Effective Transfer Learning with the Use of Distance Metrics

by

Shiva Soleimany Dizicheh

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Reinforcement learning (RL) offers agents a framework for learning to perform hard-to-engineer behaviors that other machine learning (ML) approaches cannot due to the complex nature of these problems [1]. However, it is impractical to learn a complex task from scratch due to reasons such as the huge sample complexity of RL algorithms, experience feasibility in dangerous setups, or the need for long periods of training in order for the algorithms to converge [1]. The agent's training can further be hindered by the great difficulty of the target task, poor state representation, or sparse reward signals [1].

Transfer learning is the area of research concerned with the class of methods that seek to speed up the training of RL agents by transferring the knowledge that the agent has gained through one or more source task Markov decision processes (MDP) to the target task [2], [3]. Transfer learning can eliminate the need for training from scratch every time the environment changes slightly and help the agent to make use of its past experiences in similar domains [4]. However, transfer learning may inadvertently hurt the target performance, a phenomenon known as negative transfer [5]. Therefore, having a metric to approximately measure the similarity between the source task and the goal task can help us to pick our source task more wisely and perform better on the goal task. The transfer learning literature includes different metrics to measure the level of similarity between MDPs [2], [6], [7]; among them are distance metrics based on the averaged difference between the corresponding state-action transition distributions of the two tasks [6], or based on graph-similarity

between the graphs representing the transition and the reward functions of the source task and the target task [7] [8].

In this work, we look into three similarity metrics and their ability to estimate the similarity between two MDPs. All three metrics are based on the distance of state-action spaces between two MDPs. The first two metrics are based on the transitions spaces, but they focus on the action space and the state space separately. The third metric focuses on the difference between the immediate reward values of the state-action pairs in the source and target tasks. After pre-training on source tasks and then performing transfer learning, we look into the predictive capacities of each metric of the agent's performance on the goal task in two OpenAI gym domains: Hopper and Pendulum. The thesis is organized in a matter to first to present the needed background about the used algorithms and environments, then after explaining the process for calculating each metric, present the result of the experiments and analyze them. In the last part of the thesis, we pose questions as guidelines for future works in order to find the reason for the observed results.

To my parents,

For their endless love, support, and encouragement.

Acknowledgements

I would like to thank my supervisor, Dr. Matthew Taylor, for the invaluable insights he has provided throughout my M.Sc. as his student. I have been fortunate to have the chance to work with him, who encourages me and cares about my work. I also would like to thank professor Matthew Guzdial who helped me direct my research towards novel areas. I am further thankful to my parents and my sisters for providing me with constant love and support throughout my years of study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, first we state a short background and the motivation behind this work. After that, we mention some of the related works and how our work builds up on them and completes them. In Section 1.3, we explain the research questions that this thesis is trying to address. Finally, we explain the layout of the thesis.

1.1 Motivation

Each *reinforcement learning* (RL) task can be interpreted as a sequence of decisions that an agent should make, and it is further encoded using Markov decision processes (MDPs) [3]. These sequences of decisions represent the way that the agent will behave in each state and are formally called the learned policy [9]. The agent receives a reward after making decisions, and when the agent learns a policy that will maximize a long-term expected return for an MDP, we consider that MDP solved [3]. Even though RL is able to address problems that normal machine learning algorithms cannot, the curse of dimensionality (high number of possible states) when facing continuous state spaces of real-world tasks can result in a slow convergence rate or poor performance in RL agents [10]. One possible approach to tackle this problem is to use transfer learning.

The core idea of *transfer learning* (TL) is that experience gained in learning to perform one task can help improve learning performance in a related but different task [2]. Sometimes training the agent on the final task is time-

consuming or infeasible, dangerous, or expensive. In these situations, we would like to eliminate or limit the needed training on the goal task (MDP^*). In addition to making this goal possible, TL algorithms can also remove the need for training the agent again after every slight change in the goal task [4]. The reasons for using TL algorithms are not limited to these cases. In other cases, the goal of transfer learning might be introducing the agent to variable experiences to improve its overall performance in the goal task [11]. Transfer learning algorithms can be categorized based on many factors. One approach for classifying the TL algorithms is to focus on the type of transferred knowledge [2]. The transferred knowledge can be low-level information (like the action-value function) or high-level information (such as rules regarding how a particular domain functions) [2]. The initialization of an agent in the target task is directly possible with low-level knowledge. In contrast, higher-level knowledge, such as partial policies, is not usually used as a full policy in the target task, but rather as a starting point or a guideline for the agent during its training time [2].

Before performing transfer learning, it is important to determine the similarities and differences between the source task and the goal task. For instance, pre-training an agent in a source task with the goal of falling down might not be helpful for a goal task of jumping. The source task and the goal task can be different in various aspects. For example, they might differ in their transition functions, state spaces, or reward functions [12]. An example of two environments with different transition functions can be jumping up in environments with different gravity values. In this case, since the physical rules of the environments are different, the agent will end up in different states even when starting in the same states and taking the same actions.

Transfer learning can improve the agent's performance on the goal task by different degrees. The degree of performance improvement in MDP^* varies depending on the similarity between the source task and the goal task. The more similar the source and the goal task, the higher the transfer success would be [2]. But how exactly can we measure the similarity between tasks? Similarity metrics, or their complementary metrics, distance metrics can help

us address this problem by approximating the similarity between a source task and a goal task. The two main categories of distance metrics are 1) model-based and 2) performance-based [13]. Performance-based metrics, as it can be guessed from their name, compare the performance of the agent in the source and goal tasks. Model-based metrics compute the level of similarity between the source task and the goal task by measuring the similarity between their respective MDP models (e.g., states, actions, transitions, and rewards dynamics) [13]. Defining good distance metrics enables robust transfer and can also be used to generate new source tasks or change the existing source task's parameters to make it more similar to the goal task [14].

In short, distance metrics are useful measures both for transferring from source tasks to target tasks and generating or changing the source tasks to increase the reuse gain.

1.2 Existing Techniques and Drawbacks

Researchers have developed several techniques for providing a measure of distance between tasks [7], [15], [16]. As mentioned in the previous section, distance metrics can be divided into two categories: 1) Model-based and 2) Performance-based [13].

In performance-based distance metrics, agent performance can be defined as (i) the policies that an agent learns while performing these tasks or (ii) the benefit that an agent gains by reusing the knowledge gained from a source task in a target task [13]. Among performance-based similarity metrics, Mahmud et al. [15] proposes a similarity metric based on the reuse gain an agent gets by reusing the knowledge from that source task in the target task. Unfortunately, in most cases, we cannot calculate the reuse gain until after the transfer has taken place. Model-based approaches do not require training or transferring to the goal task in order to offer us a measure of similarity. For instance, Kuhlmann et al. [7] assumes knowledge of the full model of the MDP to construct a graph and look for the most similar source task by looking for isomorphic rule graphs. Even though this similarity metric can be

calculated without any training on MDP^*, full knowledge of the MDP model is not always available. Ammar et al. [17] does not assume knowledge of the environment and measures the similarity metric merely by comparison of the agent's interactions with the environment in the source task and the target task. The comparison of the experience tuples is performed by using restricted Boltzmann machines (RBMs). The downside of this approach is that it is computationally expensive since it requires training a generative model, and it is also dependent on exploration [18].

Another group of metrics are based on the distance between transition spaces [6], [16], [19]. For instance, Taylor et al. [19] introduces a distance metric as the Euclidean distance between state-action pairs in the source and target tasks. Carroll et al. [16] introduces another distance metric by focusing on the mean squared error between the immediate reward values of the state-action pairs in the source and target tasks. Even though similarity metrics have been introduced based on the Euclidean distance of rewards and the Euclidean distance of actions, no similarity metric has been introduced based on the Euclidean distance of the next states (s').

1.3 Contributions

The preceding motivation underlies the work in this thesis. Our contributions are summarized as follows:

1. We propose a new distance metric, called the next state distance (d_s). The distance is defined as the Euclidean distance between the next states (s') for corresponding state-action pairs in the source task and the target task. This distance metric is model-based and does not require any training on the goal task. The concept of this distance is easy to comprehend, and no model training or complex function definitions is needed to calculate this distance metric. Additionally, to compute this metric, no knowledge of the environment is necessary.

2. A statistical analysis of the correlation between the next state distance

and three transfer evaluation measures mentioned below. In order to evaluate the ability of our metric in measuring the similarity between two tasks, first we look at the Pearson correlation between this metric and each transfer evaluation metric and then the cost of picking the best MDP (for that evaluation measure) based on this metric.

(a) Jumpstart (initial performance boost on the goal task).

(b) Asymptotic performance (final performance on the goal task).

(c) Time to threshold (agent's needed learning time to reach a predefined performance level).

3. We compare the next state distance with two other distance metrics that are also calculated based on the Euclidean distance between the transition spaces of the agent in the source and target tasks to answer the following questions: Which distance metric should we use if

(a) we care about the initial performance of the agent in the goal task and we are allowed to train on the goal task

(b) we care about the initial performance of the agent in the goal task and we are not allowed to train on the goal task

(c) we care about the agent's final performance in the goal task and we are allowed to train on the goal task

(d) we care about the agent's final performance in the goal task and we are not allowed to train on the goal task

(e) we care about the agent's speed of learning in the goal task and we are allowed to train on the goal task

(f) we care about the agent's speed of learning in the goal task and we are not allowed to train on the goal task

1.4 Thesis Structure

This thesis consists of 5 main chapters. After Chapter 1, the introduction, in Chapter 2, we provide the reader with the necessary information about the background and related work for understanding this thesis. We explain the layout of our experiments and the process of calculating the distance metrics under investigation in Chapter 3. Afterward, Chapter 4 first explains our experimental setup and the two OpenAI gym domains in which our experiments were carried out. Then in the second part of this chapter, we mention the evaluation measures that we have used to analyze the results of our experiments. In the last section of Chapter 4, we present the results of our experiments for comparing the predictive abilities of three distance metrics and their analysis. Finally, in Chapter 5, we conclude our thesis with an overview of our proposed distance metric's performance compared to two other distance metrics.

Chapter 2

Background and Related Work

In this section, we provide the reader with the necessary information for understanding this thesis.

2.1 Reinforcement Learning

A *reinforcement learning* (RL) task can best be described as a sequential decision-making setup that consists of an agent interacting with an environment by taking actions and receiving rewards and observations in return. In contrast to a supervised learning setting that deals with a set of labeled examples, the RL agent is not provided with any labels: it has to accomplish the goal in an uncertain and possibly complex environment by trying different actions and learning which action gives rise to the maximum reward. The sequential interactions of the agent with the environment can be formulated as a Markov decision process (MDP) [20]. At each time step t, the agent causes the environment to transition according to the transition probabilities $T(s, a, s') = Pr(s' \mid s, a)$ from state $s \in S$ to a new state $s' \in S$ by taking an action $a \in A$. For this transition, the agent receives a reward according to the reward function $R(s, a)$, determining the reward for taking action a in state s. Furthermore, it is possible to encourage the agent to prioritize immediate reward over latent reward by discounting rewards more and more as they happen further and further in the future by using the discount factor, γ. Such an MDP is described by the tuple $\langle S, A, T, \gamma, R \rangle$.

The mapping that helps the agent to decide what action to take at each

state is called a policy, and the agent learns it through trial and error. A policy can be deterministic or stochastic. A deterministic policy π is a mapping from states to actions, $\pi : S \to A$, that is, for each state, s, $\pi(s)$ returns an action, $a = \pi(s)$ with probability of 1, i.e., $\pi(s) = 1$. Another function on states and actions is the state-action value function $Q^\pi(s, a)$. $Q^\pi(s, a)$ is defined as the expected sum of discounted rewards (using the discount factor γ) the agent will receive if it takes action a in state s and follows the policy π from that point on and it is formulated based on equation (2.1):

$$Q^\pi(s, a) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}) \mid s_t = s, a_t = a, \pi]. \tag{2.1}$$

The action value function for any given policy π satisfies the Bellman equation:

$$Q^\pi(s, a) = R(s, a) + \gamma \mathbb{E}_{s',a'}[Q^\pi(s', a')],$$

where s' is the state at the next time step and a' is the action the agent takes on the next time step. The goal of agent is to maximizes the expected sum of discounted rewards by finding the optimal policy denoted by π^*. Given the optimal value function $Q^*(s, a)$, the agent can retrieve the optimal policy π^* by acting greedily with respect to the optimal value function:

$$\pi^*(s) = \operatorname*{argmax}_{a \in A} Q^*(s, a),$$

where $Q^*(s, a)$ is defined as:

$$Q^*(s, a) = \max_{\pi \in \Pi} Q^\pi(s, a)$$

and Π is the set of all possible policies.

The Bellman equation for the optimal policy π^* is called the Bellman optimality equation:

$$Q^*(s, a) = R(s, a) + \gamma \mathbb{E}_{s',a'}[Q^*(s', a')].$$

Value-based RL algorithms try to learn the optimal value function Q^* iteratively, versus policy gradient algorithms try to learn the optimal policy π^*

directly. Actor-Critic methods are a combination of policy learning and value learning, i.e., the agents learns two models: the value function model and the policy model. The policy function plays the role of the actor: it picks what moves to play. The value function is the critic: it tracks whether the agent is ahead or behind in the course of the game. That feedback guides the training process.

When learning iteratively, it is crucial for a good policy to explore and discover new states and actions along with exploiting the knowledge it has already gained.

2.2 Twin Delayed Deep Deterministic Policy Gradients (TD3)

Policy gradient methods reformulate maximizing the expected return as minimization of a loss function $L(\theta)$ where θ encapsulates the agent parameters. Deep Deterministic Policy Gradient (DDPG) [21] is a policy gradient method often used in domains with continuous and high dimensional actions spaces. DDPG is widely-used and occasionally it can achieve great performance, but may require significant hyperparameter tuning. One problem with DDPG is the well-known overestimation problem as the result of the Q-function dramatically overestimating Q-values until eventually causing the policy to break. Fujimoto et al. [22] extended DDPG to Twin Delayed DDPG (TD3) in order to alleviate this problem using tricks such as learning two Q-functions and updating the policy less frequently.

TD3 was shown to significantly improve upon DDPG [23]. It is an algorithm for model-free deep reinforcement learning in continuous action spaces that is off-policy. The behavioral policy of TD3 is its target policy in addition to some added noise actions at training time, typically uncorrelated mean-zero Gaussian noise. The reason behind exploring off-policy in TD3 is its deterministic nature which can give rise to the agent not trying enough variety of actions.

TD3 uses an actor-critic architecture [3] that includes an actor with a

deterministic policy: $\pi : \mathcal{S} \rightarrow \mathcal{A}$, and two critics with distinct action-value function approximations: $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}_i$. Each critic acts independently in approximating the actor's action-value function Q^π. Neural networks are used as the representation of the actor and θ^π, θ^Q_a, and θ^Q_b encapsulate the parameters of the NNs.

