Transfer in Reinforcement Learning: a Framework and a Survey

Alessandro Lazaric

Transfer in Reinforcement Learning: a Framework and a Survey

Alessandro Lazaric

Abstract Transfer in reinforcement learning is a novel research area that focuses on the development of methods to transfer knowledge from a set of source tasks to a target task. Whenever the tasks are *similar*, the transferred knowledge can be used by a learning algorithm to solve the target task and significantly improve its performance (e.g., by reducing the number of samples needed to achieve a nearly optimal performance). In this chapter we provide a formalization of the general transfer problem, we identify the main settings which have been investigated so far, and we review the most important approaches to transfer in reinforcement learning.

1 Introduction

The idea of transferring knowledge across different but related tasks to improve the performance of machine learning (ML) algorithms stems from psychology and cognitive science research. A number of psychological studies (see e.g., Thorndike and Woodworth, 1901; Perkins et al, 1992) show that humans are able to learn a task better and faster by transferring the knowledge retained from solving similar tasks. Transfer in machine learning has the objective to design *transfer* methods that analyze the knowledge collected from a set of source tasks (e.g., samples, solutions) and transfer it so as to *bias* the learning process on a target task towards a set of *good* hypotheses. If the transfer method successfully identifies the similarities between source and target tasks, then the transferred knowledge is likely to improve the learning performance on the target task. The idea of retaining and reusing knowledge to improve the learning algorithms dates back to early stages of ML. In fact, it is widely recognized that a good representation is the most critical aspect of any learning algorithm, and the development of techniques that automatically change

Alessandro Lazaric
INRIA Lille-Nord Europe, 40 Avenue Halley, 59650 Villeneuve d'Ascq, France, e-mail: alessandro.lazaric@inria.fr

the representation according to the task at hand is one of the main objectives of large part of the research in ML. Most of the research in transfer learning (Fawcett et al, 1994) identified the single-problem perspective usually adopted in ML as a limit for the definition of effective methods for the inductive construction of good representations. On the other hand, taking inspiration from studies in psychology and neuroscience (Gentner et al, 2003; Gick and Holyoak, 1983), the transfer point of view, where learning tasks are assumed to be related and knowledge is retained and transferred, is considered as the most suitable perspective to design effective techniques of inductive bias (Utgoff, 1986).

Transfer in reinforcement learning. Transfer algorithms have been successful in improving the performance of learning algorithms in a number of supervised learning problems, such as recommender systems, medical decision making, text classification, and general game playing. In recent years, the research on transfer also focused on the reinforcement learning (RL) paradigm and how RL algorithms could benefit from knowledge transfer. In principle, traditional reinforcement learning already provides mechanisms to learn solutions for any task without the need of human supervision. Nonetheless, the number of samples needed to learn a nearly-optimal solution is often prohibitive in real-world problems unless prior knowledge from a domain expert is available. Furthermore, every time the task at hand changes the learning process must be restarted from scratch even when similar problems have been already solved. Transfer algorithms automatically build prior knowledge from the knowledge collected in solving a set of similar source tasks (i.e., *training* tasks) and use it to bias the learning process on any new task (i.e., *testing* task). The result is a dramatic reduction in the number of samples and a significant improvement in the accuracy of the learned solution.

Aim of the chapter. Unlike supervised learning, reinforcement learning problems are characterized by a large number of elements such as the dynamics and the reward function, and many different transfer settings can be defined depending on the differences and similarities between the tasks. Although relatively recent, research on transfer in reinforcement learning already counts a large number of works covering many different transfer problems. Nonetheless, it is often difficult to have a clear picture of the current state-of-the-art in transfer in RL because of the very different approaches and perspectives adopted in dealing with this complex and challenging problem. The aim of this chapter is to formalize what the main transfer settings are and to classify the algorithmic approaches according to the kind of knowledge they transfer from source to target tasks. Taylor and Stone (2009) also provide a thorough survey of transfer in reinforcement learning. While their survey provides a very in-depth analysis of each transfer algorithm, the objective of this chapter is not to review all the algorithms available in the literature but rather to identify the characteristics shared by the different approaches of transfer in RL and classify them into large families.

Structure of the chapter. The rest of the chapter is organized as follows. In Section 2 we formalize the transfer problem and we identify three main dimensions to categorize the transfer algorithms according to the setting, the transferred knowledge, and the objective. Then we review the main approaches of transfer in RL in

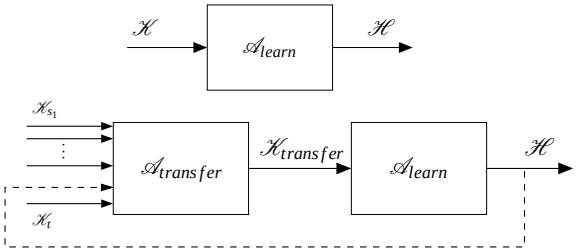

Fig. 1 *(top)* In the standard learning process, the learning algorithm gets as input some form of knowledge about the task (i.e., samples, structure of the solutions, parameters) and returns a solution. *(bottom)* In the transfer setting, a transfer phase first takes as input the knowledge retained from a set of source tasks and returns a new knowledge which is used as input for the learning algorithm. The dashed line represents the possibility to define a continual process where the experience obtained from solving a task is then reused in solving new tasks.

three different settings. In Section 3 we focus on the source-to-target setting where transfer occurs from one single source task to one single target task. A more general setting with a set of source tasks and one target task is studied in Section 4. Finally, in Section 5 we discuss the general source-to-target setting when the state-action spaces of source and target tasks are different. In Section 6 we conclude and we discuss open questions.

2 A Framework and a Taxonomy for Transfer in Reinforcement Learning

Transfer learning is a general problem and it is difficult to provide a formal definition able to take into account all the possible perspectives and approaches to the problem. Furthermore, although many different algorithms have been already proposed, a clear categorization of the main approaches to transfer in RL is still missing. In this section we first introduce a formalization of the general transfer and we then propose a taxonomy to classify transfer approaches along three main dimensions.

2.1 Transfer Framework

In this section, we adapt the formalisms introduced by Baxter (2000) and Silver (2000) for supervised learning to the RL paradigm and we introduce general definitions and symbols used throughout the rest of the chapter.

