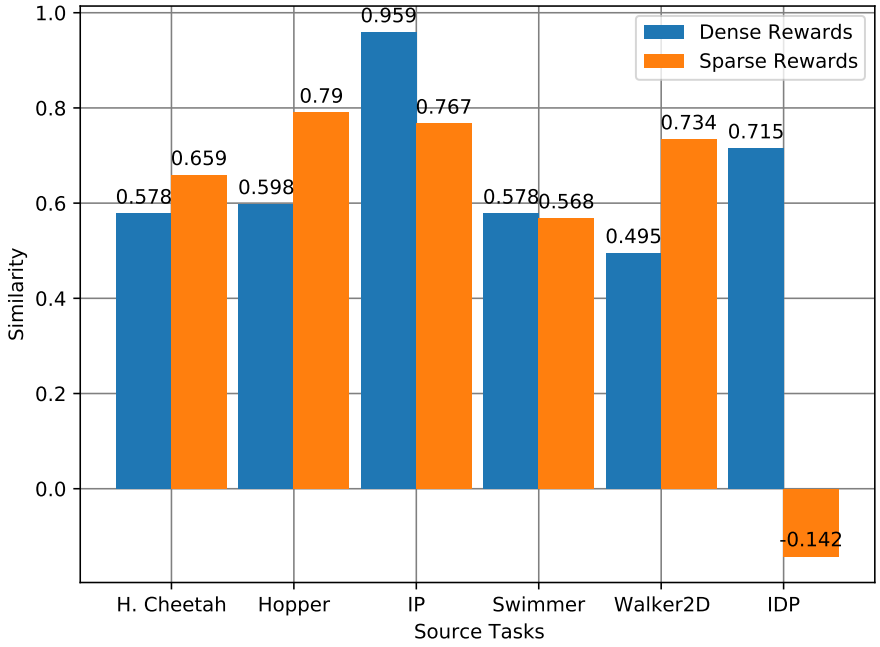


Figure 5.10: Similarity scores between the IP target task and every source task when both the target and source tasks have a dense/rich (blue) and sparse (orange) reward.

geometry preserving terms are removed from ReBA. However, if both terms are excluded (*i.e.,* leaving the regularization, reconstruction and cycle-consistency losses to train the alignment mapping functions), then the similarity score between two identical tasks plummets. Thus, it is safe to argue that both terms play an essential role in the unsupervised selection of source tasks, which in turn is the cornerstone of SimKnoT.

Table 5.2: Comparison of the alignment training setting (with IDP as source and target task) used in the transfer experiments (first row) and multiple variations of ReBA without the alignment (w/o A) and geometry preserving (w/o GP) terms. The rightmost column shows the similarity score as defined in Eq. 5.12.

Loss	Similarity
ReBA	0.9401
ReBA w/o A	0.9024
ReBA w/o GP	0.9305
ReBA w/o A, GP	-0.2037

5.5 Chapter Summary

In this chapter, a knowledge transfer method based on inter-task similarity was presented. Using the immediate reward of two tasks as alignment criterion, SimKnoT is able to match state-action spaces in a way that allows both selecting and transferring knowledge across different domains. Although SimKnoT is not the first method to achieve positive transfer in the cross-domain setting, to the best of our knowledge it is the first method to achieve it when a varied set of source tasks is available.

Chapter 6

Lifelong Cross-Domain Reinforcement Learning

In lifelong learning settings, the knowledge that is acquired after learning in a sequence of tasks is stored for two main reasons:

1. To remember how to solve previous tasks (*i.e.,* avoid catastrophic forgetting), and

2. To learn future tasks faster (in comparison than learning from scratch) by reusing the stored knowledge

Given that the lifelong learning agent does not know how many tasks it must learn, nor the order in which tasks will be presented to the agent, it is crucial to have knowledge consolidation and transfer methods capable of producing positive transfer without forgetting older knowledge. Thus, in this chapter, a lifelong learning agent is presented for the cross-domain reinforcement learning setting. Section 6.1 describes a lifelong learning system based on the knowledge transfer method introduced in Chapter 5, whereas Section 6.2 presents an experimental evaluation of the source knowledge selection and transfer abilities of the proposed lifelong learning system.

6.1 Method Description

In order to address the lifelong reinforcement learning setting, we propose adapting the SimKnoT method (see Section 5.3), that originally works with a fixed set of source tasks, to progressively add the policy and training data, acquired from learning in the latest task, to its set of available source tasks, as shown in Fig. 6.1. Because SimKnoT is a knowledge transfer method that autonomously selects what it should transfer from a set of options to the target task, it is well suited to be evaluated in a sequential setting such as lifelong reinforcement learning.

Considering that the SimKnoT-based lifelong learning agent consolidates knowledge in the most naive way (*i.e.,* storing the policy and data without performing a transformation to them), one would suspect that catastrophic forgetting (*i.e.,* forgetting how to solve previous tasks) does not represent a threat, as every piece of knowledge has been stored intact in the knowledge base. However, given that tasks are not labeled (*i.e.,* the system can not determine if the current task has already been stored in its knowledge base), being able to find a source task among its continually growing knowledge base represents a challenge, even if knowledge is stored in its original form. On the other hand, by consolidating knowledge without performing any type of compression, the SimKnoT-based lifelong learning agent may not be a feasible solution for large sequences of tasks, since the training data set must be saved in the knowledge base for each task in the sequence.

Thus, in Section 6.2 we present a set of experiment that evaluate the performance of the SimKnoT-based lifelong reinforcement learning system.