2.3 Transfer Learning

Reinforcement learning offers to robotics agents a framework for learning to perform hard-to-engineer behaviors that other ML approaches cannot due to the complex nature of this domain [1]. Inspired by how humans learn through trial-and-error processes, RL algorithms acquire their knowledge based on the rewards that agents obtain when they act in certain manners given different experiences [3]. This requires a large number of episodes, which introduces limitations regarding time, variability in experience, and experience feasibility in some dangerous environments [1]. The agent's training can further be hindered by the high difficulty of the target task, poor state representation, or sparse reward signals [1]. Transfer learning is the area of research concerned with the class of methods that seek to speed up the training of RL agents by transferring the knowledge that the agent has gained through one or more source task MDPs to the target task [2], [3]. Transfer learning can eliminate the need for training with every slight change in the environment and help the agent to make use of its past experiences in similar domains [4].

Transfer learning is the problem of how to obtain, represent and, ultimately, use the previous knowledge of an agent [24]. Transfer learning algorithms can be categorized based on the the difference between the source task(s) and the target task. Source tasks can be different from the target task in state spaces, action spaces ,start states, goal states, transition probabilities and the reward functions [12]. Another way to categorize TL algorithms is based on the type of information that is being transferred. For instance, we can choose to transfer a set of expert experiences, or the policy π learned in the source task. It is not always easy to pick the best transfer learning algorithm. Choosing the transfer

learning algorithm depends on how much and how the source and target tasks are related [2]. After choosing the TL algorithm and performing the transfer, we need metrics to measure the benefits of transfer. These metrics are usually: Jumpstart, asymptotic performance, total reward, transfer Ratio, and time to threshold [2].

1. Jumpstart: the initial performance boost of an agent in a target task as as result of transfer learning .

2. Asymptotic Performance: agent's final performance in the target task (reward).

3. Total Reward: agent's total reward can be increased as a result of using transfer compared to learning from scratch and without transfer.

4. Transfer Ratio: The ratio of the total reward of an agent with and without transfer learning.

5. Time to Threshold: the agent's needed learning time in order to achieve a pre-specified performance level can be decreased by knowledge transfer.

In this work we use jumpstart, asymptotic performance and time to threshold as our evaluation metrics. They are explained in more details in Section 4.2.1.

When presented with multiple source tasks, it is a common approach to choose the most "similar" task for transferring from hoping that it will result in the most positive transfer. Therefore, the ability to correctly assess the degree of similarity between the sources and target tasks is important. Different similarity metrics have been introduced in attempt to measuring how related two tasks are. They are briefly described in the next section.

2.4 Similarity and Distance Metrics

Similarity metrics, or their complementary distance functions, provide a measure of distance between tasks. There are various similarity metrics that use

different techniques and algorithms to measure that distance and the choice of a metric depends on the nature of tasks. The distance metric can be used to measure the distance between the source tasks and the target task in transfer learning and ultimately, singling out the source task that has the smallest distance from the target task as the best task (most similar) for transfer [13].

Assuming we have two tasks, $M_i = \langle S_i, A_i, T_i, R_i \rangle$ and $M_j =< S_j, A_j, T_j, R_j >$, that can share the state space, the action space, or the transition and reward dynamics, then the distance between M_i and M_j is defined as:

$$d(M_i, M_j) = [0, \infty) \tag{2.2}$$

and the smaller the distance $d(M_i, M_j)$, the greater the positive transfer from M_i to M_j is.

If we have a goal MDP named M_j, and two source MDPS, M_i and M_k, we expect d to have the following properties:

1. If $d(M_i, M_j) < d(M_k, M_j)$, transferring to M_j from M_i will produce the greatest positive transfer than transferring from M_k,

2. $d(M_i, M_j)$ should be computable before or during the transfer

Distance metrics can be divided into two main categories: 1) model-based and 2) performance-based.

Performance-based metrics compare the performance of the agent in the source task and the target task. Agent's performance may be defined as (i) the policies learned by the agents in these tasks, or (ii) the reuse gain an agent gets by reusing the knowledge from a source task in a target task. In case (i), the similarity metrics measure the similarity between the behavioral policies learned in the source task and the target task. The similarity between two policies can be calculated based on their policy values, such as their action-value functions Q^π [16], or their policy approximation parameters such as their parameter vector θ that approximates the action-value function Q^π [25]. Although distance metrics based on reuse gain offer a measure of similarity between two tasks, reuse gain can only be calculated after transfer learning

has been done. When our goal is to choose a task to transfer from, producing a measure of similarity after learning is too late to be useful [16]. Even though distance metrics based on policy similarity offer a measure of similarity before transfer learning is done, they require full or partial training on the target task. This can be a disadvantage when training on target tasks that are time-consuming or infeasible.

Model-based metrics do not require training on the goal task. In model-based metrics, the degree of similarity between a source task and a target task is measured by using their respective MDP models (e.g., states, actions, transitions, and rewards dynamics). There are different model-based metrics depending on which components of the MDPs are considered. Some works construct graphs representing the transition and the reward functions of both the source and target tasks and try to find the most similar source task by looking for isomorphic rule graphs [7]. This approach assumes that the full model of the MDP is given in order to build the rule graph. In contrast, one approach calculates the distance metric merely by comparison of the experience gathered by interacting with the environment in the source task and the target task. This distance metric is small if the generated experience tuples in the source and target task are similar, and the comparison of the tuples is performed by using restricted Boltzmann machines (RBMs) [17]. The problem with this approach is that it is computationally expensive since it requires training a generative model, and it is also dependent on exploration [18].

In this work, we focus on metrics that do not require any domain knowledge and are calculated simply as the Euclidean distance between actions, rewards, or states. All three metrics are computed based on the experience that the agent gathers by interacting with the environment, and they do not require having any knowledge of the model of the MDPs:

- Action distance (d_a): is a performance-based metric, and it is defined as the Euclidean distance between the corresponding state-action pairs in the source and target task [19].

- Reward distance (d_r): the third metric is a model-based distance metric,

and it is defined as the mean squared error between the immediate reward values of the state-action pairs in the source and target tasks [16].

Which distance metric is the best? Answering this question depends on the factors such as the type of tasks and the transfer techniques that the agent uses [16]. This doesn't mean that task distance metrics are not useful but rather that when we talk about the "best" distance metric, we must pay attention to the task, the transfer technique, and the type of reuse gain in the goal task that we are interested in (jumpstart, asymptotic performance, etc.).

Chapter 3

Our Approach

Previous works have shown that there is often a correlation between the similarity of the previous tasks of the agent and the goal task and the efficacy of transfer learning [17]. In this work, we propose three novel similarity metrics and analyze whether there is a correlation between transfer learning effect and similarity based on these metrics.

3.1 Overview

Training RL agents from scratch is not always possible, easy, or time-efficient. One solution to this problem is to use transfer learning to remove or reduce the required training on the goal MDP by first pre-training the agent on other simulated environments and transferring the learned knowledge to the main task. However, selecting among the simulated MDPs can be challenging. As a possible solution, this work proposes three similarity metrics to measure the similarity between the goal MDP setting and the other simulation MDP settings. The goal is to see whether there was a correlation between transfer learning effect and similarity.

3.2 Action Distance (d_a)

This distance can be thought of as the similarity of trained policies on two different MDPs. Action distance measures the distance between actions that two policies, Π_x, Π_y, trained on two distinct MDPs, MDP_x, MDP_y, take given

the same state. For measuring this similarity metric for each MDP:

1. Train an agent on MDP_x and save the policy Π_x.
2. Repeat 15 times with 15 different environment seeds.
3. Train an agent on MDP_y and save the policy Π_y and the transitions $T(s_y, a_y, s'_y, r_y)$ for the last 100 timesteps.
4. Repeat 15 times with 15 different environment seeds.
5. Load Π_x on the saved states, s_y, and save the actions that Π_x chooses, a_x.
6. For each of the 15 runs, calculate the average Euclidean distance between a_y and a_x for 100 timesteps. Actions are 3-dimensional vectors, but the range is the same for each dimension of the action vector; therefore, no normalization is needed.

In order to be able to analyze the observations isolated from the effect of training time, we perform steps 1 to 6 in three different stages: after training on MDP_x and MDP_y for 1 hour, 2 hours, and 3 hours.

3.3 Next State Distance (d_s)

is a model-based metric that calculates the Euclidean distance between the agent's immediate state for state-action pairs in the source and the target tasks. This metric is a simple version of Bisimulation metrics. Bisimulation metrics compute the distance between two states by comparing their transition and reward dynamics. In other words, in order for two states to be bisimilar, all of the transitions of one state must be matched by the transitions of the other state, and the results should also be similar. Bisimulation is a very strong notion because even a slight change can cause two states to not be bisimilar anymore[26]. In this work, we calculate our second distance metric, d_s, based on the immediate state after a single transition from each state. If policy Π_x in MDP_x on state s_1 takes action a_1 and ends up in s'_x, and policy

Π_y in MDP_y on state s_1 takes action a_1 and ends up in s'_y, how different s'_x and s'_y are. For measuring this similarity metric for each MDP:

1. Train an agent on MDP_x and save the policy Π_x.
2. Repeat 15 times with 15 different environment seeds.
3. In MDP_y, interact with the environment for 100 timesteps and save and the transitions $T(s_y, a_y, s'_y, r_y)$.
4. Repeat 15 times with 15 different environment seeds.
5. in MDP_x, load Π_x on the saved states, s_y, and take the actions a_y consecutively, saving the transitions $T(s_y, a_y, s'_x, r_x)$.
6. States are 11 dimensional, and the range of each dimension varies. In order to get more unified results, apply normalization to scale the range of each dimension to [0, 1]. For each of the 15 runs, calculate the average Euclidean distance between s_y and s_x for 100 timesteps.

In order to be able to analyze the observations isolated from the effect of training time, we perform steps 1 to 6 in three different stages: after training on MDP_x for 1 hour, 2 hours, and 3 hours.

3.4 Reward Distance (d_r)

If policy Π_x in MDP_x on state s_1 takes action a_1 and ends up in s'_x and gets reward r_x, and policy Π_y in MDP_y on state s_1 takes action a_1 and ends up in s'_y and gets reward r_y, this similarity metrics measures the difference between r_x and r_y. Reward distance measures the difference between rewards when two agents start from the same state and take the same action. This similarity metric is another way to look into the difference between the physical characteristics of environments. For instance, in Hopper, the reward value depends on how far the Hopper jumps in each timestep. Since the starting state and the action are the same, reward distance depends on the next state

that the agent ends up in. However, this metric also depends on the equation of the reward function. For measuring this distance metric:

1. Train an agent on MDP_x and save the policy Π_x.
2. Repeat 15 times with 15 different environment seeds.
3. In MDP_y, interact with the environment for 100 timesteps and save and the transitions $T(s_y, a_y, s'_y, r_y)$.
4. Repeat 15 times with 15 different environment seeds.
5. in MDP_x, load Π_x on the saved states, s_y, and take the actions a_y consecutively, saving the transitions $T(s_y, a_y, s'_x, r_x)$.
6. Rewards are one-dimensional scalar numbers, and no normalization is needed. For each of the 15 runs, calculate the average Euclidean distance between r_y and r_x for 100 timesteps.

In order to be able to analyze the observations isolated from the effect of training time, we do steps 1 to 6 in three different stages: after training on MDP_x for 1 hour, 2 hours, and 3 hours.

Chapter 4

Experiments and Results

In this chapter we first explain the settings of our experiments, then we mention the evaluation measures that we use for evaluating the transfer performance and out distance metrics. In the last part we present the results of our experiments.

4.1 Experimental Setup

To test the efficacy of our similarity metrics we perform our experiments in two domains: Hopper and Pendulum. These environments and our experiment settings are explained in this section.

4.1.1 Hopper

For this task, the states are represented by 11 dimensional vectors including parameters such as: positions (in terms of radiant or meters), and sin and

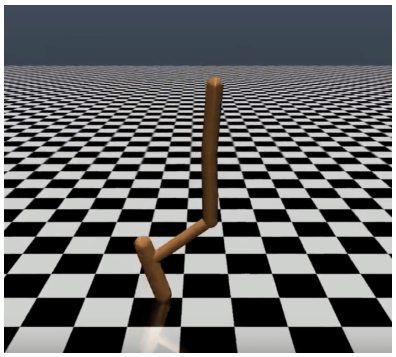

Figure 4.1: OpenAI Hopper Environment

Table 4.1: Environment Configuration in the Hopper Task

Leg Size \Gravity	2.45	4.9	9.8	19.6
0.01	MDP1	MDP5	MDP9	MDP13
0.03	MDP2	MDP6	MDP10	MDP14
0.06	MDP3	MDP7	MDP11	MDP15
0.12	MDP4	MDP8	MDP12	MDP16

cos functions of joint angles. The action space corresponds to a 3-dimensional space where each action is a continuous value that is bounded in the range $[-1, 1]$. The action values correspond to torques of the thigh joint, the leg joint and the foot joint.

The goal in this environment is to make the hopper move forward, therefore the agent reward function includes: 1) being alive bonus (b), a positive contribution for the forward velocity (calculated by taking the derivative of the movement at each step), 3) a negative contribution of the Euclidean norm among the action control space. This reward (r) is computed as:

$$r = \begin{cases} v_x - 0.0001||a|| + b & \text{otherwise} \\ -1 & \text{failures} \end{cases} \tag{4.1}$$

where a represents the actions calculated by the neural networks, and v_x is the forward velocity. Failure happens when at least one of the failure conditions occur. These conditions are:

$$\begin{aligned} Z_{torso} &< 0.7 \\ |\theta| &< 0.2 \end{aligned} \tag{4.2}$$

where θ is the forward pitch of the body and Z_{torso} is the height of the torso. The episode ends upon failure.

The Hopper itself and the environment have some characteristics that can be changed. For instance, gravity, friction, leg length, and leg size. We created 16 different environments by changing two of these variables: gravity and leg size. The values of gravity and leg size for each MDP are mentioned in Table 4.1.

Figure 4.2: OpenAI Pendulum Environment

Table 4.2: Environment Configuration in the Pendulum Task

Mass \Gravity	2.5	5	10	20
0.1	MDP1	MDP5	MDP9	MDP13
0.3	MDP2	MDP6	MDP10	MDP14
0.4	MDP3	MDP7	MDP11	MDP15
0.8	MDP4	MDP8	MDP12	MDP16

4.1.2 Pendulum

The purpose of this task is to maintain upward facing equilibrium over the vertical axis of the pendulum by applying torque to its central actuator. In other words, the goal is to remain at zero angle (vertical), with the least possible effort. For this task, the states are represented by 3 dimensional vectors including: cosine and sine of the angle and the derivative of the angle. The action space is 1-dimensional vector bounded in the range $[-2, 2]$ corresponding the torque applied to the joint. The reward function is:

$$-(\theta^2 + 0.1 * \theta_d t^2 + 0.001 * action^2)$$

θ is normalized between $-\pi$ and π. Therefore, the lowest reward is $-(\pi^2 + 0.1 * 8^2 + 0.001 * 2^2) = -16.2736044$, and the highest cost is 0.

The Pendulum itself and the environment have some characteristics that we can change. We created 16 different environments by changing two of these variables: gravity and mass. The values of gravity and mass for each MDP are mentioned in Table 4.2.