As discussed in the introduction, transfer learning leverages on the knowledge collected from a number of different tasks to improve the learning performance in new tasks. We define a *task M* as an MDP (Sutton and Barto, 1998) characterized

Symbol	*Meaning*
M	MDP
S_M	State space
A_M	Action space
T_M	Transition model (dynamics)
R_M	Reward function
$\mathscr{M}$	Task space (set of tasks M)
Ω	Probability distribution over $\mathscr{M}$
$\mathscr{E}$	Environment (task space and distribution)
$\mathscr{H}$	Hypothesis space (e.g., value functions, policies)
$h \in \mathscr{H}$	Hypothesis (e.g., one value function, one policy)
$\mathscr{K}$	Knowledge space (e.g., samples and basis functions)
$K \in \mathscr{K}$	Knowledge (e.g., specific realization of samples)
$\mathscr{K}_s$	Knowledge space from a source task
$\mathscr{K}_t$	Knowledge space from a target task
$\mathscr{K}_{\text{transfer}}$	Knowledge space return by the transfer algorithm and used in learning
$\mathscr{F}$	Space of functions defined on a specific state-action space
ϕ	State-action basis function
$\mathscr{A}_{\text{learn}}$	Learning algorithm
$\mathscr{A}_{\text{transfer}}$	Transfer algorithm
O	Set of options
$o \in O$	An option

Table 1 List of the main symbols used in the chapter.

by the tuple $\langle S_M, A_M, T_M, R_M \rangle$ where S_M is the state space, A_M is the action space, T_M is the transition function, and R_M is the reward function. While the state-action space $S_M \times A_M$ defines the *domain* of the task, the transition T_M and reward function R_M define the *objective* of the task. The *space of tasks* involved in the transfer learning problem is denoted by $\mathscr{M} = \{M\}$. Let Ω be a probability distribution over the space of tasks $\mathscr{M}$, then we denote by $\mathscr{E} = \langle \mathscr{M}, \Omega \rangle$ the *environment*, which defines the setting of the transfer problem. The tasks presented to the learner are drawn from the task distribution (i.e., $M \sim \Omega$). This general definition resembles the traditional supervised learning setting where training samples are drawn from a given distribution. As a result, similar to classification and regression, transfer learning is based on the idea that since tasks are drawn from the same distribution, an algorithm able to achieve a good performance on average on a finite number of source tasks (or training tasks), then it will also generalize well across the target tasks in $\mathscr{M}$ coming from the same distribution Ω (or testing tasks).

A standard learning algorithm takes as input some form of knowledge of the task at hand and returns a solution in a set of possible results. We use $\mathscr{K}$ to denote the space of the knowledge used as input for the learning algorithm and $\mathscr{H}$ for the space of hypotheses that can be returned. In particular, $\mathscr{K}$ refers to all the elements used by the algorithm to compute the solution of a task, notably the *instances* (e.g., samples), the *representation* of the problem (e.g., set of options, set of features), and *parameters* (e.g., learning rate) used by the algorithm. Notice that $\mathscr{K}$ includes *prior* knowledge provided by an expert, *transfer* knowledge obtained from a transfer al-

gorithm, and *direct* knowledge collected from the task. A general learning algorithm is defined as the mapping

$$\mathscr{A}_{\text{learn}} : \mathscr{K} \to \mathscr{H}. \tag{1}$$

Example 1. Let us consider fitted Q-iteration (Ernst et al, 2005) with a linear function approximator. Fitted Q-iteration first collects N samples (the *instances*) and through an iterative process returns an action-value function which approximates the optimal action-value function of the task. In this case, the hypothesis space $\mathscr{H}$ is the linear space spanned by a set of d features $\{\varphi_i : S \times A \to \Re\}_{i=1}^{d}$ designed by a domain expert, that is $\mathscr{H} = \{h(\cdot,\cdot) = \sum_{i=1}^{d} \alpha_i \varphi_i(\cdot,\cdot)\}$. Beside this prior knowledge, the algorithm also receives as input a set of N samples $\langle s,a,s',r\rangle$. As a result, the knowledge used by fitted Q-iteration can be formalized by the space $\mathscr{K} = \left((S \times A \times S \times R)^N, \mathscr{F}^d\right)$, where any specific instance $K \in \mathscr{K}$ is $K = (\{\langle s_n, a_n, r_n, s'_n\rangle\}_{n=1}^{N}, \{\varphi_i\}_{i=1}^{d})$, with $\varphi_i \in \mathscr{F}$. Given as input $K \in \mathscr{K}$ the algorithm returns an action-value function $h \in \mathscr{H}$ (i.e., $\mathscr{A}_{FQI}(K) = h$). □

Given the previous definitions, we can now define the general shape of transfer learning algorithms. In general, in single-task learning only the instances are directly collected from the task at hand, while the representation of the problem and the parameters are given as a prior by an expert. In transfer learning, the objective is to reduce the need for instances from the target task and prior knowledge from a domain expert by tuning and adapting the structure of the learning algorithm (i.e., the knowledge used as input) on the basis of the previous tasks observed so far. Let $\mathscr{E} = \langle \mathscr{M}, \Omega \rangle$ be the environment at hand and L be the number of tasks drawn from $\mathscr{M}$ according to the distribution Ω used as source tasks, a transfer learning algorithm is usually the result of a *transfer of knowledge* and a *learning* phase. Let $\mathscr{K}_s^L$ be the knowledge collected from the L source tasks and $\mathscr{K}_t$ the knowledge available (if any) from the target task. The transfer phase is defined as

$$\mathscr{A}_{\text{transfer}} : \mathscr{K}_s^L \times \mathscr{K}_t \to \mathscr{K}_{\text{transfer}}, \tag{2}$$

where $\mathscr{K}_{\text{transfer}}$ is the final knowledge transferred to the learning phase. In particular, the learning algorithm is now defined as

$$\mathscr{A}_{\text{learn}} : \mathscr{K}_{\text{transfer}} \times \mathscr{K}_t \to \mathscr{H}. \tag{3}$$