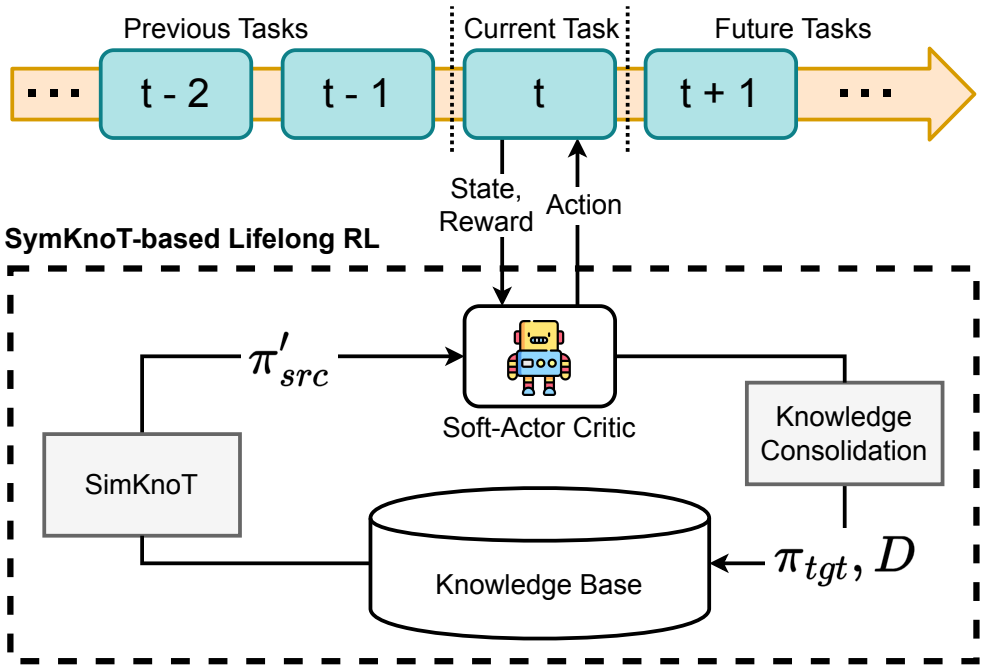


Figure 6.1: Lifelong reinforcement learning (LRL) system based on the SimKnoT knowledge transfer method (see Section 5.3). Every time the SAC algorithm finishes learning a task, the learned policy (π_{tgt}) and training data (D) are stored in the knowledge base. Then, when the next task is presented to the system, SimKnoT transfers the policy adapted from the most similar source task (π'_{src}) among all the tasks stored in the continually updated knowledge base.

6.2 Experiments

6.2.1 Experimental Setting

The objective of the present experiments is to evaluate the SimKnoT-based lifelong reinforcement learning agent in the following aspects:

1. Can the agent select source tasks that help learning faster than learning from scratch in the agent's current task, as the number of tasks in the knowledge base grows?

2. How does the agent perform in sequences of tasks of unknown order?

To evaluate the SimKnoT-based lifelong learning agent, the same set of six tasks used for the evaluation of SimKnoT will be used (see Fig. 5.4): H. Cheetah,

Hopper, Inverted Double Pendulum, Inverted Pendulum, Swimmer and Walker2D. Additionally, the same set of parameters used for the knowledge transfer evaluation of SimKnoT in Section 5.4.1 are used in these experiments.

Since the source-selection method used by SimKnoT (*i.e.*, the similarity function from Section 5.2) does not have a way to detect if none of the available source tasks is similar enough to improve the target task learner, in each experiment the lifelong learning agent starts with a knowledge base that contains six source tasks, one instance of each evaluation task (see Fig. 5.4). Then, a sequence of 12 tasks (two instances per each evaluation task) will be fed to the agent in an unknown order. The goal of the experiment is to evaluate which source task is selected as the most similar to each target task.

After the agent selects the most similar source task to the current task, the policy and data associated to learning the target task are added to the knowledge base. However, the alignment models (trained with ReBA), the policy and data of every source task have been learned/sampled *a priori*. Only the data used sampled in the target task to measure the similarity with each source task is sampled online with a SAC agent. A total number of 100 randomly generated permutations are employed, and the number of times each source task is ranked as the most similar to each target task is reported.

On the other hand, to evaluate the knowledge transferring performance of the SimKnoT-based LRL agent, 10 sequences of length 6 in which each evaluation tasks appears exactly once, were randomly generated and presented to the LRL agent for it learn in each task. The performance of the LRL agent was measured by the total reward (*i.e.*, area under the curve) ratio (see Eq. 6.1), as presented in [Taylor and Stone, 2009a]:

$$r = \frac{total\ reward\ with\ transfer - total\ reward\ without\ transfer}{total\ reward\ without\ transfer} \tag{6.1}$$

where the total reward with transfer corresponds to SimKnoT MSS's performance, while the total reward without transfer corresponds to SAC's performance. Thus, values above and below 0.0 for Eq. 6.1 indicate that positive and negative transfer was produced, respectively. To evaluate the performance of the LRL agent in a sequence of tasks, the sum of the total reward ratios obtained in each task.

6.2.2 Results and Discussion

Regarding the first question (*i.e., Can the agent select source tasks that help learning faster than learning from scratch in the agent's current task, as the number of tasks in the knowledge base grows?*), through the total number of permutations, the source task selected for each target task remained constant. That is, regardless of adding more source tasks to the knowledge base, the similarity function always chose the same source task as the most similar one for each target task. Table 6.1 shows the most similar task for each target task in the lifelong learning setting under the *Growing KB* column. Additionally, a comparison is made against the most similar task selections from the knowledge transfer experiments from Chapter 5.

The results from Table 6.1 show that in half of the set of target tasks, the same source tasks was ranked as the most similar one as in the knowledge transfer experiments. However, regarding the other three target tasks, Inverted Double Pendulum and Walker2D selected their source counterpart as the most similar task in the lifelong learning setting. Concerning the consistency of the source task selection through the set of permutations of target tasks, and the source tasks selected for each target task, the SimKnoT method shows potential to be part of a lifelong reinforcement learning system.

In relation to the second question (*i.e., How does the agent perform in sequences of tasks of unknown order?*), the average sum of total reward ratios (see Eq. 6.1) over the 10 sequences of tasks is 3.062 ± 0.381. Being a score above 0.0 shows that the overall performance of the SimKnot-based LRL agent was positive, despite that the agent was not guaranteed to have in its knowledge base a source task that would benefit the current target task. Additionally, Table 6.2 shows two sequences used for the evaluation of the agent, while Fig. 6.2 show the individual total reward ratio (see Eq. 6.1) obtained in each target task. The task sequences shown Table 6.2 and Fig. 6.2 show that despite the overall effect of SimKnoT may be positive, there are scenarios in which not having learned a task before that is related to the current task can be significantly harmful (*e.g.* in sequence **A** where knowledge is transferred from Swimmer to IDP, or in sequence **B** where SimKnoT transfers knowledge from H. Cheetah to IDP).

Therefore, even though SimKnoT is able to avoid negative transfer in certain situation (*i.e.*, when there is a source task that is related to the target task), it still

Table 6.1: Most similar source task for each target task comparison between a static knowledge base (KB), *i.e.,* the knowledge transfer setting from Section 5.4.1, and the continually growing knowledge base in a lifelong learning agent. Rows that differ between the static and growing knowledge base settings are bold. IDP and IP are abbreviations for inverted double pendulum and inverted pendulum respectively.