4.2 Evaluation Measures

While there are many metrics proposed in the literature, we focus on three metrics to measure the benefits of transfer learning, namely jumpstart, asymptotic performance, and time to threshold. They are sufficient to describe the entire training timeline of the methods experimented in this work. The third metric charges the agent for the time spent learning the source task and the first two consider initial and final performance.

4.2.1 Transfer Learning Evaluation Measures

Jumpstart

The initial performance of an agent in a target task may be improved by transfer from a source task. The first transfer measure involves taking into account an agent's performance at the beginning of a target task to evaluate how much increase in initial performance is achieved relative to an initial random policy. However, such a metric does not capture how the learner behaves during the target task and focuses on performance before the learning begins.

In this work, we calculate the jumpstart by averaging over the first 100 episodes and deducting the value of the first 100 episodes when running a policy without any pre-training.

Asymptotic Performance

Asymptotic performance, the second metric, compares the final performances (reward) of learners in the target task with and without transfer. However, it may take too long for a learner to converge or there may be difficulties in determining when the learner has actually converged (particularly in tasks with infinite state spaces). Additionally, different learning algorithms can reach the same asymptotic performance but differ in number of samples required to reach it. Therefore we also require time to threshold metric that evaluates training behaviour.

In this work, asymptotic performance is measured by averaging reward over the last 100 episodes.

Time to Threshold

The final metric, time to threshold, is the learning time needed by the agent to achieve a pre-specified performance level which requires specifying a (potentially arbitrary) performance requirement. Some have suggested how to choose these thresholds [12], but it is clear that the relative benefits of transfer learning methods will vary depending on the exact threshold chosen, which will depend on the domain and learning-method.

In this work we calculate the time to threshold by calculating the number of timesteps needed to reach the average reward of 2500 for Hopper and -500 for Pendulum and staying at that value for 10,000 timesteps. We then scale the values of needed timesteps by dividing them by 5000.

4.2.2 Distance Metrics Evaluation Measures

When we make a decision to do transfer learning, often times we have several source tasks available for pre-training and we must select one or a few among them. This work introduces three distance metrics for helping with this selection. However, when we calculate the value of distance metrics for each source task, we face another question: which metric should we use to pick the source tasks? In other words, we have distance metrics to help us compare MDPs, but we still need a measure to compare the distance metrics since it is not not immediately obvious which distance metric is better to use. In this work two different approaches have been used for distance metrics comparison which are mentioned in this section.

Cost of picking the best MDP

A perfect distance metric would be able to sort all source tasks from the best to the worst regarding their transfer learning potential, guaranteeing that the best MDP for pre-training always has the smallest distance value. However, given the complexity of RL tasks, such distance metric has not been discovered

yet. However, often it is a realistic expectation from a distance metric to help us pick the best MDP with a bigger probability than random choice. For instance, if we have 20 source tasks, the probability of choosing the best MDP randomly is 5%. If the best MDP is in the top five MDPs based on a given distance metric, the probability of picking the best MDP with the help of that distance metric increases to 20%, meaning using this metric is better than random selection. This approach can also be used for comparison among distance metrics: the higher the probability of picking the best MDP based on a metric, the better the metric is. In this work, we present the results for this approach not as the probability of picking the best MDP, but as the lowest number of top MDPs that we have to pick the best MDP among them. The correlation between the cost of picking the best MDP and the predictive ability of an MDP are reversed, meaning the cost of picking the best MDP based on a very good distance metric is small and ideally close to 1.

1. Cost of picking the best MDP in each trial: the first row is the average cost of picking the best MDP in each trial. All of the experiments in this thesis are repeated 15 times to make sure that the results are not arbitrary. When trying to figure out which MDP to pre-train on to get the highest jumpstart (or asymptotic performance), in each trial, the order of MDPs based on the distance metric and the best MDP might vary. For example, in trial 1, our maximum jumpstart might be achieved when transferring from MDP11, which has the third-smallest value for action distance; in trial 2, transferring from MDP10 with the second smallest value of action distance might result in the biggest jumpstart boost. We define the cost of picking the best MDP in these trials 3 and 2 consecutively. By averaging these numbers over 15 trials, we can reach an approximate estimate of the predictive value of a distance metric.

2. Cost of picking the best overall MDP: when averaging over all the 15 trials, one of the source tasks will prove the most beneficial. Even though this source task might not be the best in every trial, on average, it results in better transfer learning than others. Additionally, we can average the

distance metrics of source tasks over 15 runs to achieve a more robust estimation of the distance of MDPs to MDP^*. The second row is the cost of picking the best MDP over 15 runs. Although this number might represent a better estimate of the quality of a distance metric, we usually do not want to repeat the experiments several times to find the best MDP. Therefore both cost of picking the best MDP in each trial, and the cost of picking the best overall MDP matter to us.

Pearson correlation coefficient

Developed by Karl Pearson, the Pearson correlation coefficient or the Pearson product-moment correlation coefficient, measures linear correlation between two sets of data. This ratio is the product of the covariances of two variables to the product of their standard deviations, making it essentially a normalised measurement of covariance, such that it always has a value between -1 and 1.

Given two random variables, Pearson's correlation coefficient formula, commonly represented by the Greek letter ρ, is:

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{4.3}$$

where σ_X and σ_Y are standard deviations for X and Y, respectively, and cov is the covariance.

4.3 Overall Results

In this section we present the results of our experiments. We first evaluate the predictive ability of all three distance metrics in predicting jumpstart and then we do the same for asymptotic performance and time to threshold.

Throughout this section, when we mention that the best MDP is in the top n MDPs, where $n \in [1, 16]$, we mean there are $n-1$ source tasks that: 1) they have smaller distance metric values than the best MDP, 2) pre-training on them results in less performance improvement compared to pre-training on the best MDP. Therefore, picking these MDPs is the cost we have to pay in

order to pick the best MDP (the MDP that pre-training on it will result in the most performance improvement). This number is represented in the figures as the number of dots in the green shaded area. These dots are MDPs who have smaller distance metric values, but transferring from them is not as beneficial as transferring from the best MDP.

4.3.1 Jumpstart correlation with Distance Metrics

Jumpstart, defined in Section 4.2.1, is the difference between the initial performance of an agent in MDP^* with and without pre-training. In this section, we evaluate the ability of each distance metric in predicting the jumpstart in MDP^* resulting from pre-training on each MDP. To evaluate the predictive ability of each distance metric, we calculate the Pearson correlation between that distance metric and jumpstart, and also the cost of picking the best MDP (the MDP that pre-training on it would result in the highest jumpstart) based on that metric. We use the expression of "moderate correlation" for coefficient values between 0.5 to 0.7 or -0.7 to -0.5, And the expression of "strong correlation" for coefficients with bigger absolute values than 0.7 [27]. Distance metrics are complementary to similarity metrics, therefore in an effective distance metric, the smaller the distance metric, the higher the jumpstart should be.

The experiments are performed in the Hopper domain after one, two, and three hours of training. Our algorithm takes 3 hours to fully converge, which in the Hopper task at code level it equals 900000 timesteps. As a result, we divided the training time into three equal parts, each lasting for one hour, or 300000 timesteps. In order to gain more confidence in the observations, the experiments are repeated in the Pendulum domain. However, due to the simpler nature of the Pendulum task, training for more than one hour does not affect the experiments' results, and therefore they are only included for the one-hour case. In the following sections, we evaluate the predictive ability of action distance, next state distance, and reward distance consecutively.

Action Distance

As explained in Section 3.2, action distance is a performance-based metric defined as the distance of actions for corresponding states in two MDPs. In this section, we aim to evaluate the ability of this metric in predicting the resulting jumpstart from pre-training on each of $MDP_{1...16}$. To accomplish this goal, we look into the Pearson correlation between action distance and the cost of picking the best MDP when using this metric.

1) The cost of picking the best MDP in each trial: the cost of picking the best MDP in each trial that has been explained in Section 4.2.2 is mentioned in the first and the third rows of Table 4.3 for Hopper and Pendulum. In the best case, action distance can place the best MDP among the top two MDPs. In the worst case, we need to pick the top five MDPs based on action distance to include the MDP that would result in the biggest jumpstart. The probability of choosing the best MDP among five MDPs is one in five or 20%. If we were to pick a random MDP among our 16 MDPs, the chances of the MDP that we have picked being the best MDP would be one in 16 or 6.25%. This can be interpreted as saying that using action distance is more than three times better than random selection among source tasks. In the Pendulum domain, this cost is 3.6, meaning we need to pick the top four MDPs to include the best MDP (regarding jumpstart).

2) The cost of picking the best MDP over 15 trials: The cost of picking the best MDP that has been explained in Section 4.2.2 is mentioned in the second and the fourth rows of Table 4.3. This value can be interpreted as the number of MDPs in the green shaded area, including the MDPs on the green border in Figure 4.4 and subfigures of Figure 4.3. In the best case, action distance can single out the best MDP in both Hopper and Pendulum domains. In the worst case, we need to pick the top five MDPs based on action distance to include the MDP that would result in the biggest jumpstart. The probability of choosing the best MDP among five picked MDPs is one in five or 20%. If we were to pick a random MDP among our 16 MDPs, the chances of the MDP that we have picked being the best MDP would be one in 16 or 6.25%. This

can be interpreted as saying that using action distance is more than three times better than random selection among source tasks.

3) The Pearson correlation between jumpstart and action distance: The blue line in Figure 4.4 and the sub-figures of Figure 4.3 represents the Pearson correlation between action distance and jumpstart. Additionally, the fifth row in Table 4.3, shows the value of the Pearson correlation coefficient in the Hopper domain after one, two, and three hours of training time. This value ranges between -0.7 to -0.76, which indicates a strong negative correlation between the value of action distance and the jumpstart on MDP^* [27]. This negative correlation exists more strongly in the Pendulum domain, with the correlation coefficient of -0.92, shown in the last row of Table 4.3.

Takeaway: Action distance has a strong negative correlation with jumpstart. Additionally, the cost of picking the best MDP based on this metric is low. Therefore, this metric can be helpful when choosing among source tasks. However, this metric is performance-based, and it requires training on the goal task, which might not be desirable or possible. Deciding to use this metric depends on the nature of the goal task along with other factors. The results for this metric will be compared to the results of the other two distance metrics at the end of Section 4.3.1 to find the best distance metric for predicting the jumpstart on MDP^*.

Next State Distance

As explained in Section 3.3, the Next state distance (d_s) is a model-based metric. This metric is the Euclidean distance between the agent's next state for corresponding state-action pairs in the source and the target tasks. In this section, we aim to evaluate the ability of this metric in predicting the resulting jumpstart from pre-training on each of $MDP_{1...16}$. To accomplish this goal, we look into the Pearson correlation between the next state distance and the cost of picking the best MDP when using this metric.

1) The cost of picking the best MDP in each trial: the cost of picking the best MDP in each trial that has been explained in Section 4.2.2 is mentioned in the first and the third rows of Table 4.4 for Hopper and Pendulum. In the worst

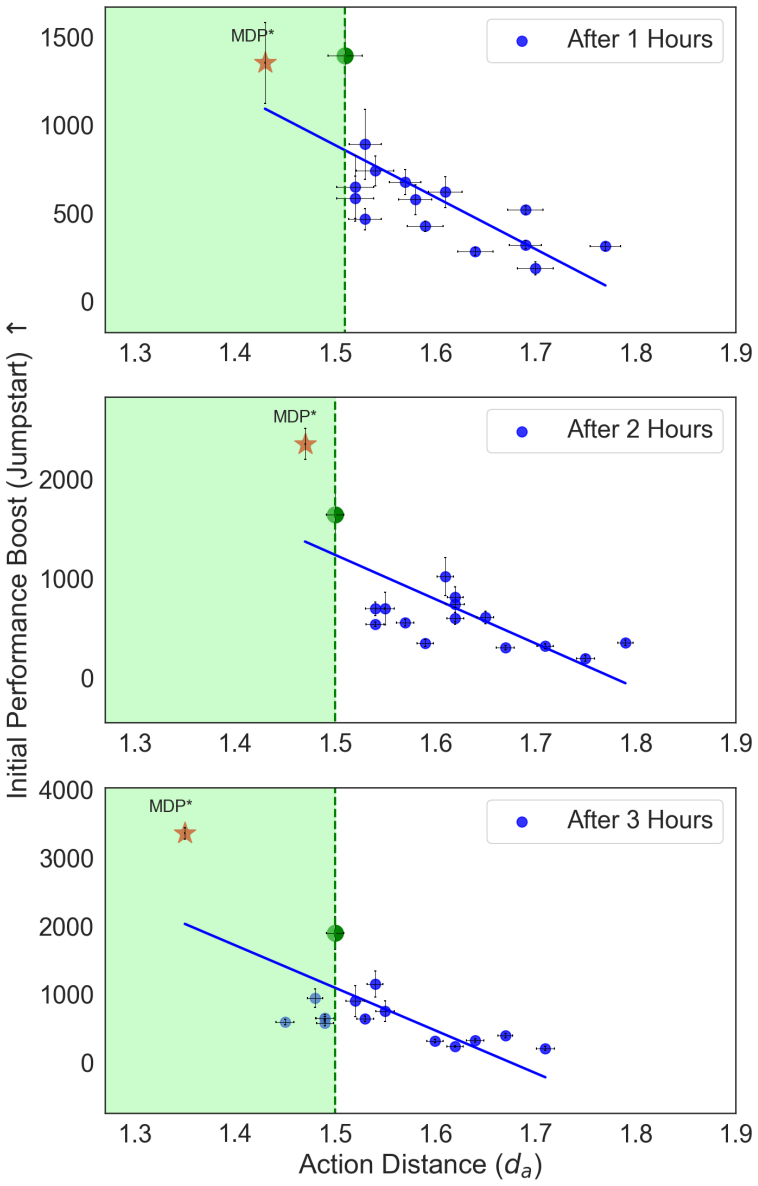

Figure 4.3: Hopper: Jumpstart on *MDP** versus action distance

The green dot: the MDP resulting in the biggest jumpstart. The dots in the shaded area: the MDPs with smaller action distances than the best MDP. Based on the strong negative correlation (blue line) between action distance and jumpstart, this metric can be helpful in choosing among source tasks.

Table 4.3: Jumpstart Correlation with Action Distance

Metric	Environment		Training Time(Hours)		
			1 h	2 h	3 h
Cost of picking the best MDP$^{\downarrow}$	Hopper	in each trial	3.2	1.9	4.8
		over 15 trials	1	1	5
	Pendulum	in each trial	3.6	-	-
		over 15 trials	1	-	-
Pearson Correlation Coefficient$^{\downarrow}$	Hopper	over 15 trials	-0.76	-0.70	- 0.72
	Pendulum	over 15 trials	-0.92	-	-

First four rows: Cost of picking the best MDP in each trial and over 15 trials. last two rows: the Pearson correlation between action distance and the jumpstart on MDP^*. The strong negative correlation and the low costs of picking the best MDP suggest that this metric can be helpful when choosing among source tasks. $^{\uparrow}/^{\downarrow}$ means the higher/lower, the better.