Example 2. Let us consider the transfer algorithm introduced by Lazaric (2008) in which a set of features is learned from a set of L source tasks. In this case $\mathscr{A}_{\text{transfer}}$ takes as input N_s samples for each of the L tasks and returns d features $\{\varphi_i\}_{i=1}^{d}$ from $\mathscr{F}$. The fitted Q-iteration algorithm is then used to learn the solution of a target task and $\mathscr{A}_{\text{learn}}$ takes as input N_t target samples and the features extracted during the transfer phase and returns a function in the space $\mathscr{H}$ spanned by the features $\{\varphi_i\}_{i=1}^{d}$. Thus, we have $\mathscr{K}_s = (S \times A \times S \times R)^{N_s}$, $\mathscr{K}_t = (S \times A \times S \times R)^{N_t}$, and $\mathscr{K}_{\text{transfer}} = \mathscr{F}^d$. □

Although in the definition in Equation (2) $\mathscr{K}_t$ is present in both the transfer and learning phase, in most of the transfer settings, no knowledge about the target is

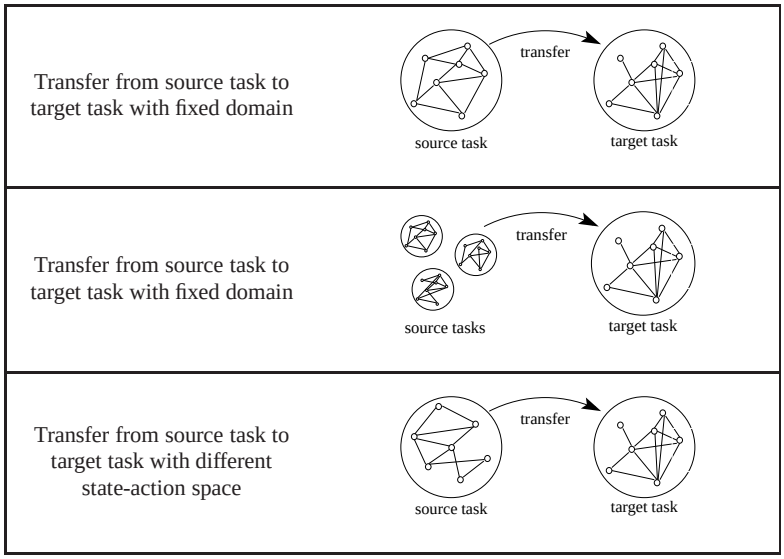

Fig. 2 The three main transfer settings defined according to the number of source tasks and their difference w.r.t. the target task.

available in the transfer phase. This formalization also shows that transfer algorithms must be compatible with the specific learning algorithm employed in the second phase, since $\mathcal{K}_{\text{transfer}}$ is used as an additional source of knowledge for $\mathcal{A}_{\text{learn}}$. The performance of the transfer algorithm is usually compared to a learning algorithm in Equation (1) which takes as input only $\mathcal{K}_t$. As discussed in the next section, the specific setting $\mathcal{E}$, the knowledge spaces $\mathcal{K}$, and the way the performance is measured define the main categories of transfer problems and approaches.

2.2 Taxonomy

In this section we propose a taxonomy of the major approaches to transfer in reinforcement learning. We define three main dimensions: the *setting*, the *transferred knowledge*, and the *objective*.

2.2.1 The Settings

In the general formulation of the transfer problem we define an environment $\mathcal{E}$ as the space of tasks $\mathcal{M}$ and the probability distribution Ω on it. Unlike other learning paradigms (see Pan and Yang (2010) for a review of the possible settings in supervised learning), an RL problem is defined by different elements such as the

dynamics and the reward, and the tasks in $\mathscr{M}$ may differ in a number of possible ways depending on the similarities and differences in each of these elements. For instance, in the transfer problem considered by Mehta et al (2008) all the tasks share the same state-action space and dynamics but the reward functions are obtained as linear combinations of basis reward functions and a weight vector. In this case, the space of tasks $\mathscr{M}$ is the set of MDPs which can be generated by varying the weights of the reward functions. Furthermore, although in the general definition tasks are drawn from a distribution Ω, there are many transfer settings in which the tasks are fixed in advance and no generalization over other tasks is considered. For instance, most of the inter-task mapping approaches (see e.g., Taylor et al, 2007a) focus on the setting in which only one source task and one target task are available. Although an implicit assumption of similarity is usually made, the tasks are simply given as input to the algorithm and no explicit distribution is defined.

In the following we will distinguish among three different categories of transfer settings (see Figure 2).

(I) *Transfer from source task to target task with fixed domain.* As defined in Section 2.1 the domain of a task is determined by its state-action space $S_M \times A_M$, while the specific structure and goal of the task are defined by the dynamics T_M and reward R_M. Most of the early literature in transfer in RL focused on the setting in which the domain is fixed and only two tasks are involved: a *source* task and a *target* task. This setting is usually referred to *inductive transfer learning* in the supervised learning literature (Pan and Yang, 2010). The transfer algorithm might or might not have access to the target task at transfer time. If no target knowledge is available, some of the transfer algorithms perform a shallow transfer of the knowledge collected in the source task (e.g., the policy) and directly use it in the target task. Other algorithms try to abstract from the source task some general characteristics (e.g., subgoals) that are likely to be relevant in solving target tasks sharing the same characteristics. On the other hand, when some target knowledge is available at transfer time, then it is used to *adapt* the source knowledge to the target task. For instance, in (Taylor et al, 2008b) target samples are used to identify the best mapping between source and target state-action variables and thus to transform the source policy into a target policy used to initialize the learning process in the target task.

(II) *Transfer across tasks with fixed domain.* In this setting, the general definition of environment $\mathscr{E}$ with a distribution over the task space is considered. In this case, tasks share the same domain and the transfer algorithm takes as input the knowledge collected from a set of source tasks and use it to improve the performance in the target task. In this setting, the objective is usually to generalize over the tasks in $\mathscr{M}$ according to the distribution Ω. Similar to supervised learning, we expect that, as the number of source tasks increases, the transfer algorithm is able to improve the average performance on the target tasks drawn from Ω when compared to a single-task learning algorithm which does not use any transferred knowledge.

(III) *Transfer across tasks with different domains.* Finally, in this setting tasks have a different domain, that is they might have different state-action variables, both in

terms of number and range. Most of the transfer approaches in this case consider the source-target scenario and focus on how to define a mapping between the source state-action variables and the target variables so as to obtain an effective transfer of knowledge.