Target Task	Most Similar Source Task	
	Static KB	Growing KB
H. Cheetah	**Swimmer**	**Hopper**
Hopper	Walker-2D	Walker-2D
IDP	**Walker-2D**	**IDP**
IP	IP	IP
Swimmer	Swimmer	Swimmer
Walker-2D	**Hopper**	**Walker-2D**

is prone to fail identifying that none of the available sources of knowledge can help learning better/faster the target task. Moreover, SimKnoT has a scalability problem regarding two aspects:

1. **Memory**: Since no knowledge consolidation process is performed, storing every data instance observed through out the LRL agent's lifespan is not a feasible solution, particularly if it is expected to operate for long periods of time.

2. **Computational Time**: Training the encoder-decoder neural networks responsible for mapping states/actions is slow. Training with a data set of 1 million observations takes 7.53 ± 0.48 hours for an epoch (*i.e.*, a single iteration over the complete data set) to finish.

Thus, future work for the SimKnoT-based LRL include addressing the computational complexity issues listed above. For the memory problem, training prototype instances to cover as much information possible from a data set is a viable option to decrease the memory requirements [Liu et al., 2023]. On the other hand, we have observed that even with the approximation presented in Section 5.1.3, the gradients for the geometry preserving loss are the most expensive terms to compute. Hence, future iterations of SimKnoT may involve replacing the effect the geometry preserving loss has in the mappings with a penalty term that is more efficient to compute.

Table 6.2: Examples of lifelong learning sequences (see Fig. 6.2 for their performance). Each row represents a task sequence, while each columns show the order in which tasks were presented to the LRL system (from left to right). Each cell shows the source task from which knowledge was transferred to the target task at the time.

Task Sequence	Task ID					
	0	1	2	3	4	5
A	Swimmer	Swimmer to IDP	Swimmer to Walker2D	Walker2D to Hopper	Swimmer to IP	Swimmer to H. Cheetah
B	H. Cheetah	H. Cheetah to Hopper	Hopper to IP	H. Cheetah to Swimmer	H. Cheetah to IDP	Hopper to Walker2D

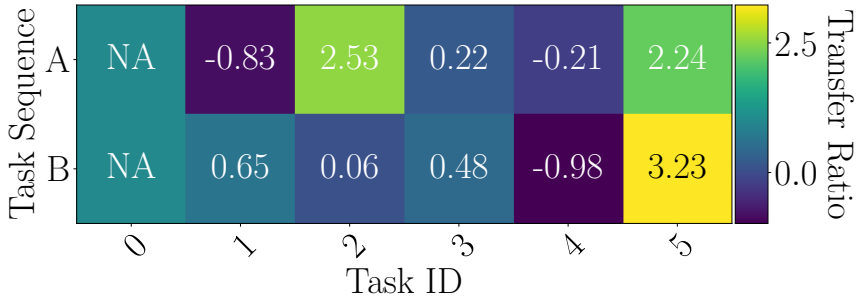


Figure 6.2: Total reward ratio performance in two sample sequences (see Table 6.2 for the order of tasks and their source task selection) of the lifelong reinforcement learning (LRL) system based on the SimKnoT method. Each row represents a task sequence, while each columns show the order in which tasks were presented to the LRL system (from left to right). The first task from each sequence does not show a score, since it is learned from scratch without knowledge transfer.

6.3 Chapter Summary

In this chapter, a lifelong reinforcement learning system was presented and evaluated. After randomly generating multiple task sequences and passing them to the SimKnoT method (introduced in Section 5.3) system's robustness regarding the selection source tasks was evaluated. It is worth noting that despite the how varied the set of evaluation tasks was, the SimKnoT-based lifelong learning agent was able to consistently select the same source of knowledge for each target task. On the other hand, through a complete evaluation of SimKnoT-based as a LRL system, it was shown the ability of the SimKnoT-based system to produce an overall positive effect. The development of more memory-efficient knowledge consolidation strategies and time-efficient training methods are left for future work.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In order to successfully transfer knowledge, it is necessary to identify the features in which tasks are similar so that knowledge can be reused in a different scenario than the one it was acquired/learned. In the case of tasks with different state-action spaces (*i.e.,* cross domain), finding equivalent attributes is particularly difficult, since there is no clear correspondence between states and actions across tasks. Furthermore, if there are multiple options from which knowledge can be transferred, finding a piece of knowledge that benefits the learning agent entails an additional challenge.

In this research, we addressed the problem of cross-domain knowledge transfer in lifelong reinforcement learning. With the development of a similarity measure and knowledge transfer method (for tasks with discrete but different spaces) we were able to associate unlabeled tasks, based on the similarity of partially learned action-value functions. Moreover, we also developed a similarity-based transfer method for tasks with continuous spaces that compares tasks based on their dynamics. These knowledge selection and transfer strategies allow taking sequences of tasks defined over different state and action spaces. The similarity measures were evaluated, in tandem with their respective knowledge transfer methods, to estimate their ability to serve as heuristic for the selection of sources of knowledge that accelerate the learning process in target tasks.

A pair of similarity functions and knowledge transfer methods were developed,

which address tasks with discrete and continuous state-action spaces. In the case of discrete-space tasks, the similarity function measures how similar their action-value functions are by comparing the distributions of states and actions (based on their q values). In this way, tasks that were highly related to each other (*e.g.,* Frozen Lake and Frozen Lake 8×8) were matched by the similarity function and positive transfer was achieved. On the other hand, the similarity function for the continuous-space scenario compares the reward and state-transition dynamics. By learning a set of inter-domain mappings, aligned based on the immediate reward, states and actions from different representations could be directly compared and matched across spaces. These developments allowed transferring policies across different control-oriented problems and achieve higher performance with less training data. Thus, the first and second specific objective have been completed, as the similarity functions and knowledge transfer have been able to yield learning improvements by measuring similarity and transferring knowledge across tasks from different domains.

Moreover, the evaluation of the knowledge transfer method (for the continuous-space case) was performed in multiple sequences of tasks to determine its ability to select reliable sources of knowledge for transfer purposes, as well as to learn tasks faster (than learning from scratch) over task sequences. In the lifelong learning setting, the knowledge transfer method consistently selected the same source of knowledge for the target task, regardless of the order in which tasks were presented to the system, and was able to produce a positive transfer over the task sequences. These results, in conjunction with the knowledge transfer evaluation, show the method's proficiency to select and transfer knowledge despite the number and variety of available tasks. Therefore, with the knowledge transfer and source-selection for lifelong learning evaluations the third specific objective, regarding the integration of a similarity function and transfer method for their evaluation in the cross-domain lifelong learning setting, has been completed.