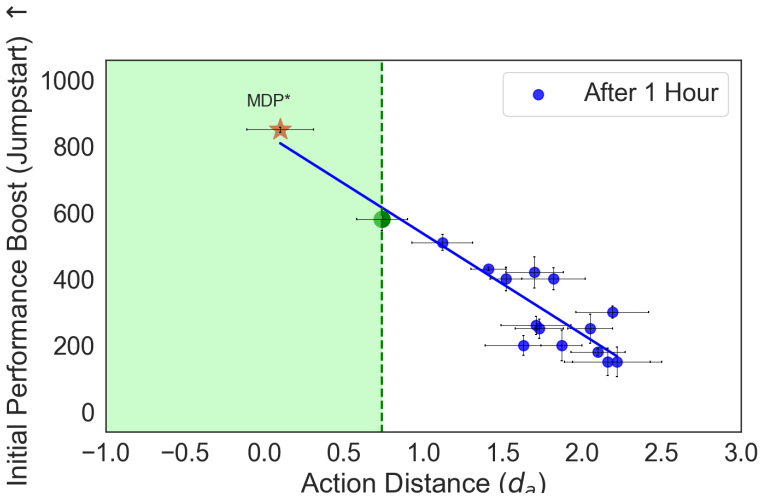

Figure 4.4: Pendulum: Jumpstart on MDP^* versus action distance

The green dot: the MDP resulting in the biggest jumpstart. The dots in the shaded area: the MDPs with smaller action distances than the best MDP. Based on the strong negative correlation (blue line) between action distance and jumpstart, this metric can be helpful in choosing among source tasks.

case, the cost of picking the best MDP in each trial is 6.3, so we need to pick the top seven MDPs based on the next state distance in order to include the MDP that would result in the biggest jumpstart. The probability of choosing the best MDP among seven picked MDPs is one in seven or 14.28%. If we were to pick a random MDP among our 16 MDPs, the chances of the MDP that we have picked being the best MDP would be one in 16 or 6.25%. This can be interpreted as saying that using action distance is more than two times better than random selection among source tasks. In the Pendulum domain, this cost is 3.6, meaning we need to pick the top four MDPs to include the best MDP (regarding jumpstart).

2) The cost of picking the best MDP over 15 trials: The cost of picking the best MDP that has been explained in Section 4.2.2 is mentioned in the second and the fourth rows of Table 4.4. This value can be interpreted as the number of MDPs in the green shaded area, including the MDPs on the green border in Figure 4.6 and subfigures of Figure 4.5. In the best case, the next state distance is able to place the best MDP among the top four MDPS. In the worst case, we need to pick the top seven MDPs based on the next state distance in order to include the MDP that would result in the biggest jumpstart. The probability of choosing the best MDP among seven picked MDPs is one in seven or 14.28%. If we were to pick a random MDP among our 16 MDPs, the chances of the MDP that we have picked being the best MDP would be one in 16 or 6.25%. This can be interpreted as saying that using action distance is more than two times better than random selection among source tasks.

3) The Pearson correlation between jumpstart and next state distance: The blue line in Figure 4.6 and the sub-figures of Figure 4.5 represents the Pearson correlation between next state distance and jumpstart. Additionally, the fifth row in Table 4.4 shows the value of the Pearson correlation coefficient in the Hopper domain after one, two, and three hours of training time. This value ranges between -0.7 to -0.46, which indicates a moderate to strong negative correlation between the value of next state distance and the jumpstart on MDP^* [27]. This negative correlation exists more strongly in the Pendulum

Table 4.4: Jumpstart Correlation with Next State Distance

Metric	Environment		Training Time(Hours)		
			1 h	2 h	3h
Cost of picking the best MDP $^{\downarrow}$	Hopper	in each trial	4.6	4.6	6.3
		over 15 trials	5	7	4
	Pendulum	in each trial	3.6	-	
		over 15 trials	2	-	-
Pearson Correlation Coefficient $^{\downarrow}$	Hopper	over 15 trials	-0.65	-0.78	-0.85
	Pendulum	over 15 trials	-0.79	-	-

First four rows: Cost of picking the best MDP in each trial and over 15 trials. last two rows: the Pearson correlation between next state distance and the jumpstart on MDP^*. The negative correlation between this metric and the jumpstart suggests that this metric can be helpful when choosing among source tasks. $^{\uparrow}/^{\downarrow}$ means the higher/lower, the better.

domain, with the correlation coefficient of -0.79, shown in the last row of Table 4.4.

Takeaway: Next state distance has a moderate to strong negative correlation with jumpstart. Additionally, the cost of picking the best MDP based on this metric ranges between two to 6.3. Therefore, this metric can be helpful when choosing among source tasks. Additionally, Next state distance has an advantage over action distance: since it is a model-based metric, it does not require any training on the goal task. The results for this metric will be compared to the results of the other two distance metrics at the end of Section 4.3.1 to find the best distance metric for predicting the jumpstart on MDP^*.

Reward Distance

As explained in Section 3.4, reward distance is defined as the mean squared error between the immediate reward values of the state-action pairs in the source and target tasks. This metric has an advantage over action distance: it is a model-based metric, and therefore, it does not require any training on the goal task. In this section, we aim to evaluate the ability of this metric in predicting the resulting jumpstart from pre-training on each of $MDP_{1...16}$. To accomplish this goal, we look into the Pearson correlation between reward distance and jumpstart and the cost of picking the best MDP when using this

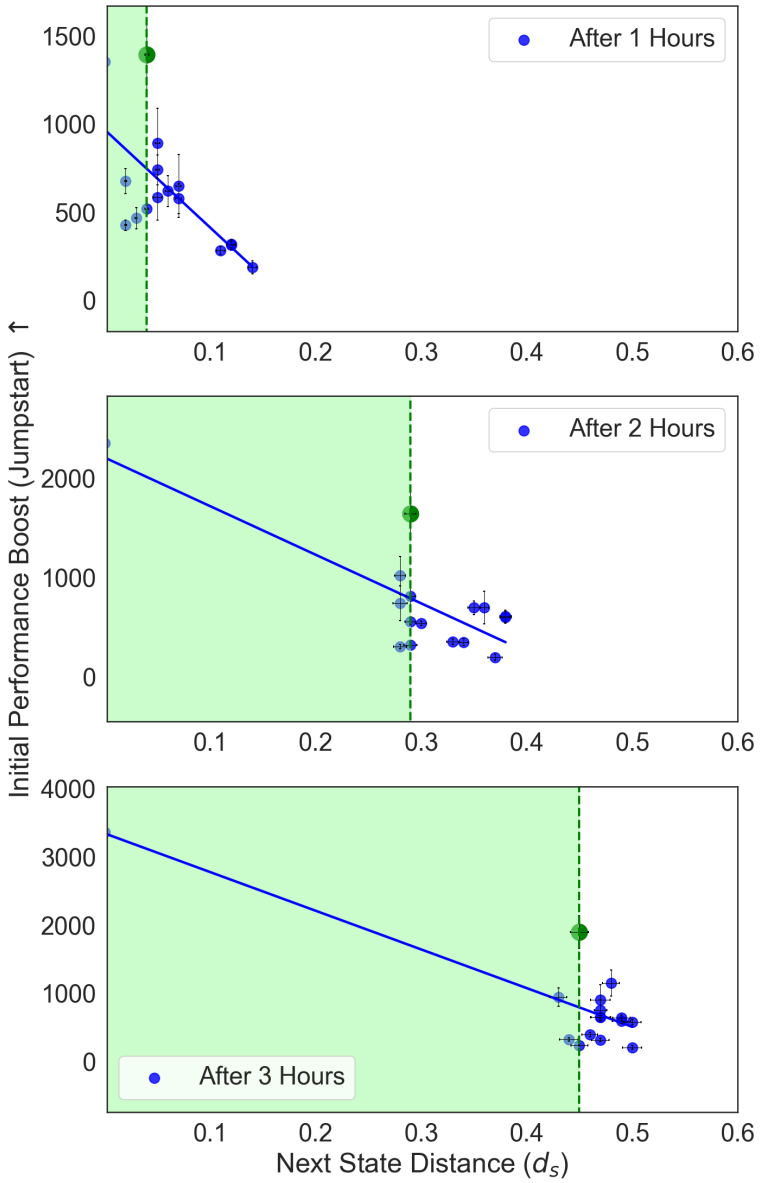

Figure 4.5: Hopper: Jumpstart on MDP^* versus next state distance

The green dot: the MDP resulting in the biggest jumpstart. The dots in the shaded area: the MDPs with smaller action distances than the best MDP. Based on the moderate to strong negative correlation (blue line) between next state distance and jumpstart, this metric can be helpful in choosing among source tasks.

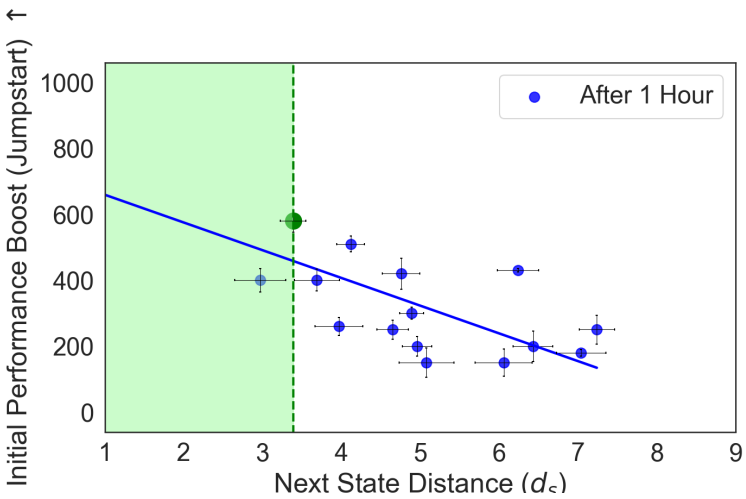

Figure 4.6: Pendulum: Jumpstart on MDP^* versus next state distance

The green dot: the MDP resulting in the biggest jumpstart. The dots in the shaded area: the MDPs with smaller action distances than the best MDP. Based on the moderate to strong negative correlation (blue line) between next state distance and jumpstart, this metric can be helpful in choosing among source tasks.

metric.

1) The cost of picking the best MDP in each trial: the cost of picking the best MDP in each trial that has been explained in Section 4.2.2 is mentioned in the first and the third rows of Table 4.5 for Hopper and Pendulum. In the worst case, the cost of picking the best MDP in each trial is 6.3, so we need to pick the top seven MDPs based on reward distance in order to include the MDP that would result in the biggest jumpstart. The probability of choosing the best MDP among seven picked MDPs is one in seven or 14.28%. If we were to pick a random MDP among our 16 MDPs, the chances of the MDP that we have picked being the best MDP would be 1 in 16 or 6.25%. This can be interpreted as saying that using reward distance is more than two times better than random selection among source tasks. In the Pendulum domain, this cost is 3.8, meaning we need to pick the top four MDPs to include the best MDP (regarding jumpstart).

2) The cost of picking the best MDP over 15 trials: The cost of picking the best MDP that has been explained in Section 4.2.2 is mentioned in the

second and the fourth rows of Table 4.5. This value can be interpreted as the number of MDPs in the green shaded area, including the MDPs on the green border in Figure 4.8 and subfigures of Figure 4.7. In the best case, the next state distance is able to place the best MDP among the top two MDPs. In the worst case, we need to pick the top six MDPs based on the next state distance in order to include the MDP that would result in the biggest jumpstart. The probability of choosing the best MDP among six picked MDPs is one in six or 16.6%. If we were to pick a random MDP among our 16 MDPs, the chances of the MDP that we have picked being the best MDP would be one in 16 or 6.25%. This can be interpreted as saying that using action distance is more than two times better than random selection among source tasks.

3) The Pearson correlation between jumpstart and reward distance: The blue line in Figure 4.8 and the sub-figures of Figure 4.7 represents the Pearson correlation between reward distance and jumpstart. Additionally, the fifth row in Table 4.5, shows the value of the Pearson correlation coefficient in the Hopper domain after one, two, and three hours of training time. This value ranges between -0.88 to -0.62, which indicates a moderate to strong negative correlation between the value of reward distance and the jumpstart on MDP^*[27]. This negative correlation exists in the Pendulum domain too, with the correlation coefficient of -0.85, shown in the last row of Table 4.5.

The results for this metric will be compared to the results of the other two distance metrics at the end of Section 4.3.1 to find the best distance metric for predicting the jumpstart on MDP^*.

Takeaway: Reward distance has a strong negative correlation with jumpstart. Additionally, the cost of picking the best MDP based on this metric ranges between two to 6.3. Therefore, this metric can be helpful when choosing among source tasks. Additionally, reward distance has an advantage over action distance: since it is a model-based metric, it does not require any training on the goal task. The results for this metric will be compared to the results of the other two distance metrics at the end of Section 4.3.1 to find the best distance metric for predicting the jumpstart on MDP^*.

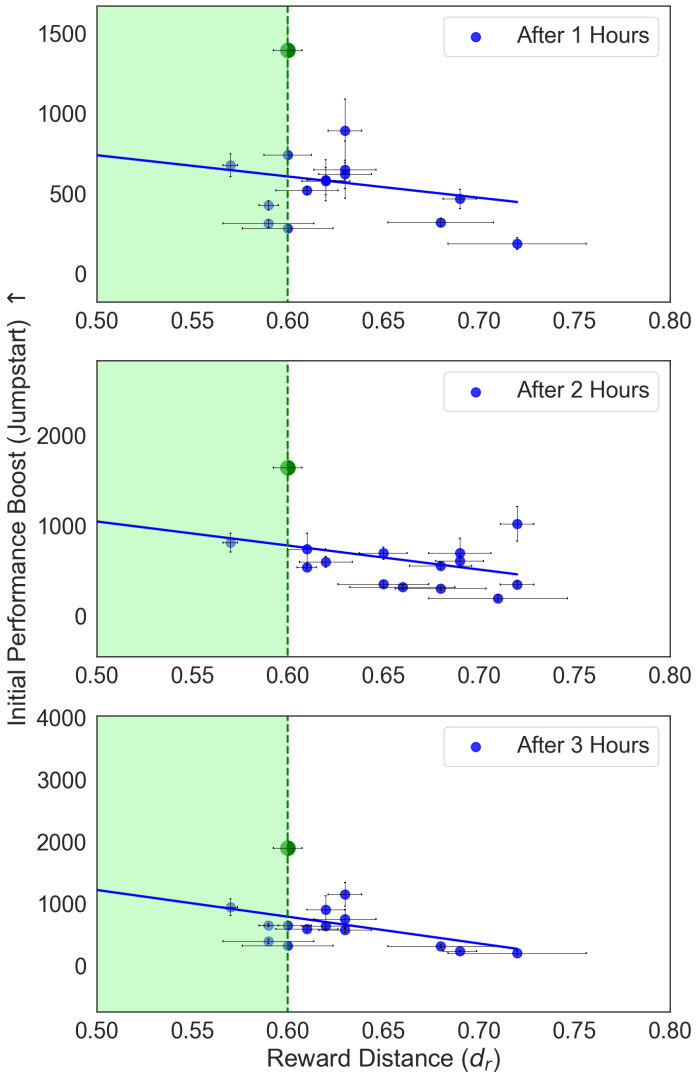

Figure 4.7: Hopper: Jumpstart on MDP^* versus reward distance

The green dot: the MDP resulting in the biggest jumpstart. The dots in the shaded area: the MDPs with smaller reward distances than the best MDP. Based on the moderate to strong negative correlation (blue line) between reward distance and jumpstart, this metric can be helpful in choosing among source tasks.