2.2.2 The Knowledge

The definition of transferred knowledge and the specific transfer process are the main aspects characterizing a transfer learning algorithm. In the definition of Section 2.1 the space $\mathscr{K}$ contains the *instances* collected from the environment (e.g., sample trajectories), the *representation* of the solution and the *parameters* of the algorithm itself. Once the space of knowledge considered by the algorithm is defined, it is important to design how this knowledge is actually used to transfer information from the source tasks to the target task. Silver (2000) and Pan and Yang (2010) propose a general classification of the knowledge retained and transferred across tasks in supervised learning. Taylor and Stone (2009) introduces a very detailed classification for transfer in RL. Here we prefer to have a broader classification identifying macro-categories of approaches along the lines of Lazaric (2008). We classify the possible knowledge transfer approaches into three categories: *instance* transfer, *representation* transfer, *parameter* transfer.

(I) *Instance transfer.* Unlike dynamic programming algorithms, where the dynamics and reward functions are known in advance, all the RL algorithms rely on a set of samples collected from a direct interaction with the MDP to build a solution for the task at hand. This set of samples can be used to estimate the model of the MDP in *model-based* approaches or to directly build an approximation of the value function or policy in *model-free* approaches. The most simple version of transfer algorithm collects samples coming from different source tasks and reuses them in learning the target task. For instance, the transfer of trajectory samples can be used to simplify the estimation of the model of new tasks (Sunmola and Wyatt, 2006) or the estimation of the action value function as in (Lazaric et al, 2008).

(II) *Representation transfer.* Each RL algorithm uses a specific representation of the task and of the solution, such as state-aggregation, neural networks, or a set of basis functions for the approximation of the optimal value function. After learning on different tasks, transfer algorithms often perform an abstraction process which changes the representation of the task and of the solutions. In this category, many possible approaches are possible varying from reward shaping (Konidaris and Barto, 2006) and MDP augmentation through options (Singh et al, 2004) to basis function extraction (Mahadevan and Maggioni, 2007).

(III) *Parameter transfer.* Most of the RL algorithms are characterized by a number of parameters which define the initialization and the behavior of the algorithm itself. For instance, in Q-learning (Watkins and Dayan, 1992) the Q-table is initialized with arbitrary values (e.g., the highest possible value for the action values $R_{max}/(1-\gamma)$) and it is updated using a gradient-descent rule with a learning rate

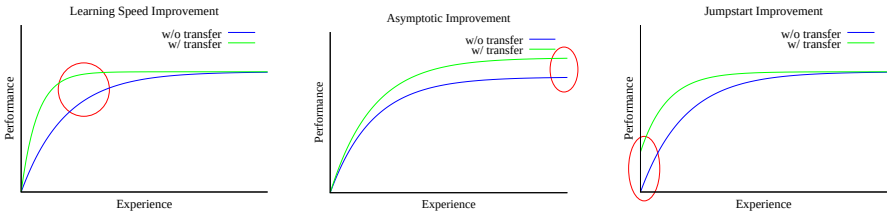

Fig. 3 The three main objectives of transfer learning (Langley, 2006). The red circles highlight the improvement in the performance in the learning process expected by using transfer solutions w.r.t. single-task approaches.

α. The initial values and the learning rate define the set of input parameters used by the algorithm. Some transfer approaches change and adapt the algorithm parameters according to the source tasks. For instance, if the action values in some state-action pairs are very similar across all the source tasks, the Q-table for the target task could be initialized to more convenient values thus speeding-up the learning process. In particular, the transfer of initial solutions (i.e., policies or value functions) is commonly adopted to initialize the learning algorithm in the transfer setting with only one source task.

2.2.3 The Objectives

While in supervised learning the performance of a classifier or a regressor are usually measured in terms of prediction error, in RL many possible measures can be employed to evaluate how good is the solution returned by the learning algorithm. As a result, transfer algorithms can be evaluated according to a number of different performance measures. Depending on how the learning performance is measured, different transfer metrics may be used. In Taylor and Stone (2009) a number of metrics is proposed to measure the improvement of transfer over single-task approaches. Here we discuss three main transfer objectives adapted from the objectives suggested for the general problem of transfer suggested by Langley (2006) (see Figure 3):

(I) *Learning speed improvement*. This objective is about the reduction in the amount of the experience needed to learn the solution of the task at hand. As new tasks are sampled according to Ω, the knowledge retained from a set of previously solved tasks can be used to bias the learning algorithm towards a limited set of solutions, so as to reduce its learning time. The complexity of a learning algorithm is usually measured by the number of samples needed to achieve a desired performance. In RL, this objective is pursued following two different approaches. The first approach is to make the algorithm more effective in using the experience collected from the exploration of the environment. For instance, Kalmar and Szepesvari (1999) and Hauskrecht (1998) show that the use of options can

improve the effectiveness of value iteration backups by updating value function estimates with the total reward collected by an option, and thus reducing the number of iterations to converge to a nearly optimal solution. The second aspect is about the strategy used to collect the samples. In online RL algorithms samples are collected from direct interaction with the environment through an exploration strategy. The experience collected by solving a set of tasks can lead to the definition of better exploration strategies for new related tasks. For instance, if all the tasks have goals in a limited region of the state space, an exploration strategy that frequently visits that region will lead to more informative samples.

In practice, at least three different methods can be used to measure the improvement in the learning speed: *time to threshold*, *area ratio*, and *finite-sample analysis*. In all the problems where a target performance is considered (e.g., a small enough number of steps-to-go in a navigation problem), it is possible to set a threshold and measure how much experience (e.g., samples, episodes, iterations) is needed by the single-task and transfer algorithms to achieve that threshold. If the transfer algorithm successfully takes advantage of the knowledge collected from the previous tasks, we expect it to need much less experience to reach the target performance. The main drawback of this metric is that the threshold might be arbitrary and that it does not take into account the whole learning behavior of the algorithms. In fact, it could be the case that an algorithm is faster in reaching a given threshold but it has a very poor initial performance or does not achieve the asymptotic optimal performance. The area ratio metric introduced by Taylor and Stone (2009) copes with this problem by considering the whole area under the learning curves with and without transfer. Formally, the area ratio is defined as

$$r = \frac{\text{area with transfer} - \text{area without transfer}}{\text{area without transfer}}. \tag{4}$$

Although this metric successfully takes into consideration the behavior of the algorithms until a given number of samples, it is scale dependent. For instance, when the reward-per-episode is used as a measure of performance, the scale of the rewards impacts on the area ratio and changes in the rewards might lead to different conclusions in the comparison of different algorithms. While the two previous measures allow to empirically compare the learning performance with and without transfer, it is also interesting to have a more rigorous comparison by deriving sample-based bounds for the algorithms at hand. In such case, it is possible to compute an upper bound on the error of the solution returned by the algorithm depending on the parameters of the task and the number of samples is available. For instance, if the algorithm returns a function $h \in \mathscr{H}$ and Q^* is the optimal action value function, a finite sample bound is usually defined as

$$||h - Q^*||_\rho \leq \varepsilon_1(\mathscr{H}, Q^*) + \varepsilon_2(N), \tag{5}$$

where ρ is a distribution over the state space S, $\varepsilon_1(\mathscr{H}, Q^*)$ is the *approximation* error and it accounts for the asymptotic error of the best possible solution in $\mathscr{H}$, and $\varepsilon_2(N)$ is the *estimation* error and it decreases with the number of samples.