With the integration and evaluation of the similarity function and knowledge transfer method in the cross-domain lifelong reinforcement learning setting, the general objective of this thesis has been fulfilled. Additionally, through the knowledge selection and transfer results our hypothesis has been validated, since the knowledge transfer method has shown its ability to accelerate learning from cross-domain related tasks, as well as to not forget what has been learned by adopting a store-all approach. Furthermore, the developments presented in this document represent a step towards more autonomous systems, where agents are responsible for deciding

how previous experiences could be reused to accelerate learning when no supervision is available.

Additionally, the following contributions were made:

- A systematic review of knowledge transfer methods focused on the cross-domain reinforcement learning setting.
- A model-based similarity measure for source selection purposes in cross-domain reinforcement learning.
- A knowledge transfer method for cross-domain reinforcement learning.
- A lifelong cross-domain reinforcement learning system.

7.2 Future Work

There are multiple directions in which the presented work could be furthered explored. As lifelong learning systems are expected to operate for an undetermined number of tasks, having methods that consolidate knowledge in a memory efficient way is an essential element for its scalability. Developing strategies that exploit redundancy across tasks to save memory space can provide a way to remain memory efficient while preventing catastrophic forgetting. Similarly, reducing the training time required to learn the state-action mappings is necessary for the system to operate in real-time task sequences (instead of taking days to train).

On the other hand, adapting the similarity function and knowledge transfer method for image-based representations could potentially expand its applicability to real-world scenarios. Given that training robots can require very large amounts of data, transferring knowledge from image-based policies, or third-person point of view demonstrations, would significantly widen the range of tasks robots could be trained to solve, as knowledge transfer methods would be prepared to transfer from multiple modalities.

Moreover, even though the proposed lifelong learning system is endowed with the similarity function to select the best source task for the current target task, there is no way for the agent to determine if none of the source tasks can produce positive

transfer. Similarly, the agent does not have a method to evaluate the amount of data needed to learn the mappings and measure similarity, according to the target task's complexity. Addressing these limitations could significantly mitigate the risk of experiencing negative transfer.

Appendix A

Appendix A

In Tables A.1, A.2, and A.3 one will find the parameters used to train the reward/transition approximation models, the alignment encoder-decoders and the RL agents, respectively. For more detail on our implementation, please refer to `https://github.com/saSerrano/simknot`.

Table A.1: Parameters used to train the neural networks that approximate the reward and transition models, and that are used to measure similarity, as described in Eq. 5.12. $|S|$ and $|Y|$ are the number of state and action variables, respectively.

Parameter	Value	
	Reward Model	Transition Model
# Output Units	1	$\|S\|$
Output Act. Fun.	Linear	
No. of Hidden Layers	4	
No. Units in Hidden Layers	64	
Hidden Act. Fun.	ReLU	
# Input Units	$\|S\| + \|A\|$	
Loss Function	Mean Squared Error	
Optimizer	Adam	
Batch Size	512	
Epochs	300	
Learning Rate	0.001	

Table A.2: Parameters used to train the encoder-decoder pairs that are used to align the state and action spaces. $|X|$ and $|Y|$ are the dimensionality of the two spaces that will be aligned.

Parameter	Value	
	Encoder	Decoder
# Input Units	$\|X\|$	$max(\|X\|, \|Y\|) + 1$
# Output Units	$max(\|X\|, \|Y\|) + 1$	$\|X\|$
Output Act. Fun.	Linear	
No. of Hidden Layers	4	
No. Units in Hidden Layers	64	
Hidden Act. Fun.	ReLU	
Loss Function	Approximated ReBA (Eq. 5.14)	
Loss Weights	$\lambda_1 = 1, \lambda_2 = 1, \lambda_3 = 1, \lambda_4 = 0.5, \lambda_5 = 0.05$	
δ in L_A (Eq. 5.2)	0.25	
k in $\tilde{L_G}$ (Eq. 5.11)	2	
Optimizer	Adam	
Batch Size	512	
Epochs	10	
Learning Rate	0.01	
Clipping Method	Adaptive Gradient Clipping (AGC)	
AGC Factor	0.01	

Table A.3: Parameters used to train the policy in SAC, SAC+FBO and SimKnoT.

Parameter	Value
Initial Replay	100
Batch Size	64
No. of Hidden Layers	2
No. Units in Hidden Layers	64
Hidden Act. Function	ReLU
Warm up Transitions	100
τ	0.005
Learning Rate	0.001
Learning Rate α	0.001
Discount Factor	0.99
Optimizer	Adam
Critic Loss	Mean Squared Error

Bibliography

[Aktas et al., 2023] Aktas, H., Nagai, Y., Asada, M., Oztop, E., and Ugur, E. (2023). Correspondence learning between morphologically different robots via task demonstrations. *arXiv e-prints*, pages arXiv–2310.

[Ammar et al., 2015a] Ammar, H. B., Eaton, E., Luna, J. M., and Ruvolo, P. (2015a). Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning. In *Twenty-fourth international joint conference on artificial intelligence*.

[Ammar et al., 2014a] Ammar, H. B., Eaton, E., Ruvolo, P., and Taylor, M. (2014a). Online multi-task learning for policy gradient methods. In *International conference on machine learning*, pages 1206–1214. PMLR.

[Ammar et al., 2015b] Ammar, H. B., Eaton, E., Ruvolo, P., and Taylor, M. (2015b). Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

[Ammar et al., 2014b] Ammar, H. B., Eaton, E., Taylor, M. E., Mocanu, D. C., Driessens, K., Weiss, G., and Tüyls, K. (2014b). An automated measure of mdp similarity for transfer in reinforcement learning. In *28th AAAI Conference on Artificial Intelligence, AAAI 2014*, pages 31–37. AI Access Foundation.

[Ammar and Taylor, 2012] Ammar, H. B. and Taylor, M. E. (2012). Reinforcement learning transfer via common subspaces. In *Adaptive and Learning Agents: International Workshop, ALA 2011, Held at AAMAS 2011, Taipei, Taiwan, May 2, 2011, Revised Selected Papers*, pages 21–36. Springer.

[Ammar et al., 2012] Ammar, H. B., Tuyls, K., Taylor, M. E., Driessens, K., and Weiss, G. (2012). Reinforcement learning transfer via sparse coding. In *Pro-*

ceedings of the 11th international conference on autonomous agents and multiagent systems, volume 1, pages 383–390. International Foundation for Autonomous Agents and Multiagent Systems ....

[Andrychowicz et al., 2020] Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20.

[Banerjee and Stone, 2007] Banerjee, B. and Stone, P. (2007). General game learning using knowledge transfer. In *IJCAI*, pages 672–677. Citeseer.

[Bank et al., 2023] Bank, D., Koenigstein, N., and Giryes, R. (2023). Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pages 353–374.