Table 4.5: Jumpstart Correlation with Reward Distance

Metric	Environment		Training Time(Hours)		
			1 h	2 h	3h
Cost of picking the best MDP ↓	Hopper	in each trial	4.6	4.6	6.3
		over 15 runs	5	2	6
	Pendulum	in each trial	3.8	-	-
		over 15 runs	3	-	-
Pearson Correlation Coefficient ↓	Hopper	over 15 runs	-0.62	-0.82	-0.88
	Pendulum	over 15 runs	-0.85	-	-

First four rows: Cost of picking the best MDP in each trial and over 15 trials. last two rows: the Pearson correlation between reward distance and the jumpstart on MDP^*. The negative correlation between this metric and the jumpstart suggest that this metric can be helpful when choosing among source tasks. $^{\uparrow}/_{\downarrow}$ means the higher/lower, the better.

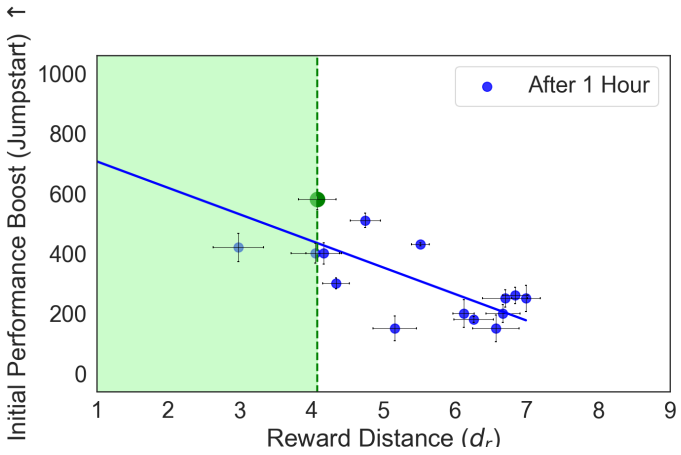

Figure 4.8: Pendulum: Jumpstart on MDP^* versus reward distance

The green dot: the MDP resulting in the biggest jumpstart. The dots in the shaded area: the MDPs with smaller reward distances than the best MDP. Based on the moderate to strong negative correlation (blue line) between action distance and jumpstart, this metric can be helpful in choosing among source tasks.

Table 4.6: Jumpstart Correlation with Distance Metrics

Metric		Action Distance	Next State Distance	Reward Distance
		Hopper		
Cost of picking the best MDP	in each trial	3.2	4.6	4.6
	over 15 run	1	5	5
Pearson Correlation	over 15 run	-0.76	-0.65	-0.62
		Pendulum		
Cost of picking the best MDP	in each trial	3.6	3.6	3.8
	over 15 run	1	2	3
Pearson Correlation	over 15 run	-0.92	-0.79	-0.85

The action distance has a lower cost of picking the best MDP and a bigger absolute value for the Pearson correlation coefficient (indicating a stronger negative correlation) compared to the other two metrics. Action distance is the best metric for picking a source task if our goal is to get a high jumpstart on the goal task.

Comparison of the three distance metrics for prediction jumpstart

As it can be seen in Table 4.6, action distance has the strongest correlation with jumpstart and the lowest cost of picking the best MDP in all cases. As a result, we conclude that action distance has the strongest ability in predicting the jumpstart on MDP^*. However, action distance is performance-based and requires training on the goal task. It is not always possible or preferable to train on MDP^*. Therefore, when choosing a distance metric for selecting among the source tasks, the cost of calculating the metric should be considered along with the metric's performance.

4.3.2 Asymptotic Performance correlation with Distance Metrics

Asymptotic performance, defined in Section 4.2.1, compares the final performances of learners in MDP^* with and without transfer. In this section, we evaluate the ability of each distance metric in predicting the asymptotic performance in MDP^* resulting from pre-training on each MDP. To evaluate the predictive ability of each distance metric, we calculate the Pearson correlation between that distance metric and asymptotic performance, and also the cost of picking the best MDP (the MDP that pre-training on it would result in the

highest asymptotic performance) based on that metric. We use the expression of "moderate correlation" for coefficient values between 0.5 to 0.7 or -0.7 to -0.5, And the expression of "strong correlation" for coefficients with bigger absolute values than 0.7 [27]. Distance metrics are complementary to similarity metrics, therefore in an effective distance metric, the smaller the distance metric, the higher the asymptotic performance should be.

The experiments are performed in the Hopper domain after one, two, and three hours of training. Our algorithm takes 3 hours to fully converge, which in the Hopper task at code level it equals 900000 timesteps. As a result, we divided the training time into three equal parts, each lasting for one hour, or 300000 timesteps. In order to gain more confidence in the observations, the experiments are repeated in the Pendulum domain. However, due to the simpler nature of the Pendulum task, training for more than one hour does not affect the experiments' results, and therefore they are only included for the one-hour case. In the following sections, we evaluate the predictive ability of action distance, next state distance, and reward distance consecutively.

Action Distance

As explained in Section 3.2, action distance is a performance-based metric defined as the distance of actions for corresponding states in two MDPs. In this section we aim to evaluate the ability of this metric in predicting the resulting asymptotic performance from pre-training on each of $MDP_{1...16}$. To accomplish this goal, we look into the Pearson correlation between action distance and the cost of picking the best MDP when using this metric.

1) The cost of picking the best MDP in each trial: the cost of picking the best MDP in each trial that has been explained in Section 4.2.2 is mentioned in the first and the third rows of Table 4.7 for Hopper and Pendulum. In the best case, action distance can place the best MDP among the top three MDPs. In the Pendulum domain, this cost is 1.4, meaning we need to pick the top two MDPs to include the best MDP (regarding asymptotic performance).

2) The cost of picking the best MDP over 15 trials: The cost of picking the best MDP that has been explained in Section 4.2.2 is mentioned in the second

and the fourth rows of Table 4.7. This value can be interpreted as the number of MDPs in the green shaded area, including the MDPs on the green border in Figure 4.10 and subfigures of Figure 4.9. In the best case, action distance can single out the best MDP in both Hopper and Pendulum domains. In the worst case, we need to pick the top seven MDPs based on action distance to include the MDP that would result in the biggest asymptotic performance. The probability of choosing the best MDP among seven picked MDPs is one in seven or 14.28%. If we were to pick a random MDP among our 16 MDPs, the chances of the MDP that we have picked being the best MDP would be one in 16 or 6.25%. This can be interpreted as saying that using action distance is more than two times better than random selection among source tasks.

3) The Pearson correlation between asymptotic performance and action distance: The blue line in Figure 4.10 and the sub-figures of Figure 4.9 represents the Pearson correlation between action distance and asymptotic performance. Additionally, the fifth row in Table 4.7 shows the value of the Pearson correlation coefficient in the Hopper domain after one, two, and three hours of training time. This value indicates a negative correlation between the value of action distance and the asymptotic performance on MDP^*. This negative correlation exists in the Pendulum domain too, with the correlation coefficient of -0.86, shown in the last row of Table 4.7.

Takeaway: Action distance has a negative correlation with asymptotic performance. Therefore, this metric can be helpful when choosing among source tasks. However, this metric is performance-based, and it requires training on the goal task, which might not be desirable or possible. Deciding to use this metric depends on the nature of the goal task along with other factors. The results for this metric will be compared to the results of the other two distance metrics at the end of Section 4.3.2 to find the best distance metric for predicting the asymptotic performance on MDP^*.

Next state Distance

As explained in Section 3.3, next state distance is a model-based metric defined as the distance of next states for corresponding states in two MDPs. In

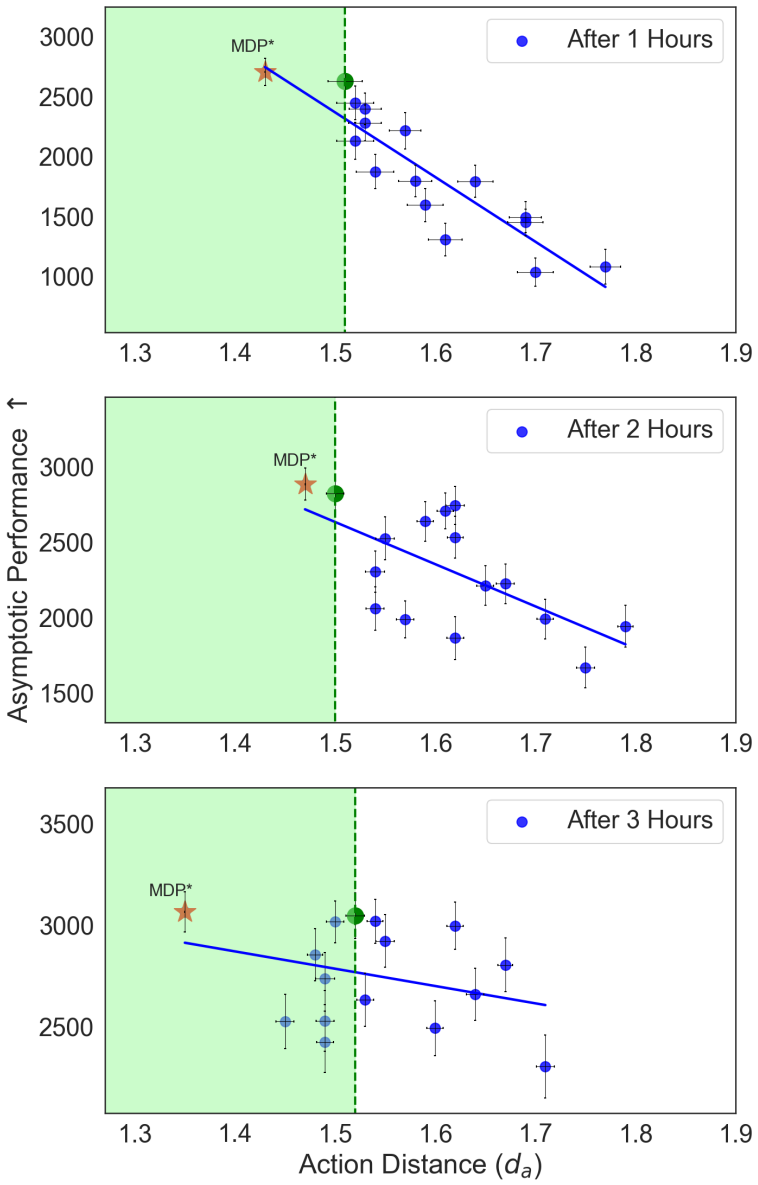

Figure 4.9: Hopper: Asymptotic performance on MDP^* versus action distance

The green dot: the MDP resulting in the biggest asymptotic performance. The dots in the shaded area: the MDPs with smaller action distances than the best MDP. Based on the negative correlation (blue line) between action distance and asymptotic performance, this metric can be helpful in picking source tasks.

Table 4.7: Asymptotic Performance Correlation with Action Distance

Metric	Environment		Training Time(Hours)		
			1 h	2 h	3 h
Cost of picking the best MDP ↓	Hopper	in each trial	3.2	5.9	8.5
		over 15 runs	1	1	7
	Pendulum	in each trial	1.4	-	-
		over 15 runs	1	-	-
Pearson Correlation Coefficient ↓	Hopper	over 15 runs	-0.9	-0.64	-0.31
	Pendulum	over 15 runs	-0.86	-	-

First four rows: Cost of picking the best MDP in each trial and over 15 trials. last two rows: the Pearson correlation between action distance and the asymptotic performance on MDP^*. The negative correlation and the low costs of picking the best MDP suggest that this metric can be helpful when choosing among source tasks. $^{\uparrow}/^{\downarrow}$ means the higher/lower, the better.

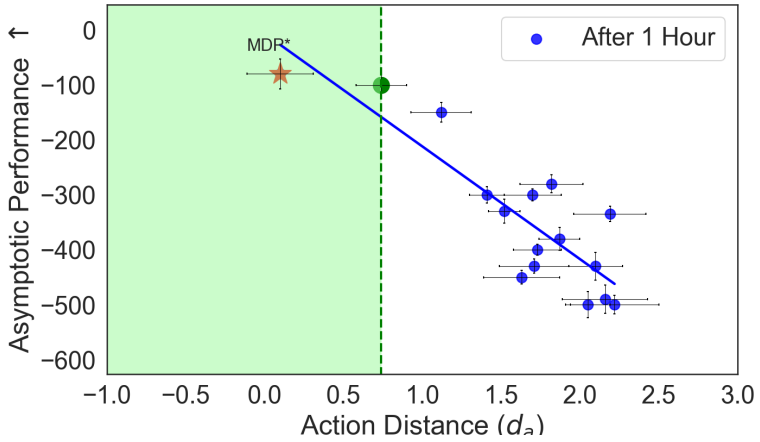

Figure 4.10: Pendulum: Asymptotic performance on MDP^* vs. action distance

The green dot: the MDP resulting in the biggest asymptotic performance. The dots in the shaded area: the MDPs with smaller action distances than the best MDP. Based on the strong negative correlation (blue line) between action distance and asymptotic performance, this metric can be helpful in picking source tasks.

this section, we aim to evaluate the ability of this metric in predicting the resulting asymptotic performance from pre-training on each of $MDP_{1...16}$. To accomplish this goal, we look into the Pearson correlation between the next state distance and the cost of picking the best MDP when using this metric.

1) The cost of picking the best MDP in each trial: the cost of picking the best MDP in each trial that has been explained in Section 4.2.2 is mentioned in the first and the third rows of Table 4.8 for Hopper and Pendulum. In the best case, the next state distance can place the best MDP among the top five MDPs. In the Pendulum domain, this cost is 2.3, meaning we need to pick the top three MDPs to include the best MDP (regarding asymptotic performance).

2) The cost of picking the best MDP over 15 trials: The cost of picking the best MDP that has been explained in Section 4.2.2 is mentioned in the second and the fourth rows of Table 4.8. This value can be interpreted as the number of MDPs in the green shaded area, including the MDPs on the green border in Figure 4.12 and subfigures of Figure 4.11. In the best case, the next state distance can place the best MDP in the five top MDPs in Hopper and in the top two MDPs in the Pendulum. The probability of choosing the best MDP among five picked MDPs is one in five or 20%. If we were to pick a random MDP among our 16 MDPs, the chances of the MDP that we have picked being the best MDP would be one in 16 or 6.25%. This can be interpreted as saying that using the next state distance is more than three times better than random selection among source tasks.