Transfer algorithms should be able to reduce the estimation error such that with the same number of samples as in the single-task algorithm, they could achieve a better performance. Although recent works in RL provide finite-sample analysis for a number of popular algorithms such as fitted value iteration (Munos and Szepesvári, 2008), LSTD (Farahmand et al, 2008; Lazaric et al, 2010), and Bellman residual minimization (Antos et al, 2008; Maillard et al, 2010), at the best of our knowledge, at the moment there is no finite-sample analysis for any transfer algorithm in RL.

As it will be reviewed in next sections, this objective is usually pursued by instance-transfer by adding source samples to the set of samples used to learn the target task and by parameter-transfer approaches by initializing the learning process to a convenient solution. Representation-transfer algorithms achieve a learning speed improvement by augmenting the current representation of the task (e.g., adding options to the action set) and of the solutions (i.e., adding features).

(II) *Asymptotic improvement*. In most of the problems of practical interest, a perfect approximation of the optimal value function or policy is not possible (e.g., problems with continuous state-action spaces) and the use of function approximation techniques is mandatory. The more accurate the approximation, the better the generalization (and the performance) at convergence. The accuracy of the approximation is strictly dependent on the structure of the space of hypotheses $\mathscr{H}$ used to represent the solution (e.g., value functions). This objective is usually targeted by representation-transfer algorithms which adapt the structure of $\mathscr{H}$ (e.g., by changing the features in a linear approximation space) so as to accurately approximate the solutions of the tasks in $\mathscr{M}$. An empirical measure of the quality of $\mathscr{H}$ is to compare the asymptotic performance (i.e., when a large number of samples is available) of transfer and single-task learning. Also in this case it would be interesting to analyze the effectiveness of the transfer algorithms by providing a finite-sample analysis of their performance. In particular, the asymptotic improvement corresponds to a better approximation error term in the bound of Equation (5). Similar to the learning speed improvement, at the moment no transfer algorithm is guaranteed to improve the average approximation error over the tasks in $\mathscr{M}$.

(III) *Jumpstart improvement*. The learning process usually starts from either a random or an arbitrary hypothesis h in the hypothesis space $\mathscr{H}$. According to the definition of environment, all the tasks are drawn from the same distribution Ω. As a result, after observing a number of source tasks, the transfer algorithm may build an effective prior on the solution of the tasks in $\mathscr{M}$ and initialize the learning algorithm to a suitable initial hypothesis with a better performance w.r.t. to a random initialization. It is worth noting that this objective does not necessarily correspond to an improvement in the learning speed. Let us consider a source task whose optimal policy is significantly different from the optimal policy of the target task but that, at the same time, it achieves only a slightly suboptimal performance (e.g., two goal states with different final positive rewards in different regions of the state space). In this case, the improvement of the initial performance can be obtained by initializing the learning algorithm to the optimal

Setting	Knowledge	Objective
Transfer from source to target with fixed domain	Instance	Learning speed
Transfer across tasks with fixed domain	Representation	Asymptotic performance
Transfer across tasks with different domains	Parameter	Jumpstart

Table 2 The three dimensions of transfer learning in RL. Each transfer solution is specifically designed for a *setting*, it transfers some form of *knowledge*, and it pursues an *objective*. The survey classifies the existing algorithms according to the first dimension, it then reviews the approaches depending on the transferred knowledge and discusses which objectives they achieve.

policy of the source task, but this may lead to worsen the learning speed. In fact, the initial policy does not provide samples of the actual optimal policy of the task at hand, thus slowing down the learning algorithm. On the other hand, it could be possible that the policy transferred from the source task is an effective exploration strategy for learning the optimal policy of the target task, but that it also achieves very poor performance. This objective is usually pursued by parameter-transfer algorithms in which the learning algorithm is initialized with a suitable solution whose performance is better compared to a random (or arbitrary) initialization.

2.2.4 The Survey

Given the framework introduced in the previous sections, the survey is organized along the dimensions in Table 2. In the following sections we first classify the main transfer approaches in RL according to the specific setting they consider. In each setting, we further divide the algorithms depending on the type of knowledge they transfer from source to target, and, finally, we discuss which objectives are achieved. As it can be noticed, the literature on transfer in RL is not equally distributed on the three settings. Most of the early literature on transfer in RL focused on the source-to-target setting, while the most popular scenario of recent research is the general problem of transfer from a set of source tasks. Finally, research on the problem of mapping different state and action spaces mostly relied on hand-coded transformations and much room for further investigation is available.

3 Methods for Transfer from Source to Target with a Fixed State-Action Space

In this section we consider the most simple setting in which transfer occurs from one source task to a target task. We first formulate the general setting in the next section and we then review the main approaches to this problem by categorizing them according to the type of transferred knowledge. Most of the approaches reviewed in the following change the representation of the problem or directly transfer the source solution to the target task. Furthermore, unlike the other two settings considered in

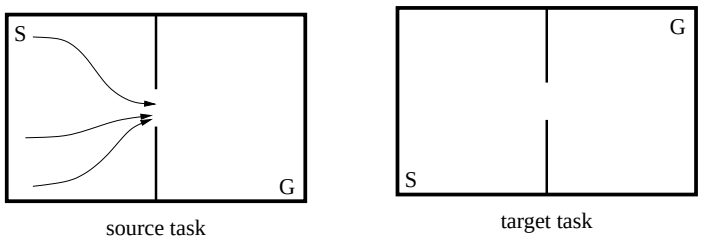

Fig. 4 Example of the setting of transfer from source to target with a fixed state-action space.