[Bellemare et al., 2013] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

[Bellman and Dreyfus, 2015] Bellman, R. E. and Dreyfus, S. E. (2015). *Applied dynamic programming*, volume 2050. Princeton university press.

[Bishop, 1998] Bishop, C. M. (1998). Latent variable models. In *Learning in graphical models*, pages 371–403. Springer.

[Bogue, 2016] Bogue, R. (2016). Growth in e-commerce boosts innovation in the warehouse robot market. *Industrial Robot: An International Journal*, 43(6):583–587.

[Bone, 2008] Bone, N. (2008). A survey of transfer learning methods for reinforcement learning.

[Brockman et al., 2016] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

[Brys et al., 2015] Brys, T., Harutyunyan, A., Taylor, M. E., and Nowé, A. (2015). Policy transfer using reward shaping. In *AAMAS*, pages 181–188.

[Cao et al., 2022] Cao, Z., Wang, Z., and Sadigh, D. (2022). Learning from imperfect demonstrations via adversarial confidence transfer. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 441–447. IEEE.

[Carbonell, 1983] Carbonell, J. G. (1983). Learning by analogy: Formulating and generalizing plans from past experience. In *Machine learning*, pages 137–161. Elsevier.

[Carroll and Seppi, 2005] Carroll, J. L. and Seppi, K. (2005). Task similarity measures for transfer in reinforcement learning task libraries. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 803–808. IEEE.

[Castro et al., 2021] Castro, P. S., Kastner, T., Panangaden, P., and Rowland, M. (2021). Mico: Improved representations via sampling-based state similarity for markov decision processes. *Advances in Neural Information Processing Systems*, 34:30113–30126.

[Chen et al., 2024] Chen, L. Y., Hari, K., Dharmarajan, K., Xu, C., Vuong, Q., and Goldberg, K. (2024). Mirage: Cross-embodiment zero-shot policy transfer with cross-painting. *arXiv preprint arXiv:2402.19249*.

[Chen et al., 2019] Chen, Y., Chen, Y., Hu, Z., Yang, T., Fan, C., Yu, Y., and Hao, J. (2019). Learning action-transferable policy with action embedding. *arXiv preprint arXiv:1909.02291*.

[Chen and Liu, 2018] Chen, Z. and Liu, B. (2018). Lifelong Machine Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207.

[Cheng et al., 2018] Cheng, Q., Wang, X., Niu, Y., and Shen, L. (2018). Reusing source task knowledge via transfer approximator in reinforcement transfer learning. *Symmetry*, 11(1):25.

[Da Silva and Costa, 2019] Da Silva, F. L. and Costa, A. H. R. (2019). A Survey on Transfer Learning for Multiagent Reinforcement Learning Systems. *Journal of Artificial Intelligence Research*, 64:645–703.

[Deng and Du, 2009] Deng, Y. and Du, W. (2009). The Kantorovich Metric in Computer Science: A Brief Survey. *Electronic Notes in Theoretical Computer Science*, 253(3):73–82.

[D'Eramo et al., 2021] D'Eramo, C., Tateo, D., Bonarini, A., Restelli, M., and Peters, J. (2021). Mushroomrl: Simplifying reinforcement learning research. *Journal of Machine Learning Research*, 22(131):1–5.

[Devin et al., 2017] Devin, C., Gupta, A., Darrell, T., Abbeel, P., and Levine, S. (2017). Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2169–2176. IEEE.

[Dick et al., 2024] Dick, J., Nath, S., Peridis, C., Benjamin, E., Kolouri, S., and Soltoggio, A. (2024). Statistical context detection for deep lifelong reinforcement learning. *arXiv preprint arXiv:2405.19047*.

[Dietterich, 2000] Dietterich, T. G. (2000). Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303.

[Dwibedi et al., 2019] Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., and Zisserman, A. (2019). Temporal cycle-consistency learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1801–1810.

[Fernando et al., 2017] Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. (2017). Pathnet: Evolution Channels Gradient Descent in Super Neural Networks. *arXiv preprint arXiv:1701.08734*.

[Ferns et al., 2004] Ferns, N., Panangaden, P., and Precup, D. (2004). Metrics for finite markov decision processes. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 162–169.

[Ferns et al., 2012] Ferns, N., Panangaden, P., and Precup, D. (2012). Metrics for finite markov decision processes. *arXiv preprint arXiv:1207.4114*.

[Fickinger et al., 2021] Fickinger, A., Cohen, S., Russell, S., and Amos, B. (2021). Cross-domain imitation learning via optimal transport. In *International Conference on Learning Representations*.

[Fix, 1985] Fix, E. (1985). *Discriminatory analysis: nonparametric discrimination, consistency properties*, volume 1. USAF school of Aviation Medicine.

[Franzmeyer et al., 2022] Franzmeyer, T., Torr, P., and Henriques, J. F. (2022). Learn what matters: cross-domain imitation learning with task-relevant embeddings. *Advances in Neural Information Processing Systems*, 35:26283–26294.

[Gao and Chien, 2017] Gao, Y. and Chien, S. (2017). Review on space robotics: Toward top-level science through space exploration. *Science Robotics*, 2(7):eaan5074.

[García et al., 2022] García, J., Visús, Á., and Fernández, F. (2022). A taxonomy for similarity metrics between markov decision processes. *Machine Learning*, 111(11):4217–4247.

[Gehr et al., 2016] Gehr, T., Misailovic, S., and Vechev, M. (2016). Psi: Exact symbolic inference for probabilistic programs. In *Computer Aided Verification: 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I 28*, pages 62–83. Springer.

[Gibbs and Su, 2002] Gibbs, A. L. and Su, F. E. (2002). On Choosing and Bounding Probability Metrics. *International statistical review*, 70(3):419–435.

[Glatt et al., 2016] Glatt, R., Da Silva, F. L., and Costa, A. H. R. (2016). Towards knowledge transfer in deep reinforcement learning. In *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 91–96. IEEE.

[Gleave et al., 2020] Gleave, A., Dennis, M., Legg, S., Russell, S., and Leike, J. (2020). Quantifying differences in reward functions. *arXiv preprint arXiv:2006.13900*.

[Golub and Van Loan, 2013] Golub, G. H. and Van Loan, C. F. (2013). *Matrix computations*. JHU press.

[Grosse et al., 2002] Grosse, I., Bernaola-Galván, P., Carpena, P., Román-Roldán, R., Oliver, J., and Stanley, H. E. (2002). Analysis of symbolic sequences using the jensen-shannon divergence. *Physical Review E*, 65(4):041905.