3) The Pearson correlation between asymptotic performance and the next state distance: The blue line in Figure 4.12 and the sub-figures of Figure 4.11 represents the Pearson correlation between the next state distance and asymptotic performance. Additionally, the fifth row in Table 4.8 shows the value of the Pearson correlation coefficient in the Hopper domain after one, two, and three hours of training time. This value indicates a moderate negative correlation between value of the next state distance and the asymptotic performance on MDP^*. This negative correlation exists in the Pendulum domain too, with the correlation coefficient of -0.7, shown in the last row of Table 4.8.

Takeaway: Next state distance has a negative correlation with asymptotic

Table 4.8: Asymptotic Performance Correlation with Next State Distance

Metric	Environment		Training Time(Hours)		
			1 h	2 h	3 h
Cost of picking the best MDP $^{\downarrow}$	Hopper	in each trial	5	6.4	8.6
		over 15 runs	5	7	10
	Pendulum	in each trial	2.3	-	-
		over 15 runs	2	-	-
Pearson Correlation Coefficient $^{\downarrow}$	Hopper	over 15 runs	-0.67	-0.51	-0.42
	Pendulum	over 15 runs	-0.7	-	-

First four rows: Cost of picking the best MDP in each trial and over 15 trials. last two rows: the Pearson correlation between next state distance and the asymptotic performance on MDP^*. The moderate negative correlation and the low costs of picking the best MDP suggest that this metric can be helpful when choosing among source tasks. $^{\uparrow}/^{\downarrow}$ means the higher/lower, the better.

performance. Therefore, this metric can be helpful when choosing among source tasks. Additionally, Next state distance has an advantage over action distance: since it is a model-based metric, it does not require any training on the goal task. Deciding to use this metric depends on the nature of the goal task along with other factors. The results for this metric will be compared to the results of the other two distance metrics at the end of Section 4.3.2 to find the best distance metric for predicting the asymptotic performance on MDP^*.

Reward distance

As explained in Section 3.4, reward distance is defined as the mean squared error between the immediate reward values of the state-action pairs in the source and target tasks. This metric has an advantage over action distance: it is a model-based metric, and therefore, it does not require any training on the goal task. In this section, we aim to evaluate the ability of this metric in predicting the resulting asymptotic performance from pre-training on each of $MDP_{1...16}$. To accomplish this goal, we look into the Pearson correlation between reward distance and asymptotic performance and the cost of picking the best MDP when using this metric.

1) The cost of picking the best MDP in each trial: the cost of picking the best MDP in each trial that has been explained in Section 4.2.2 is mentioned

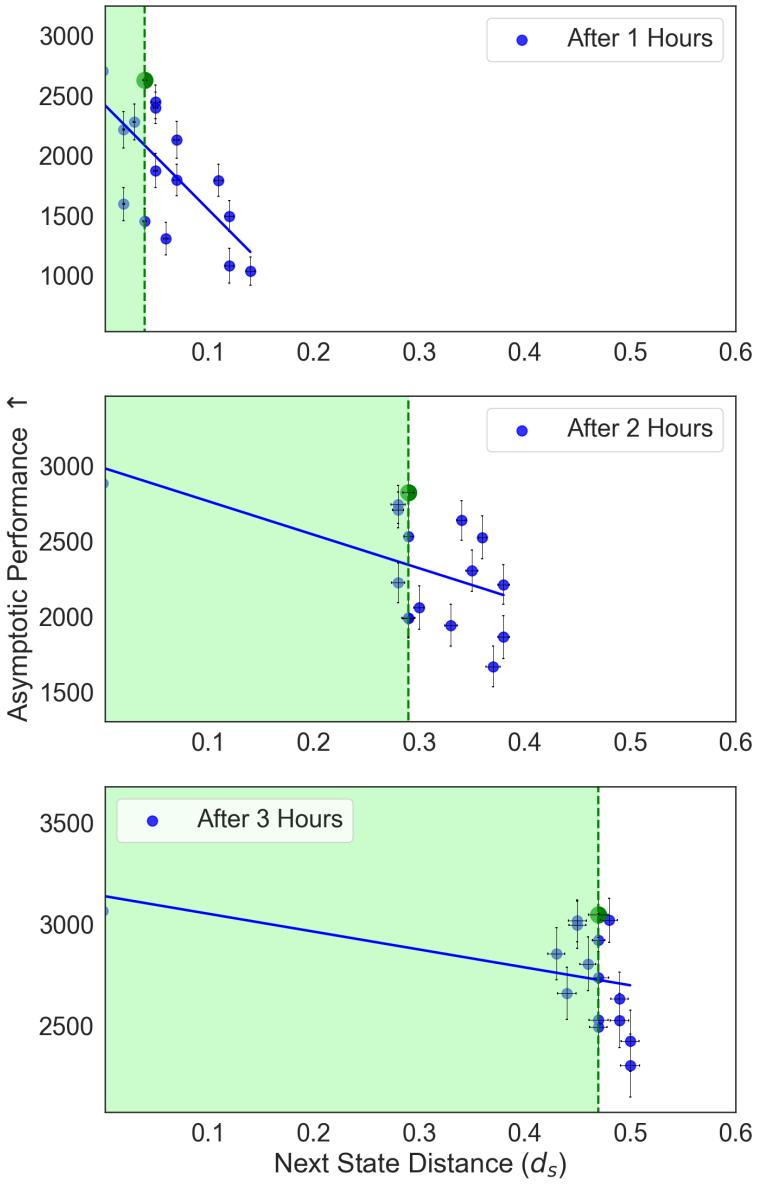

Figure 4.11: Hopper: Asymptotic performance vs. next state distance

The green dot: the MDP resulting in the biggest asymptotic performance. The dots in the shaded area: the MDPs with smaller next state distances than the best MDP. Based on the negative correlation (blue line) between next state distance and asymptotic performance, this metric can be helpful in picking source tasks.

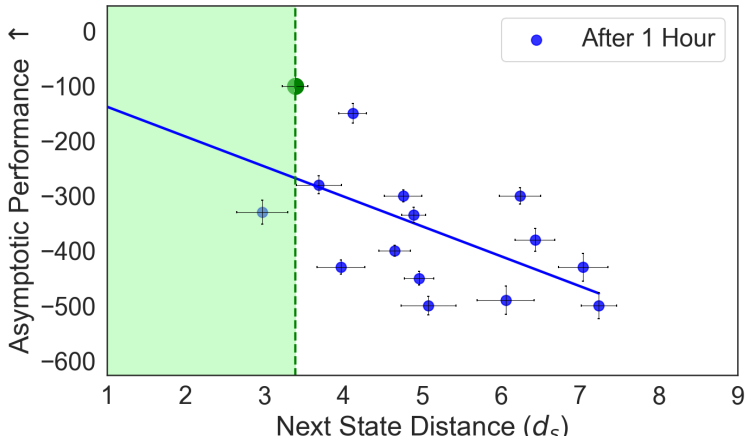

Figure 4.12: Pendulum: Asymptotic performance on MDP^* vs. next state distance

The green dot: the MDP resulting in the biggest asymptotic performance. The dots in the shaded area: the MDPs with smaller next state distances than the best MDP. Based on the strong negative correlation (blue line) between next state distance and asymptotic performance, this metric can be helpful in picking source tasks.

in the first and the third rows of Table 4.9 for Hopper and Pendulum. In the Pendulum domain, this cost is 4.1, meaning we need to pick the top four MDPs to include the best MDP (regarding asymptotic performance).

2) The cost of picking the best MDP over 15 trials: The cost of picking the best MDP that has been explained in Section 4.2.2 is mentioned in the second and the fourth rows of Table 4.9. This value can be interpreted as the number of MDPs in the green shaded area, including the MDPs on the green border in Figure 4.14 and subfigures of Figure 4.13. In the best case, the next state distance can place the best MDP among the top two MDPs. The probability of choosing the best MDP among two picked MDPs is one in two or 50%. If we were to pick a random MDP among our 16 MDPs, the chances of the MDP that we have picked being the best MDP would be one in 16 or 6.25%.

3) The Pearson correlation between asymptotic performance and reward distance: The blue line in Figure 4.14 and the sub-figures of Figure 4.13 represents the Pearson correlation between reward distance and asymptotic perfor-

Table 4.9: Asymptotic Performance Correlation with Reward Distance

Metric	Environment		Training Time(Hours)		
			1 h	2 h	3 h
Cost of picking the best MDP $^{\downarrow}$	Hopper	in each trial	6	6.4	9.8
		over 15 runs	6	2	9
	Pendulum	in each trial	4.1	-	-
		over 15 runs	3	-	-
Pearson Correlation Coefficient $^{\downarrow}$	Hopper	over 15 runs	-0.44	-0.41	-0.39
	Pendulum	over 15 runs	-0.7	-	-

First four rows: Cost of picking the best MDP in each trial and over 15 trials. last two rows: the Pearson correlation between reward distance and the asymptotic performance on MDP^*. The moderate to strong negative correlation and the low costs of picking the best MDP suggest that this metric can be helpful when choosing among source tasks. $^{\uparrow}/^{\downarrow}$ means the higher/lower, the better.

mance. Additionally, the fifth row in Table 4.9, shows the value of the Pearson correlation coefficient in the Hopper domain after one, two, and three hours of training time. This value indicates a negative correlation between the value of reward distance and the asymptotic performance on MDP^*. This negative correlation exists in the Pendulum domain too, with the correlation coefficient of -0.7, shown in the last row of Table 4.9.

The results for this metric will be compared to the results of the other two distance metrics at the end of Section 4.3.2 to find the best distance metric for predicting the asymptotic performance on MDP^*.

Takeaway: Reward distance has a negative correlation with asymptotic performance. Additionally, reward distance has an advantage over action distance: since it is a model-based metric, it does not require any training on the goal task. Therefore, this metric can be helpful when choosing among source tasks. The results for this metric will be compared to the results of the other two distance metrics at the end of Section 4.3.2 to find the best distance metric for predicting the asymptotic performane on MDP^*.

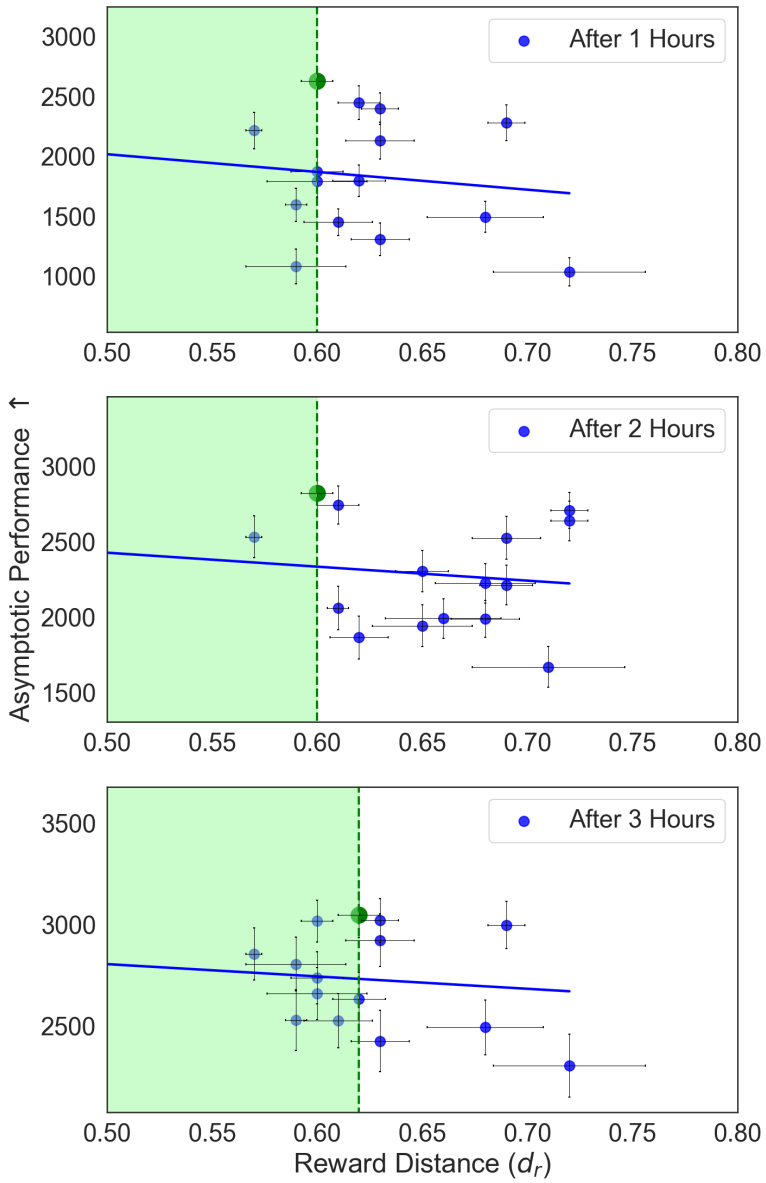

Figure 4.13: Hopper: Asymptotic performance on MDP^* vs. reward distance

The green dot: the MDP resulting in the biggest asymptotic performance. The dots in the shaded area: the MDPs with smaller reward distances than the best MDP. Based on the negative correlation (blue line) between reward distance and asymptotic performance, this metric can be helpful in picking source tasks.

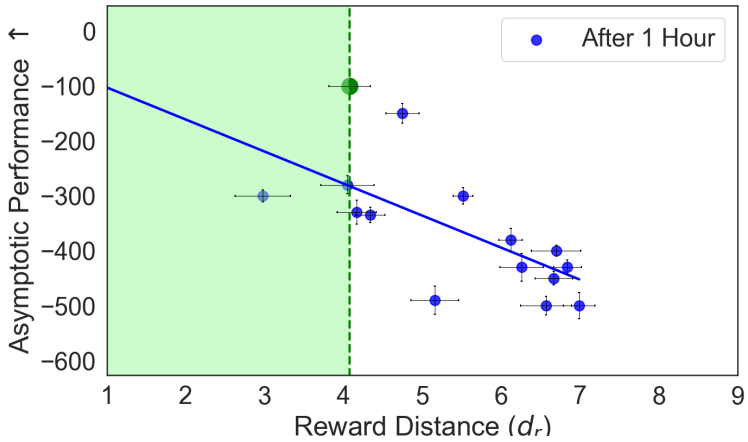

Figure 4.14: Pendulum: Asymptotic performance on MDP^* vs. reward distance

The green dot: the MDP resulting in the biggest asymptotic performance. The dots in the shaded area: the MDPs with smaller reward distances than the best MDP. Based on the negative correlation (blue line) between next reward and asymptotic performance, this metric can be helpful in picking source tasks.

Comparison of the three distance metrics for prediction asymptotic performance

As it can be seen in Table 4.10, action distance has the strongest correlation with asymptotic performance and the lowest cost of picking the best MDP in all cases. As a result, we conclude that action distance has the strongest ability in predicting the asymptotic performance on MDP^*. However, action distance is performance-based and requires training on the goal task. It is not always possible or preferable to train on MDP^*. Among the two model-based metrics, our metric , next state distance, has a lower cost of picking the best MDP and a stronger Pearson correlation with asymptotic performance. Therefore, we conclude that the next state distance has a better predictive ability for asymptotic performance.