Section 4 and 5, not all the possible knowledge transfer models are considered and at the best of our knowledge no instance-transfer method has been proposed for this specific setting.

3.1 Problem Formulation

In this transfer setting we define two MDPs, a *source* task $M_s = \langle S, A, T_s, R_s \rangle$ and a *target* task $M_t = \langle S, A, T_t, R_t \rangle$, sharing the same state-action space $S \times A$. The environment $\mathscr{E}$ is defined by the task space $\mathscr{M} = \{M_s, M_t\}$ and a task distribution Ω which simply returns M_s as the first task and M_t as second.

Example 3. Let us consider the transfer problem depicted in Figure 4. The source task is a navigation problem where the agent should move from the region marked with S to the goal region G. The target task shares exactly the same state-action space and the same dynamics as the source task but the initial state and the goal (and the reward function) are different. The transfer algorithm first collect some form of knowledge from the interaction with the source task and then generates a transferrable knowledge that can be used as input to the learning algorithm on the target task. In this example, the transfer algorithm can exploit the similarity in the dynamics and identify regularities that could be useful in learning the target task. As reviewed in the next section, one effective way to perform transfer in this case is to discover policies (i.e., options) useful to navigate in an environment with such a dynamics. For instance, the policy sketched in Figure 4 allows the agent to move from any point in the left room to the door between the two rooms. Such a policy is useful to solve any navigation task requiring the agent to move from a starting region in the left room to a goal region in the right room. Another popular approach is to discover features which are well-suited to approximate the optimal value functions in the environment. In fact, the dynamics displays symmetries and discontinuities which are likely to be preserved in the value functions. For instance, both the source and target value functions are discontinuous close to the walls separating the two rooms. As a result, once the source task is solved, the transfer algorithm should analyze the dynamics and the value function and return a set of features which capture this discontinuity and preserve the symmetries of the problem. □

3.2 Representation Transfer

In some transfer problems no knowledge about the target task is available before transfer actually takes place and $\mathscr{K}_t$ in Equation (2) is always empty. In this case, it is important to abstract from the source task general characteristics that are likely to apply to the target task as well. The transfer algorithm first collects some knowledge from the source task and it then changes the representation either of the solution space $\mathscr{H}$ or of the MDP so as to speed-up the learning in the target task.

Option discovery. One of the most popular approaches to the source-target transfer problem is to change the representation of the MDP by adding *options* (Sutton et al, 1999) to the set of available actions (see Chapter **??** for a review of hierarchical RL methods). In discrete MDPs, options do not affect the possibility to achieve the optimal solution (since all the primitive actions are available, any possible policy can still be represented), but they are likely to improve the learning speed if they reach regions of the state space which are useful to learn the target task. All the option-transfer methods consider discrete MDPs, a tabular representation of the action-value function, and source and target tasks which differ only in the reward function (i.e., $T_s = T_t$). The idea is to exploit the structure of the dynamics shared by the two tasks and to ignore the details about the specific source reward function. Most of these methods share a common structure. A set of samples $\langle s_i, a_i, r_i, s'_i \rangle$ is first collected from the source task and an estimated MDP $\hat{M}_s$ is computed. On the basis of the characteristics of the estimated dynamics a set of relevant subgoals is identified and a set of d options is learned to reach each of them. According to the model in Section 2.1, the source knowledge is $\mathscr{K}_s = (S \times A \times S \times R)^{N_s}$, and for any specific realization $K \in \mathscr{K}_s$, the transfer algorithm returns $\mathscr{A}_{\text{transfer}}(K) = (O, \mathscr{H})$, where $O = \{o_i\}_{i=1}^{d}$ and $\mathscr{H} = \{h : S \times \{A \cup O\} \to \Re\}$. The learning algorithm can now use the new augmented action space to learn the solution to the target task using option Q-learning (Sutton et al, 1999). Although all these transfer algorithms share the same structure, the critical point is how to identify the subgoals and learn options from the estimated dynamics. McGovern and Barto (2001) define the concept of bottleneck state as a state which is often traversed by the optimal policy of the source task and that can be considered as critical to solve tasks in the same MDP. Metrics defined for graph partitioning techniques are used in (Menache et al, 2002) and (Simsek et al, 2005) to identify states connecting different regions of the state space. Hengst (2003) proposes a method to automatically develop a MAXQ hierarchy on the basis of the concept of access states. Finally, Bonarini et al (2006) a psychology-inspired notion of interest aimed at identifying states from which the environment can be easily explored.

Action space transfer. A different approach to representation-transfer involving the action space is proposed in (Sherstov and Stone, 2005). Using random task perturbation, a set of tasks is artificially generated from one single source task and a new action set is obtained by removing from A all the actions which are not optimal in any of the source tasks. In this case, the transfer algorithm returns a pair $(A', \mathscr{H})$ where $A' \subseteq A$ is a subset of the original action space A and $\mathscr{H} = \{h : S \times A' \to \Re\}$. With a smaller action set the learning speed in the target task is significantly im-

proved at the cost of a loss in the optimality of the learned policy. In fact, some of the actions removed according to the artificially generated source tasks might be optimal in the target task. Nonetheless, if source and target tasks are similar and the perturbed tasks are different enough from the source task, then the method is likely to preserve most of the actions necessary to solve the target task.