[Gui et al., 2023] Gui, H., Pang, S., Yu, S., Qiao, S., Qi, Y., He, X., Wang, M., and Zhai, X. (2023). Cross-domain policy adaptation with dynamics alignment. *Neural Networks*, 167:104–117.

[Guo et al., 2017] Guo, T., Dong, J., Li, H., and Gao, Y. (2017). Simple convolutional neural network on image classification. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pages 721–724. IEEE.

[Gupta et al., 2017] Gupta, A., Devin, C., Liu, Y., Abbeel, P., and Levine, S. (2017). Learning invariant feature spaces to transfer skills with reinforcement learning. In *International Conference on Learning Representations*.

[Haarnoja et al., 2018] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.

[Hejna et al., 2020] Hejna, D., Pinto, L., and Abbeel, P. (2020). Hierarchically decoupled imitation for morphological transfer. In *International Conference on Machine Learning*, pages 4159–4171. PMLR.

[Heng et al., 2022] Heng, Y., Yang, T., ZHENG, Y., HAO, J., and Taylor, M. E. (2022). Cross-domain adaptive transfer reinforcement learning based on state-action correspondence. In *The 38th Conference on Uncertainty in Artificial Intelligence*.

[Henrikson, 1999] Henrikson, J. (1999). Completeness and Total Boundedness of the Hausdorff Metric. *MIT Undergraduate Journal of Mathematics*, 1:69–80.

[Hoffman et al., 2018] Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A., and Darrell, T. (2018). Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. Pmlr.

[Howes, 2012] Howes, N. R. (2012). *Modern analysis and topology*. Springer Science & Business Media.

[Hu and Montana, 2019] Hu, Y. and Montana, G. (2019). Skill transfer in deep reinforcement learning under morphological heterogeneity. *arXiv preprint arXiv:1908.05265*.

[Huttenlocher et al., 1993] Huttenlocher, D. P., Klanderman, G. A., and Rucklidge, W. J. (1993). Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863.

[Isele and Cosgun, 2018] Isele, D. and Cosgun, A. (2018). Selective experience replay for lifelong learning. In *Thirty-second AAAI conference on artificial intelligence*.

[Jain and Dubes, 1988] Jain, A. K. and Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

[Joshi and Chowdhary, 2018] Joshi, G. and Chowdhary, G. (2018). Cross-domain transfer in reinforcement learning using target apprentice. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7525–7532. IEEE.

[Kaelbling et al., 1998] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134.

[Khetarpal et al., 2022] Khetarpal, K., Riemer, M., Rish, I., and Precup, D. (2022). Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476.

[Kim et al., 2020] Kim, K., Gu, Y., Song, J., Zhao, S., and Ermon, S. (2020). Domain adaptive imitation learning. In *International Conference on Machine Learning*, pages 5286–5295. PMLR.

[Kirkpatrick et al., 2017] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

[Kober et al., 2013] Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274.

[Kuhlmann and Stone, 2007] Kuhlmann, G. and Stone, P. (2007). Graph-based domain mapping for transfer learning in general games. In *Machine Learning: ECML 2007: 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007. Proceedings 18*, pages 188–200. Springer.

[Lambora et al., 2019] Lambora, A., Gupta, K., and Chopra, K. (2019). Genetic algorithm-a literature review. In *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)*, pages 380–384. IEEE.

[Lazaric, 2012a] Lazaric, A. (2012a). Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning: State-of-the-Art*, pages 143–173. Springer.

[Lazaric, 2012b] Lazaric, A. (2012b). Transfer in Reinforcement Learning: A Framework and a Survey. In *Reinforcement Learning*, pages 143–173. Springer.

[Lecarpentier et al., 2020] Lecarpentier, E., Abel, D., Asadi, K., Jinnai, Y., Rachelson, E., and Littman, M. L. (2020). Lipschitz Lifelong Reinforcement Learning. *arXiv preprint arXiv:2001.05411*.

[Li et al., 2023a] Li, G., Wong, T.-W., Shih, B., Guo, C., Wang, L., Liu, J., Wang, T., Liu, X., Yan, J., Wu, B., et al. (2023a). Bioinspired soft robots for deep-sea exploration. *Nature Communications*, 14(1):7097.

[Li et al., 2023b] Li, T., Jung, H., Gombolay, M., Cho, Y., and Ha, S. (2023b). Crossloco: Human motion driven control of legged robots via guided unsupervised reinforcement learning. In *The Twelfth International Conference on Learning Representations*.

[Li et al., 2023c] Li, T., Won, J., Clegg, A., Kim, J., Rai, A., and Ha, S. (2023c). Ace: Adversarial correspondence embedding for cross morphology motion retargeting from human to nonhuman characters. *arXiv preprint arXiv:2305.14792*.

[Li, 2017] Li, Y. (2017). Deep Reinforcement Learning: An Overview. *ArXiv preprint arXiv:1701.07274*.

[Liu et al., 2019] Liu, I.-J., Peng, J., and Schwing, A. G. (2019). Knowledge Flow: Improve Upon Your Teachers. *arXiv preprint arXiv:1904.05878*.

[Liu et al., 2023] Liu, X., Chen, Y., Li, H., Li, B., and Zhao, D. (2023). Cross-domain random pre-training with prototypes for reinforcement learning. *arXiv preprint arXiv:2302.05614*.

[Makoviychuk et al., 2021] Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., et al. (2021). Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*.

[Mendez et al., 2018] Mendez, J., Shivkumar, S., and Eaton, E. (2018). Lifelong Inverse Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pages 4502–4513.

[Mendez and Eaton, 2020] Mendez, J. A. and Eaton, E. (2020). Lifelong learning of factored policies via policy gradients. In *4th Lifelong Machine Learning Workshop at ICML 2020*.

[Mendez et al., 2022] Mendez, J. A., van Seijen, H., and Eaton, E. (2022). Modular lifelong reinforcement learning via neural composition. *arXiv preprint arXiv:2207.00429*.

[Mnih et al., 2016] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. In *International conference on machine learning*, pages 1928–1937.

[Mnih et al., 2015] 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. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

[Moore, 1990] Moore, A. W. (1990). Efficient memory-based learning for robot control. Technical report, University of Cambridge.

[Muppidi et al., 2024] Muppidi, A., Zhang, Z., and Yang, H. (2024). Pick up the pace: A parameter-free optimizer for lifelong reinforcement learning. *arXiv preprint arXiv:2405.16642*.

[Nagatani et al., 2013] Nagatani, K., Kiribayashi, S., Okada, Y., Otake, K., Yoshida, K., Tadokoro, S., Nishimura, T., Yoshida, T., Koyanagi, E., Fukushima, M., et al. (2013). Emergency response to the nuclear accident at the fukushima daiichi nuclear power plants using mobile rescue robots. *Journal of Field Robotics*, 30(1):44–63.