Table 4.10: Asymptotic performance Correlation with Distance Metrics

Metric		Action Distance	Next State Distance	Reward Distance
		Hopper		
Cost of picking the best MDP	in each trial	3.2	5	6
	over 15 run	1	5	6
Pearson Correlation	over 15 run	-0.9	-0.67	-0.44
		Pendulum		
Cost of picking the best MDP	in each trial	1.4	2.3	4.1
	over 15 run	1	2	3
Pearson Correlation	over 15 run	-0.86	-0.7	-0.7

The action distance has a lower cost of picking the best MDP and a bigger absolute value for the Pearson correlation coefficient (indicating a stronger negative correlation) compared to the other two metrics. Action distance is the best metric for picking a source task if our goal is to get a high asymptotic performance on the goal task.

4.3.3 Time to threshold correlation with Distance Metrics

Time to threshold, defined in Section 4.2.1, measures the time that it takes to reach and stay in a certain performance threshold with and without transfer. In this section, we evaluate the ability of each distance metric in predicting the time to threshold in MDP^* resulting from pre-training on each MDP. To evaluate the predictive ability of each distance metric, we calculate the Pearson correlation between that distance metric and time to threshold, and the cost of picking the best MDP (the MDP that pre-training on it would result in the lowest time to threshold) based on that metric. We use the expression of "moderate correlation" for coefficient values between 0.5 to 0.7 or -0.7 to -0.5, And the expression of "strong correlation" for coefficients with bigger absolute values than 0.7 [27]. Distance metrics are complementary to similarity metrics, therefore in an effective distance metric, the smaller the distance metric, the lower the time to threshold should be.

The experiments are performed in the Hopper domain after one, two, and three hours of training. Our algorithm takes 3 hours to fully converge, which in the Hopper task at code level it equals 900000 timesteps. As a result, we divided the training time into three equal parts, each lasting for one hour, or 300000 timesteps. In order to gain more confidence in the observations,

the experiments are repeated in the Pendulum domain. However, due to the simpler nature of the Pendulum task, training for more than one hour does not affect the experiments' results, and therefore they are only included for the one-hour case. In the following sections, we evaluate the predictive ability of action distance, next state distance, and reward distance consecutively.

Action distance

As explained in Section 3.2, action distance is a performance-based metric defined as the distance of actions for corresponding states in two MDPs. In this section, we aim to evaluate the ability of this metric in predicting the resulting time to threshold from pre-training on each of $MDP_{1...16}$. To accomplish this goal, we look into the Pearson correlation between action distance and time to threshold, and the cost of picking the best MDP when using this metric.

1) The cost of picking the best MDP in each trial: the cost of picking the best MDP in each trial that has been explained in Section 4.2.2 is mentioned in the first and the third rows of Table 4.11 for Hopper and Pendulum. In the best case, action distance can place the best MDP among the top five MDPs. In the Pendulum domain, this cost is 2.2, meaning we need to pick the top two MDPs to include the best MDP (regarding time to threshold).

2) The cost of picking the best MDP over 15 trials: The cost of picking the best MDP that has been explained in Section 4.2.2 is mentioned in the second and the fourth rows of Table 4.11. This value can be interpreted as the number of MDPs in the green shaded area, including the MDPs on the green border in Figure 4.16 and subfigures of Figure 4.15. In the best case, action distance can single out the best MDP in Hopper.

3) The Pearson correlation between time to threshold and action distance: The blue line in Figure 4.16 and the sub-figures of Figure 4.15 represents the Pearson correlation between action distance and time to threshold. Additionally, the fifth row in Table 4.11 shows the value of the Pearson correlation coefficient in the Hopper domain after one, two, and three hours of training time. This value indicates a positive correlation between the value of action distance and the time to threshold on MDP^*. This positive correlation exists

Table 4.11: Time to threshold Correlation with Action Distance

Metric	Environment		Training Time(Hours)		
			1 h	2 h	3 h
Cost of picking the best MDP ↓	Hopper	in each trial	5.3	5.3	9.5
		over 15 runs	1	1	5
	Pendulum	in each trial	2.2	-	-
		over 15 runs	2	-	-
Pearson Correlation Coefficient ↑	Hopper	over 15 runs	0.73	0.35	0.54
	Pendulum	over 15 runs	-	-	0.78

- First four rows: Cost of picking the best MDP in each trial and over 15 trials. last two rows: the Pearson correlation between action distance and the time to threshold on MDP^*. The positive correlation and the low costs of picking the best MDP suggest that this metric can be helpful when choosing among source tasks. $^{\uparrow}/^{\downarrow}$ means the higher/lower, the better.

in the Pendulum domain too, with the correlation coefficient of -0.78, shown in the last row of Table 4.11.

Takeaway: Action distance has a positive correlation with time to threshold. Therefore, this metric can be helpful when choosing among source tasks. However, this metric is performance-based, and it requires training on the goal task, which might not be desirable or possible. Deciding to use this metric depends on the nature of the goal task along with other factors. The results for this metric will be compared to the results of the other two distance metrics at the end of Section 4.3.3 to find the best distance metric for predicting the time to threshold on MDP^*.

Next state distance

As explained in Section 3.3, next state distance is a model-based metric defined as the distance of next states for corresponding state-action pairs in two MDPs. In this section, we aim to evaluate the ability of this metric in predicting the resulting time to threshold from pre-training on each of $MDP_{1...16}$. To accomplish this goal, we look into the Pearson correlation between next state distance and time to threshold, and the cost of picking the best MDP when using this metric.

1) The cost of picking the best MDP in each trial: the cost of picking the

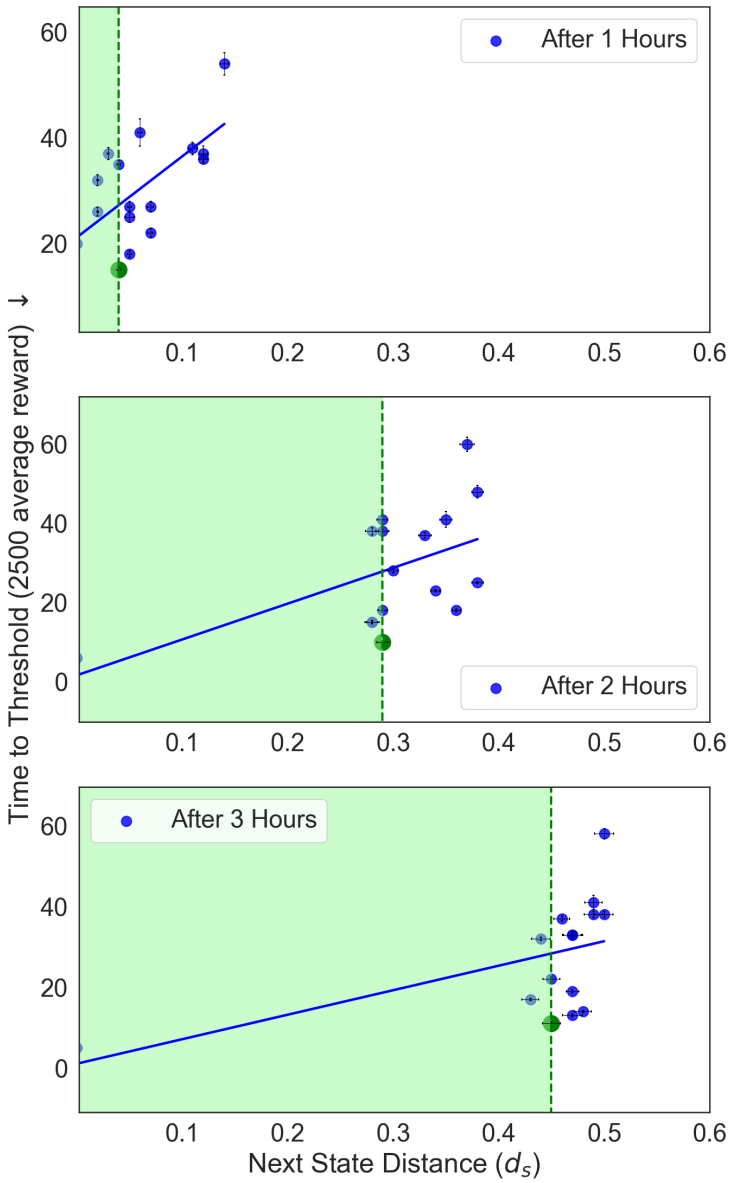

Figure 4.15: Hopper: Time to threshold on MDP^* versus action distance

The green dot: the MDP resulting in the lowest time to threshold. The dots in the shaded area: the MDPs with smaller action distances than the best MDP. Based on the positive correlation (blue line) between action distance and time to threshold, this metric can be helpful in picking source tasks.

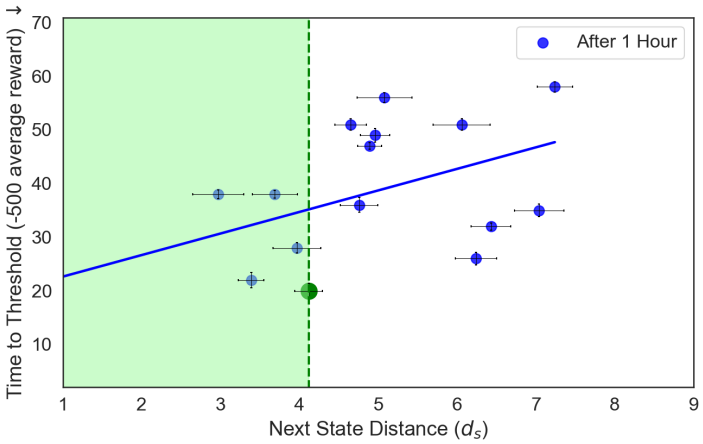

Figure 4.16: Pendulum: Time to threshold on MDP^* vs. action distance

The green dot: the MDP resulting in the lowest time to threshold. The dots in the shaded area: the MDPs with smaller action distances than the best MDP. Based on the strong positive correlation (blue line) between action distance and time to threshold, this metric can be helpful in picking source tasks.

best MDP in each trial that has been explained in Section 4.2.2 is mentioned in the first and the third rows of Table 4.12 for Hopper and Pendulum. In the best case, next state distance can place the best MDP among the top five MDPs.

2) The cost of picking the best MDP over 15 trials: The cost of picking the best MDP that has been explained in Section 4.2.2 is mentioned in the second and the fourth rows of Table 4.12. This value can be interpreted as the number of MDPs in the green shaded area, including the MDPs on the green border in Figure 4.18 and subfigures of Figure 4.17.

3) The Pearson correlation between time to threshold and action distance: The blue line in Figure 4.18 and the sub-figures of Figure 4.17 represents the Pearson correlation between action distance and time to threshold. Additionally, the fifth row in Table 4.12 shows the value of the Pearson correlation coefficient in the Hopper domain after one, two, and three hours of training time. This value indicates a positive correlation between the value of action distance and the time to threshold on MDP^*. This positive correlation exists in the Pendulum domain too, with the correlation coefficient of -0.6, shown

Table 4.12: Time to threshold Correlation with Next State Distance

Metric	Environment		Training Time(Hours)		
			1 h	2 h	3 h
Cost of picking the best MDP ↓	Hopper	in each trial	6.6	4.6	9.5
		over 15 runs	5	7	4
	Pendulum	in each trial	4.8	-	-
		over 15 runs	5	-	-
Pearson Correlation Coefficient ↑	Hopper	over 15 runs	0.62	0.53	0.51
	Pendulum	over 15 runs	0.6	-	-

First four rows: Cost of picking the best MDP in each trial and over 15 trials. last two rows: the Pearson correlation between next state distance and the time to threshold on MDP^*. The positive correlation and the low costs of picking the best MDP suggest that this metric can be helpful when choosing among source tasks. $^{\uparrow}/^{\downarrow}$ means the higher/lower, the better.

in the last row of Table 4.12.

Takeaway: Next state distance has a positive correlation with time to threshold. Therefore, this metric can be helpful when choosing among source tasks. Deciding to use this metric depends on the nature of the goal task along with other factors. The results for this metric will be compared to the results of the other two distance metrics at the end of Section 4.3.3 to find the best distance metric for predicting the time to threshold on MDP^*.

Reward distance

As explained in Section 3.4, reward distance is a model-based metric defined as the distance of rewards for corresponding state-action pairs in two MDPs. In this section, we aim to evaluate the ability of this metric in predicting the resulting time to threshold from pre-training on each of $MDP_{1...16}$. To accomplish this goal, we look into the Pearson correlation between reward distance and time to threshold, and the cost of picking the best MDP when using this metric.

1) The cost of picking the best MDP in each trial: the cost of picking the best MDP in each trial that has been explained in Section 4.2.2 is mentioned in the first and the third rows of Table 4.13 for Hopper and Pendulum. In the best case, next state distance can place the best MDP among the top five

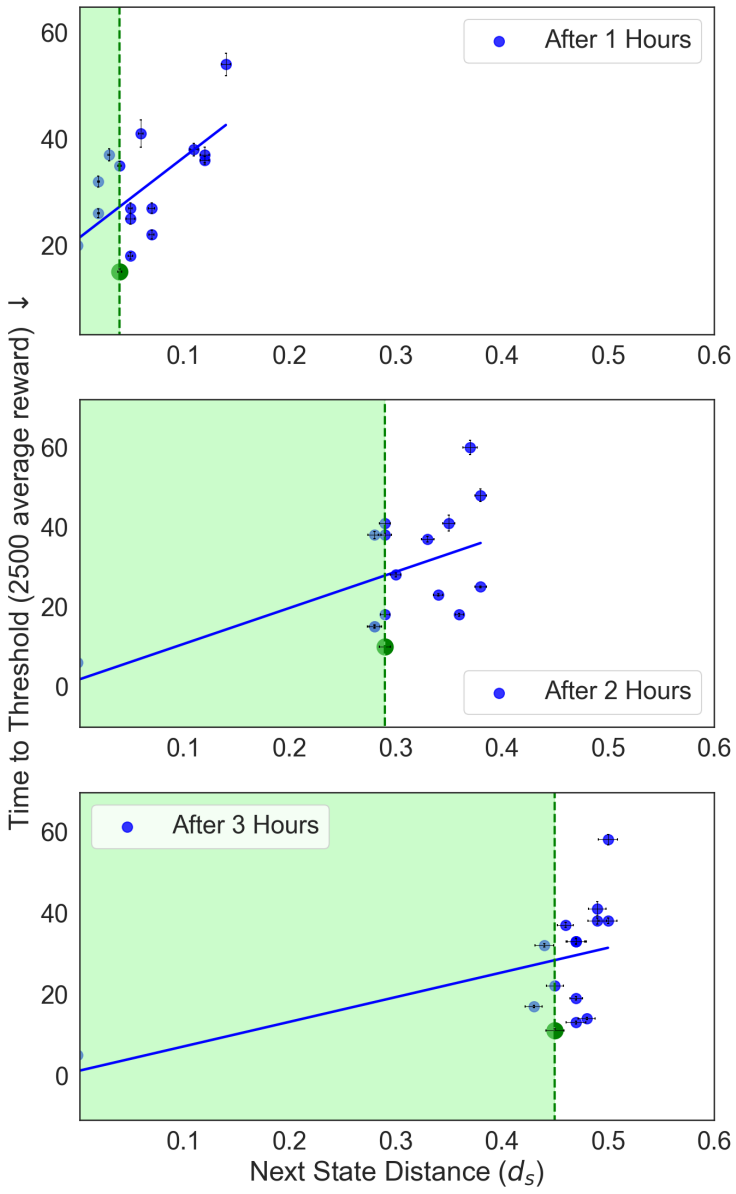

Figure 4.17: Hopper: Time to threshold on MDP^* versus next state distance

The green dot: the MDP resulting in the lowest time to threshold. The dots in the shaded area: the MDPs with smaller next state distances than the best MDP. Based on the positive correlation (blue line) between next state distance and time to threshold, this metric can be helpful in picking source tasks.