Feature discovery. In (Mahadevan and Maggioni, 2007; Ferguson and Mahadevan, 2006; Ferrante et al, 2008) a representation-transfer approach similar to option-transfer is proposed. The main difference is that instead of options, the transfer algorithm extracts a set of features $\{\varphi_i\}_{i=1}^{d}$ which defines the hypothesis space $\mathscr{H}$. Similar to the option-transfer algorithms, the tasks are assumed to share the same dynamics and an estimated transition model $\hat{T}_s$ is used to extract the features. While the source knowledge $\mathscr{K}_s$ is again the set of samples collected from M_s, the transferred knowledge is $\mathscr{K}_{\text{transfer}} = \mathscr{F}^d$ and once a specific set of features $\{\varphi_i\}_{i=1}^{d}$ (with $\varphi_i \in \mathscr{F}$) is extracted, the solution space is defined as $\mathscr{H} = \{h(x,a) = \sum_{i=1}^{d} \alpha_i \varphi_i(x,a)\}$. Furthermore, in option-transfer the objective is to improve the learning speed, while feature-transfer aims at achieving a better approximation of the target value function (i.e., asymptotic improvement). While option-transfer approaches are specifically designed for on-line algorithms such as option Q-learning, the feature-transfer algorithms can be paired to any RL algorithm using a linear function approximation scheme. Mahadevan and Maggioni (2007) introduces a method to generate *proto-value functions* (i.e., the features) using spectral analysis of the Laplacian of the estimated graph of the source MDP. Proto-value functions capture the intrinsic structure of the manifold underlying the dynamics of the tasks at hand (e.g., symmetries) and thus they are likely to approximate well the value function of any task sharing the same dynamics. Ferguson and Mahadevan (2006) generalize this approach to problems with slightly different dynamics. Finally, Ferrante et al (2008) further generalize this appraoch to a more general setting is considered in which both the dynamics and reward function may be different in source and target task. A different method is proposed to build the source graph and extract proto-value functions which are well suited to approximate functions obtained from similar dynamics and reward functions.

3.3 Parameter transfer

All the previous methods about representation transfer rely on the implicit assumption that source and target tasks are *similar* enough so that options or features extracted from the source task are effective in learning the solution of the target task. Nonetheless, it is clear that many different notions of similarity can be defined. For instance, we expect the option-transfer methods to work well whenever the two optimal policies have some parts in common (e.g., they both need passing through some specific states to achieve the goal), while proto-value functions are effective when the value functions preserve the structure of the transition graph (e.g., symmetries). The only explicit attempt to measure the expected performance of transfer

from source to target as a function of a distance between the two MDPs is pursued by Ferns et al (2004) and Phillips (2006). In particular, they analyze the case in which a policy π_s is transferred from source to target task. The learning process in the target task is then initialized using π_s and its performance is measured. If the MDPs are similar enough, then we expect this policy-transfer method to achieve a jumpstart improvement. According to the formalism introduced in Section 2.1, in this case $\mathscr{K}_s$ is any knowledge collected from the source task used to learn π_s, while the transferred knowledge $\mathscr{K}_{\text{transfer}}$ only contains π_s and no learning phase actually takes place. Phillips (2006) defines a state distance between M_s and M_t along the lines of the metrics proposed in (Ferns et al, 2004). In particular, the distance $d : S \to \Re$ is defined as

$$d(s) = \max_{a \in A} \left(|R_s(s,a) - R_t(s,a)| + \gamma \mathscr{T}(d)\left(T_s(\cdot|s,a), T_t(\cdot|s,a)\right)\right), \tag{6}$$

where $\mathscr{T}(d)$ is the Kantorovich distance which measures the difference between the two transition distributions $T_s(\cdot|s,a)$ and $T_t(\cdot|s,a)$ given the state distance d. The recursive Equation (6) is proved to have a fixed point d^* which is used a state distance. Phillips (2006) prove that when a policy π_s is transferred from source to target, its performance loss w.r.t. the optimal target policy π_t^* can be upper bounded by d^* as

$$||V_t^{\pi_s} - V_t^{\pi_t^*}|| \leq \frac{2}{1-\gamma} \max_{s \in S} d^*(s) + \frac{1+\gamma}{1-\gamma} ||V_s^{\pi_s} - V_s^{\pi_s^*}||.$$

As it can be noticed, when the transferred policy is the optimal policy π_s^* of the source task, then its performance loss is upper bounded by the largest value of d^* which takes into consideration the difference between the reward functions and transition models of the two tasks at hand.

As discussed in Section 5, many other approaches parameter-transfer approaches have been investigated in the setting of source and target tasks with different state-action spaces.

4 Methods for Transfer across Tasks with a Fixed State-Action Space

While in the previous section we considered the setting in which only one source task is available, here we review the main transfer approaches to the general setting when a set of source tasks is available. Transfer algorithms in this setting should deal with two main issues: how to merge knowledge coming from different sources and how to avoid the transfer from sources which differ too much from the target task (*negative* transfer).

4.1 Problem Formulation

In this section we consider the more general setting in which the environment $\mathscr{E}$ is defined by a set of tasks $\mathscr{M}$ and a distribution Ω. Similar to the setting in Section 3.1, here all the tasks share the same state-action space, that is for any $M \in \mathscr{M}$, $S_M = S$ and $A_M = A$. Although not all the approaches reviewed in the next section explicitly define a distribution Ω, they all rely on the implicit assumption that all the tasks involved in the transfer problem share some characteristics in the dynamics and reward function and that by observing a number of source tasks, the transfer algorithm is able to generalize well across all the tasks in $\mathscr{M}$.

Example 4. Let us consider a similar scenario to the real-time strategy (RTS) game introduced in (Mehta et al, 2008). In RTS, there is a number of basic tasks such as attacking the enemy, mining gold, building structures, which are useful to accomplish more complex tasks such as preparing an army and conquering an enemy region. The more complex tasks can be often seen as a combination of the low level tasks and the specific combination depends also on the phase of the game, the characteristics of the map, and many other parameters. A simple way to formalize to problem is to consider the case in which all the tasks in $\mathscr{M}$ share the same state-action space and dynamics but have different rewards. In particular, each reward function is the result of a linear combination of a set of d basis reward function, that is, for each task M, the reward is defined as $R_M(\cdot) = \sum_{i=1}^{d} w_i r_i(\cdot)$ where w is a weight vector and $r_i(\cdot)$ is a basis reward function. Each basis reward function encodes a specific objective (e.g., *defeat the enemy*, *collect gold*), while the weights represent a combination of them as in a multi-objective problem. It is reasonable to assume that the specific task at hand is randomly generated by setting the weight vector w. In particular, we can define a generative distribution Ω_ψ with hyper-parameters ψ from which the weights w_M are drawn. For instance, the hyper-parameter could be a pair $\psi = (\mu, \Sigma)$ and Ω_ψ could be a multivariate d-dimensional Gaussian distribution $\mathscr{N}(\mu, \Sigma)$. In this case, the objective of the transfer algorithm is to estimate as accurately as possible the parameters Ψ so as to have a reliable prior on any new weight vector w_M. □

4.2 Instance transfer

The main idea of instance-transfer algorithms is that the transfer of source samples may improve the learning on the target task. Nonetheless, if samples are transferred from sources which differ too much from the target task, then *negative* transfer might occur. In this section we review the only instance-transfer approach for this transfer setting proposed in (Lazaric et al, 2008) which selectively transfers samples on the basis of the similarity between source and target tasks.