[Narayan and Leong, 2019] Narayan, A. and Leong, T. Y. (2019). Effects of task similarity on policy transfer with selective exploration in reinforcement learning. In *Proceedings of the 18th international conference on autonomous agents and multiagent systems*, pages 2132–2134.

[Ng et al., 1999] Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287.

[Nielsen, 2019] Nielsen, F. (2019). On the jensen–shannon symmetrization of distances relying on abstract means. *Entropy*, 21(5):485.

[Ontañón, 2020] Ontañón, S. (2020). An overview of distance and similarity functions for structured data. *Artificial Intelligence Review*, 53(7):5309–5351.

[Pan and Yang, 2009] Pan, S. J. and Yang, Q. (2009). A Survey on Transfer Learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[Panaretos and Zemel, 2019] Panaretos, V. M. and Zemel, Y. (2019). Statistical aspects of wasserstein distances. *Annual review of statistics and its application*, 6(1):405–431.

[Parisi et al., 2019] Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71.

[Puterman, 2014] Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[Qian et al., 2020] Qian, Y., Xiong, F., and Liu, Z. (2020). Intra-domain knowledge generalization in cross-domain lifelong reinforcement learning. In *International Conference on Neural Information Processing*, pages 386–394. Springer.

[Ramos et al., 2021] Ramos, S., Girgin, S., Hussenot, L., Vincent, D., Yakubovich, H., Toyama, D., Gergely, A., Stanczyk, P., Marinier, R., Harmsen, J., Pietquin, O., and Momchev, N. (2021). Rlds: an ecosystem to generate, share and use datasets in reinforcement learning.

[Ravindran, 2004] Ravindran, B. (2004). *An algebraic approach to abstraction in reinforcement learning*. University of Massachusetts Amherst.

[Raychaudhuri et al., 2021] Raychaudhuri, D. S., Paul, S., Vanbaar, J., and Roy-Chowdhury, A. K. (2021). Cross-domain imitation from observations. In *International Conference on Machine Learning*, pages 8902–8912. PMLR.

[Robins, 1995] Robins, A. (1995). Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146.

[Rockafellar and Wets, 2009] Rockafellar, R. T. and Wets, R. J.-B. (2009). *Variational analysis*, volume 317. Springer Science & Business Media.

[Rummery and Niranjan, 1994] Rummery, G. A. and Niranjan, M. (1994). *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK.

[Rusu et al., 2015] Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., and Hadsell, R. (2015). Policy distillation. *arXiv preprint arXiv:1511.06295*.

[Rusu et al., 2016] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive Neural Networks. *arXiv preprint arXiv:1606.04671*.

[Salhotra et al., 2023] Salhotra, G., Liu, I.-C. A., and Sukhatme, G. S. (2023). Learning robot manipulation from cross-morphology demonstration. In *7th Annual Conference on Robot Learning*.

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[Schwarz et al., 2018] Schwarz, J., Luketina, J., Czarnecki, W. M., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. (2018). Progress & Compress: A Scalable Framework for Continual Learning. *arXiv preprint arXiv:1805.06370*.

[Serrano, 2021] Serrano, S. A. (2021). Inter-task similarity for lifelong reinforcement learning in heterogeneous tasks. In Zhou, Z.-H., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4915–4916. International Joint Conferences on Artificial Intelligence Organization. Doctoral Consortium.

[Serrano et al., 2021a] Serrano, S. A., Martinez-Carranza, J., and Sucar, L. E. (2021a). Inter-task similarity measure for heterogeneous tasks. In *Robot World Cup*, pages 40–52. Springer.

[Serrano et al., 2023] Serrano, S. A., Martinez-Carranza, J., and Sucar, L. E. (2023). Similarity-based knowledge transfer for cross-domain reinforcement learning. *arXiv preprint arXiv:2312.03764*.

[Serrano et al., 2024] Serrano, S. A., Martinez-Carranza, J., and Sucar, L. E. (2024). Knowledge transfer for cross-domain reinforcement learning: A systematic review. *IEEE Access*, pages 1–1.

[Serrano et al., 2021b] Serrano, S. A., Santiago, E., Martinez-Carranza, J., Morales, E. F., and Sucar, L. E. (2021b). Knowledge-based hierarchical pomdps for task planning. *Journal of Intelligent & Robotic Systems*, 101:1–30.

[Shankar et al., 2022] Shankar, T., Lin, Y., Rajeswaran, A., Kumar, V., Anderson, S., and Oh, J. (2022). Translating robot skills: Learning unsupervised skill correspondences across robots. In *International Conference on Machine Learning*, pages 19626–19644. PMLR.

[Song et al., 2016] Song, J., Gao, Y., Wang, H., and An, B. (2016). Measuring the distance between finite markov decision processes. In *Proceedings of the 2016 international conference on autonomous agents & multiagent systems*, pages 468–476.

[Soni and Singh, 2006] Soni, V. and Singh, S. (2006). Using homomorphisms to transfer options across continuous reinforcement learning domains. In *AAAI*, volume 6, pages 494–499.

[Sorg and Singh, 2009] Sorg, J. and Singh, S. (2009). Transfer via soft homomorphisms. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 741–748.

[Sucar, 2015] Sucar, L. E. (2015). Probabilistic Graphical Models. *Advances in Computer Vision and Pattern Recognition. London: Springer London. doi*, 10:978–1.

[Sutskever et al., 2009] Sutskever, I., Hinton, G. E., and Taylor, G. W. (2009). The Recurrent Temporal Restricted Boltzmann Machine. In *Advances in neural information processing systems*, pages 1601–1608.

[Sutton, 1995] Sutton, R. S. (1995). Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, 8.

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

[Talvitie and Singh, 2007] Talvitie, E. and Singh, S. (2007). An experts algorithm for transfer learning. In *IJCAI*, pages 1065–1070. Citeseer.

[Tanioka, 2019] Tanioka, T. (2019). Nursing and rehabilitative care of the elderly using humanoid robots. *The Journal of Medical Investigation*, 66(1.2):19–23.

[Tao et al., 2021] Tao, Y., Genc, S., Chung, J., Sun, T., and Mallya, S. (2021). Repaint: Knowledge transfer in deep reinforcement learning. In *International Conference on Machine Learning*, pages 10141–10152. PMLR.

[Tasse et al., 2021] Tasse, G. N., James, S., and Rosman, B. (2021). Generalisation in lifelong reinforcement learning through logical composition. In *International Conference on Learning Representations*.