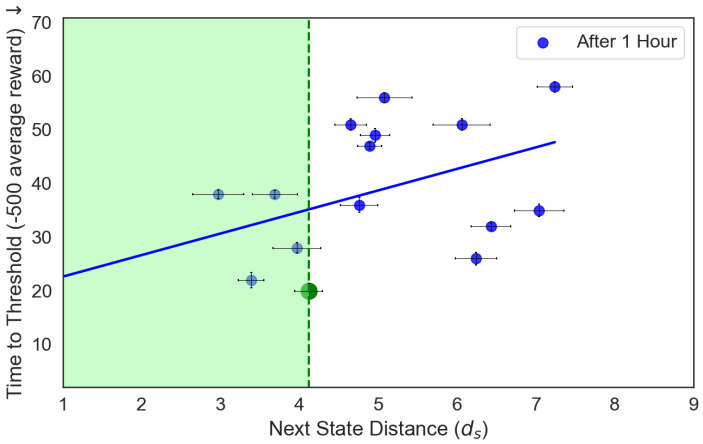

Figure 4.18: Pendulum: Time to threshold on MDP^* vs. next state distance

The green dot: the MDP resulting in the lowest time to threshold. The dots in the shaded area: the MDPs with smaller next state distances than the best MDP. Based on the positive correlation (blue line) between next state distance and time to threshold, this metric can be helpful in picking source tasks.

MDPs.

2) The cost of picking the best MDP over 15 trials: The cost of picking the best MDP that has been explained in Section 4.2.2 is mentioned in the second and the fourth rows of Table 4.13. This value can be interpreted as the number of MDPs in the green shaded area, including the MDPs on the green border in Figure 4.20 and subfigures of Figure 4.19.

3) The Pearson correlation between time to threshold and next state distance: The blue line in Figure 4.20 and the sub-figures of Figure 4.19 represents the Pearson correlation between action distance and time to threshold. Additionally, the fifth row in Table 4.13 shows the value of the Pearson correlation coefficient in the Hopper domain after one, two, and three hours of training time. This value indicates a positive correlation between the value of action distance and the time to threshold on MDP^*. This positive correlation exists in the Pendulum domain too, with the correlation coefficient of 0.6, shown in the last row of Table 4.13.

Takeaway: Reward distance has a positive correlation with time to thresh-

Table 4.13: Time to threshold Correlation with Reward Distance

Metric	Environment		Training Time(Hours)		
			1 h	2 h	3 h
Cost of picking the best MDP ↓	Hopper	in each trial	6.6	4.6	6.3
		over 15 runs	6	2	6
	Pendulum	in each trial	5.4	-	-
		over 15 runs	5	-	-
Pearson Correlation Coefficient ↑	Hopper	over 15 runs	0.4	0.44	0.5
	Pendulum	over 15 runs	0.56	-	-

First four rows: Cost of picking the best MDP in each trial and over 15 trials. Last two rows: the Pearson correlation between reward distance and the time to threshold on MDP^*. The positive correlation and the low costs of picking the best MDP suggest that this metric can be helpful when choosing among source tasks. $^{\uparrow}/^{\downarrow}$ means the higher/lower, the better.

old. Therefore, this metric can be helpful when choosing among source tasks. Deciding to use this metric depends on the nature of the goal task along with other factors. The results for this metric will be compared to the results of the other two distance metrics at the end of Section 4.3.3 to find the best distance metric for predicting the time to threshold on MDP^*.

Comparison of the three distance metrics for prediction time to threshold

As it can be seen in Table 4.14, action distance has the strongest correlation with time to threshold and the lowest cost of picking the best MDP in all cases. As a result, we conclude that action distance has the strongest ability in predicting the time to threshold on MDP^*. However, action distance is performance-based and requires training on the goal task. It is not always possible or preferable to train on MDP^*. Among the two model-based metrics, our metric , next state distance, has a lower cost of picking the best MDP and a stronger Pearson correlation with time to threshold. Therefore, we conclude that the next state distance has a better predictive ability for time to threshold.

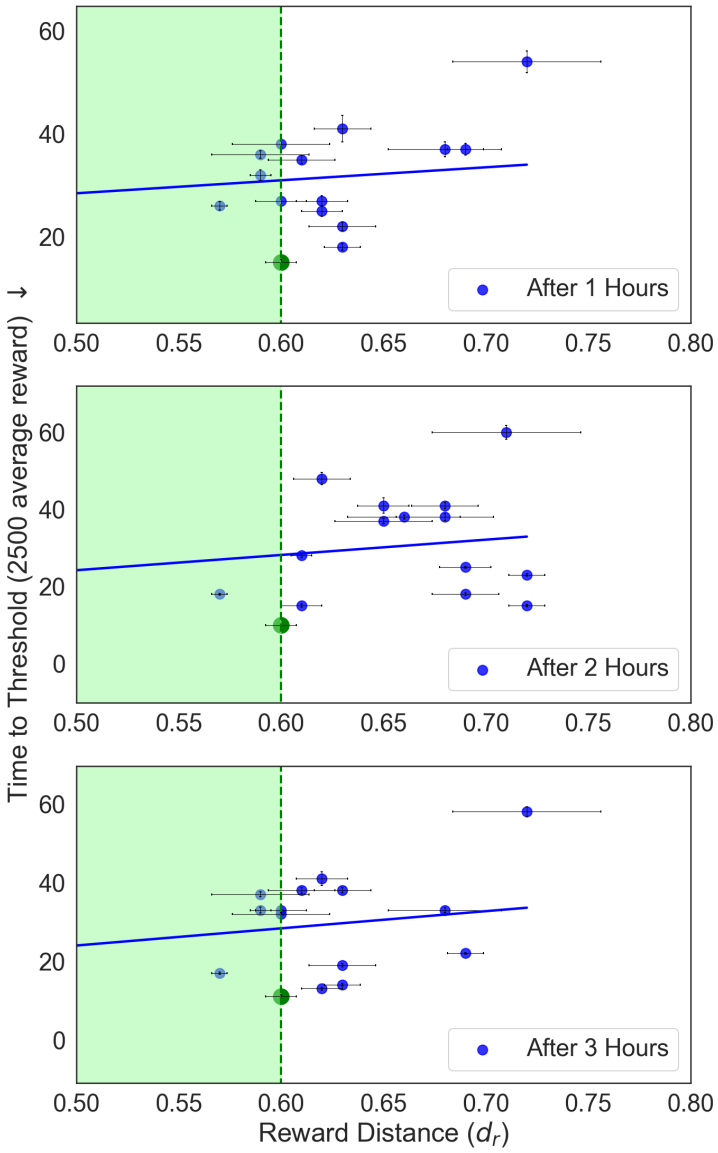

Figure 4.19: Hopper: Time to threshold on MDP^* versus reward distance

The green dot: the MDP resulting in the lowest time to threshold. The dots in the shaded area: the MDPs with smaller reward distances than the best MDP. Based on the positive correlation (blue line) between reward distance and time to threshold, this metric can be helpful in picking source tasks.

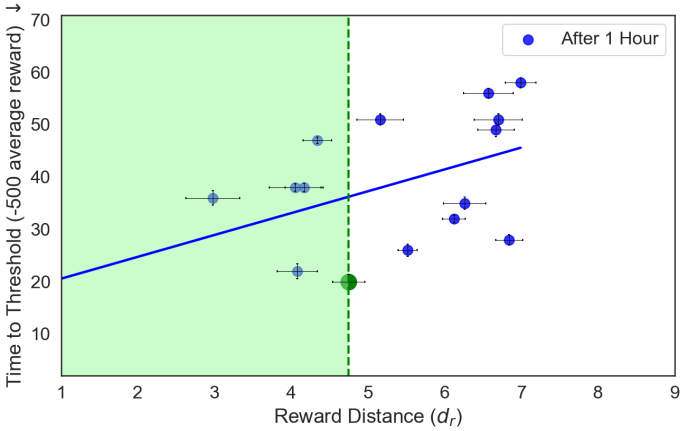

Figure 4.20: Pendulum: Time to threshold on MDP^* vs. reward distance

Table 4.14: Time to threshold Correlation with Distance Metrics

Metric		Action Distance	Next State Distance	Reward Distance
		Hopper		
Cost of picking the best MDP	in each trial	5.3	6.6	6.6
	over 15 run	1	5	6
Pearson Correlation	over 15 run	0.73	0.62	0.4
		Pendulum		
Cost of picking the best MDP	in each trial	2.2	4.8	5.4
	over 15 run	2	5	5
Pearson Correlation	over 15 run	0.78	0.6	0.56

The action distance has a lower cost of picking the best MDP and a bigger Pearson correlation coefficient value (indicating a stronger positive correlation) compared to the other two metrics. Action distance is the best metric for picking a source task if our goal is to get a low time to threshold on the goal task. Next state distance is better than reward distance for the same reasons.

Chapter 5

Conclusion and Future Work

In this last chapter, first, we summarize the results of our experiments, and then we pose several research questions that can act as guidelines for future work.

5.1 Conclusion

In this work, we looked into three similarity metrics and their predictive ability for finding the best MDP for transfer learning. Action distance was a performance-based distance metric. Although it proved to be the most predictive metric for jumpstart, final performance, and time to threshold, it has a big disadvantage because it requires training on the goal MDP. This requirement can defy the sole purpose of transfer learning which is no training or limited training on MDP^*. However, action distance still can prove beneficial since by training for only a short period of time on MDP^*, for example, 1 hour, it can single out the best MDP for transfer learning. The next state distance was based on the similarity of the next states when taking the same action on the same states. For calculating this metric, since the agent does not have to choose an action, having a policy is not a requirement; therefore, no training on MDP^* is needed, which is a huge advantage in cases when we do not want to train on the goal MDP. Even though the next state distance cannot single out the best MDP, in all the cases of this work, it has proven effective to eliminate the worst MDPs and place the best MDP in the top 5 or 6 MDPs. Therefore when choosing an MDP for transfer learning based on

this metric, the chance of selecting the best MDP among 16 MDPs is around 16%, which is more than twice the chance of choosing that MDP randomly. Reward distance was based on how the similarity of immediate rewards for corresponding state-action pairs. For calculating this metric also no training on MDP^* is needed, which is a huge advantage in cases when we do not want to train on the goal MDP.

5.2 Future Work

In this work, the number of source tasks was limited to 16. Testing on more MDP settings can result in a higher level of confidence in the results and probabilities presented in this work. Furthermore, all 16 different MDP settings were simulation environments. Transfer learning is commonly used for training an agent in simulation for performing a task in the real world. Given the unlimited complications of real-world tasks and the possibly huge similarity gap between the source tasks and the target tasks in those cases, these metrics might not prove effective. Additionally, even when transferring between simulation tasks, the similarity gap might be huge, or the task might be too complicated. Therefore more experiments in other domains and with more source tasks is needed to confirm the results of this work. Another possible approach to expand on this work is to try to understand the reasons behind these observations more deeply. In order to find more effective similarity metrics or make sure that certain similarity metrics can be helpful in unprecedented domains, having a deep understanding of these metrics and the results they show is essential.

References

[1] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[2] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey.," *Journal of Machine Learning Research*, vol. 10, no. 7, 2009.

[3] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[4] J. Hua, L. Zeng, G. Li, and Z. Ju, "Learning for a robot: Deep reinforcement learning, imitation learning, transfer learning," *Sensors*, vol. 21, no. 4, p. 1278, 2021.

[5] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[6] A. Narayan and T. Y. Leong, "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*, 2019, pp. 2132–2134.

[7] G. Kuhlmann and P. Stone, "Graph-based domain mapping for transfer learning in general games," in *European Conference on Machine Learning*, Springer, 2007, pp. 188–200.

[8] Y. Liu and P. Stone, "Value-function-based transfer for reinforcement learning using structure mapping," in *AAAI*, 2006, pp. 415–420.

[9] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[10] M. Köppen, "The curse of dimensionality," in *5th Online World Conference on Soft Computing in Industrial Applications (WSC5)*, vol. 1, 2000, pp. 4–8.

[11] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 2020, pp. 737–744.

[12] M. E. Taylor, P. Stone, and Y. Liu, "Transfer learning via inter-task mappings for temporal difference learning.," *Journal of Machine Learning Research*, vol. 8, no. 9, 2007.

[13] Á. Visús, J. Garcıa, and F. Fernández, "A taxonomy of similarity metrics for markov decision processes," *arXiv preprint arXiv:2103.04706*, 2021.

[14] S. Narvekar, J. Sinapov, and P. Stone, "Autonomous task sequencing for customized curriculum design in reinforcement learning.," in *IJCAI*, 2017, pp. 2536–2542.

[15] M. Mahmud, M. Hawasly, B. Rosman, and S. Ramamoorthy, "Clustering markov decision processes for continual transfer," *arXiv preprint arXiv:1311.3959*, 2013.

[16] J. L. Carroll and K. Seppi, "Task similarity measures for transfer in reinforcement learning task libraries," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, IEEE, vol. 2, 2005, pp. 803–808.

[17] H. Bou Ammar, E. Eaton, M. E. Taylor, *et al.*, "An automated measure of mdp similarity for transfer in reinforcement learning," in *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[18] A. Salehi, A. Coninx, and S. Doncieux, "Few-shot quality-diversity optimisation," *arXiv preprint arXiv:2109.06826*, 2021.

[19] M. E. Taylor, G. Kuhlmann, and P. Stone, "Autonomous transfer for reinforcement learning.," in *AAMAS (1)*, 2008, pp. 283–290.

[20] F. Garcia and E. Rachelson, "Markov decision processes," *Markov Decision Processes in Artificial Intelligence*, pp. 1–38, 2013.

[21] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[22] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, PMLR, 2018, pp. 1587–1596.

[23] S. Khadka, S. Majumdar, T. Nassar, *et al.*, "Collaborative evolutionary reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2019, pp. 3341–3350.

[24] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, IGI global, 2010, pp. 242–264.

[25] T. G. Karimpanal and R. Bouffanais, "Self-organizing maps as a storage and transfer mechanism in reinforcement learning," *arXiv preprint arXiv:1807.07530*, 2018.

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

[27] D. Mindrila and P. Balentyne, "Scatterplots and correlation," *Retrieved from*, 2017.