[Taylor et al., 2008a] Taylor, M. E., Jong, N. K., and Stone, P. (2008a). Transferring instances for model-based reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II 19*, pages 488–505. Springer.

[Taylor et al., 2008b] Taylor, M. E., Kuhlmann, G., and Stone, P. (2008b). Autonomous transfer for reinforcement learning. In *AAMAS (1)*, pages 283–290.

[Taylor and Stone, 2005] Taylor, M. E. and Stone, P. (2005). Behavior transfer for value-function-based reinforcement learning. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 53–59.

[Taylor and Stone, 2007a] Taylor, M. E. and Stone, P. (2007a). Cross-domain transfer for reinforcement learning. In *Proceedings of the 24th international conference on Machine learning*, pages 879–886.

[Taylor and Stone, 2007b] Taylor, M. E. and Stone, P. (2007b). Representation transfer for reinforcement learning. In *AAAI Fall Symposium: Computational Approaches to Representation Change during Learning and Development*, pages 78–85.

[Taylor and Stone, 2009a] Taylor, M. E. and Stone, P. (2009a). Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685.

[Taylor and Stone, 2009b] Taylor, M. E. and Stone, P. (2009b). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7).

[Taylor et al., 2005] Taylor, M. E., Stone, P., and Liu, Y. (2005). Value functions for rl-based behavior transfer: A comparative study. In *AAAI*, volume 5, pages 880–885.

[Taylor et al., 2007a] Taylor, M. E., Stone, P., and Liu, Y. (2007a). Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(9).

[Taylor et al., 2007b] Taylor, M. E., Whiteson, S., and Stone, P. (2007b). Transfer via inter-task mappings in policy search reinforcement learning. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8.

[Tessler et al., 2017] Tessler, C., Givony, S., Zahavy, T., Mankowitz, D. J., and Mannor, S. (2017). A Deep Hierarchical Approach to Lifelong Learning in Minecraft. In *Thirty-First AAAI Conference on Artificial Intelligence*.

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

[Torrey et al., 2006] Torrey, L., Shavlik, J., Walker, T., and Maclin, R. (2006). Skill acquisition via transfer learning and advice taking. In *European Conference on Machine Learning*, pages 425–436. Springer.

[Torrey et al., 2008] Torrey, L., Shavlik, J., Walker, T., and Maclin, R. (2008). Relational macros for transfer in reinforcement learning. In *Inductive Logic Programming: 17th International Conference, ILP 2007, Corvallis, OR, USA, June 19-21, 2007, Revised Selected Papers 17*, pages 254–268. Springer.

[Torrey et al., 2005] Torrey, L., Walker, T., Shavlik, J., and Maclin, R. (2005). Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005. Proceedings 16*, pages 412–424. Springer.

[Towers et al., 2023] Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., Cola, G. d., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Shen, A. T. J., and Younis, O. G. (2023). Gymnasium.

[Tschannen et al., 2018] Tschannen, M., Bachem, O., and Lucic, M. (2018). Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*.

[Tversky, 1977] Tversky, A. (1977). Features of Similarity. *Psychological review*, 84(4):327.

[Wan et al., 2020] Wan, M., Gangwani, T., and Peng, J. (2020). Mutual information based knowledge transfer under state-action dimension mismatch. In *Conference on Uncertainty in Artificial Intelligence*, pages 1218–1227. PMLR.

[Wang and Mahadevan, 2009] Wang, C. and Mahadevan, S. (2009). Manifold alignment without correspondence. In *IJCAI*, volume 2, page 3.

[Wang et al., 2019] Wang, H., Dong, S., and Shao, L. (2019). Measuring structural similarities in finite mdps. In *IJCAI*, pages 3684–3690.

[Wang et al., 2022] Wang, Z., Cao, Z., Hao, Y., and Sadigh, D. (2022). Weakly supervised correspondence learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 469–476. IEEE.

[Watahiki et al., 2023] Watahiki, H., Iwase, R., Unno, R., and Tsuruoka, Y. (2023). Leveraging behavioral cloning for representation alignment in cross-domain policy transfer. In *NeurIPS 2023 Workshop on Generalization in Planning*.

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

[Wilson, 1931] Wilson, W. A. (1931). On semi-metric spaces. *American Journal of Mathematics*, 53(2):361–373.

[Xie and Finn, 2022] Xie, A. and Finn, C. (2022). Lifelong robotic reinforcement learning by retaining experiences. In *Conference on Lifelong Learning Agents*, pages 838–855. PMLR.

[Yang et al., 2023] Yang, Q., Stork, J. A., and Stoyanov, T. (2023). Learn from robot: Transferring skills for diverse manipulation via cycle generative networks. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pages 1–6. IEEE.

[Zakka et al., 2022] Zakka, K., Zeng, A., Florence, P., Tompson, J., Bohg, J., and Dwibedi, D. (2022). Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning*, pages 537–546. PMLR.

[Zhang et al., 2024] Zhang, G., Feng, L., Wang, Y., Li, M., Xie, H., and Tan, K. C. (2024). Reinforcement learning with adaptive policy gradient transfer across heterogeneous problems. *IEEE Transactions on Emerging Topics in Computational Intelligence*.

[Zhang et al., 2021a] Zhang, Q., Xiao, T., Efros, A. A., Pinto, L., and Wang, X. (2021a). Learning cross-domain correspondence for control with dynamics cycle-consistency. In *International Conference on Learning Representations*.

[Zhang et al., 2021b] Zhang, Y., Zhang, X., Shen, T., Zhou, Y., and Wang, Z. (2021b). Feature-option-action: a domain adaption transfer reinforcement learning framework. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–12. IEEE.

[Zhao et al., 2017] Zhao, Y., Deng, B., Shen, C., Liu, Y., Lu, H., and Hua, X.-S. (2017). Spatio-temporal autoencoder for video anomaly detection. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1933–1941.

[Zhong Hong, 2024] Zhong Hong, M. (2024). Real-world drl: 5 essential reward functions for modeling objectives and constraints. https://medium.com/@zhonghong9998/real-world-drl-5-essential-reward-functions-for-modeling-objectives-and-constraints-e742325d4747: :text=Dense Accessed: (August 9, 2024).

[Zhu et al., 2023] Zhu, Z., Lin, K., Jain, A. K., and Zhou, J. (2023). Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Zucker et al., 2015] Zucker, M., Joo, S., Grey, M. X., Rasmussen, C., Huang, E., Stilman, M., and Bobick, A. (2015). A general-purpose system for teleoperation of the drc-hubo humanoid robot. *Journal of Field Robotics*, 32(3):336–351.