Let L be the number of source tasks, Lazaric et al (2008) propose an algorithm proposed which first collects N_s samples for each source task $\mathscr{K}_s = (S \times A \times S \times R)^{N_s}$ and N_t samples (with $N_t \ll N_s$) from the target task $\mathscr{K}_t = (S \times A \times S \times R)^{N_t}$, and the

transfer algorithm takes as input $\mathscr{K}_s$ and $\mathscr{K}_t$. Instead of returning as output a set containing all the source samples, the method relies on a measure of similarity between the source and the target tasks to select which source samples should be included in $\mathscr{K}_{\text{transfer}}$. Let $K_{s_l} \in \mathscr{K}_s$ and $K_t \in \mathscr{K}_t$ be the specific source and target samples available to the transfer algorithm. The number of source samples is assumed to be large enough to build an accurate kernel-based estimation of each source model $\hat{M}_{s_l}$. Given the estimated model, the similarity between the source task M_{s_l} and the target task M_t is defined as

$$\Lambda_{s_l} = \frac{1}{N_t} \sum_{n=1}^{N_t} \mathbb{P}\left(\langle s_n, a_n, s'_n, r_n \rangle | \hat{M}_{s_l}\right)$$

where $\mathbb{P}\left(\langle s_n, a_n, s'_n, r_n \rangle | \hat{M}_{s_l}\right)$ is the probability of the transition $\langle s_n, a_n, s'_n, r_n \rangle \in K_t$ according to the (estimated) model of M_{s_l}. The intuition behind this measure of similarity is that it is more convenient to transfer samples collected from source tasks which are likely to generate target samples. Finally, source samples are transferred proportionally to their similarity Λ_{s_l} to the target task. The method is further refined using another measure of utility for each source sample so that from each source task only the samples that are more likely to improve the learning performance in the target task. In the experiments reported in (Lazaric et al, 2008) this method is shown to successfully identify which sources are more relevant to transfer samples from and to avoid negative transfer.

4.3 Representation Transfer

Unlike in the source-to-target transfer, the objective of representation transfer in this setting is to infer from the source tasks general characteristics that are preserved across the tasks in $\mathscr{M}$ and that can be effectively exploited to improve the average (measured according to the distribution Ω) learning performance w.r.t. single-task learning.

Option transfer. Bernstein (1999) introduce a new option (called *reuse* option) which is added to the set of available actions and which is built as a combination of the optimal policies learned on the L source tasks and it is then reused to speed-up learning on the target task. The process can be re-iterated so that the target task is added to the source tasks and the reuse option is updated accordingly. After a sufficient number of source tasks, the reuse option is shown to significantly speed-up the learning process on new tasks. Options are used to speed-up learning also in (Perkins and Precup, 1999) where a POMDP-like framework is considered. In particular, they assume the set $\mathscr{M}$ to be known in advance and a belief about the identity of the current task is estimated. The value of an option in the current task is then computed as an average of its value in different tasks weighted by the current belief. Although in both these approaches the tasks are assumed to be drawn from a common distribution Ω, they do not provide any analysis about which options

could guarantee the best improvement. Kalmar and Szepesvari (1999) consider the problem of finding the set of options which reduces the most the number of iterations needed for value iteration to converge. Unlike the previous approaches, in this case the transfer algorithm is guaranteed to return the optimal set of options. Finally, Asadi and Huber (2007) propose a method for incremental discovery of skills using the MAX-Q hierarchical architecture.

Feature transfer to speed-up learning. A different approach to representation-transfer is to identify a function space which is likely to contain functions able either to speed-up the learning process or to accurately approximate the optimal value functions in $\mathscr{M}$. Similar to option-transfer, the first objective is usually achieved in the setting of discrete MDPs in which a tabular approach is used. In this case, the space of functions $\mathscr{H} = \{h : S \times A \to \Re\}$ already guarantees the possibility to compute the optimal action value function with an arbitrary small approximation error. Nonetheless, when the number of states and actions is large, the learning process could be very slow. The augmentation of the space $\mathscr{H}$ with features which accurately approximate the optimal action-value functions of the tasks in $\mathscr{M}$ in some parts of the state-action space could considerably speed-up the learning process. The transfer phase in this case takes as input for each source task a source knowledge $K_{s_l} \in \mathscr{K}_{s_l}$ defined as $K_{s_l} = \left\{ \{\langle s_n, a_n, s'_n, r_n \rangle\}_{n=1}^{N_s}, \mathscr{H} \right\}$ while no knowledge about the target task is available. The output knowledge is a set of d new features $\mathscr{K}_{\text{transfer}} = \mathscr{F}^d$, so that $\mathscr{A}_{\text{transfer}}(\{K_{s_l}\}_{l=1}^{L}) = \{\varphi_i \in \mathscr{F}, i = 1, \ldots, d\}$ where the new features are used by the learning algorithm in addition to $\mathscr{H}$. Foster and Dayan (2002) propose an automatic method to decompose the MDPs into elemental fragments. In particular, an unsupervised method is first used to analyze the optimal value functions learned so far and to decompose the state space into fragments. Each fragment (i.e., a sub-task) can be solved independently and its value function is then used as an additional feature when learning the target task. Drummond (2002) propose a similar method which identifies subtasks and features according to the analysis of the dynamics of the source tasks. In both cases, the methods are able to identify useful features in maze problems with highly structured optimal value functions. Madden and Howley (2004) introduce a hybrid representation-parameter transfer approach. According to the Q-tables learned on the source tasks, a symbolic learner generates a set of decision rules defined on a higher-level of abstraction compared to the state features used in learning the Q-tables. This representation is then transferred to the target task together with the rules which are used to initialize the Q-table for the target task.

Feature transfer to improve asymptotic performance. While the previous methods consider discrete MDPs where it is possible to exactly compute the optimal solution and the objective is to speed-up the learning, other methods focus on the continuous state-action spaces in which function approximation is mandatory. Walsh et al (2006) consider a simple state-aggregation function approximation scheme. In this case, the objective is to find an aggregation of states able to accurately approximate all the optimal value functions of the tasks at hand. A similar objective is pursued in (Lazaric, 2008) in which the transfer algorithm identifies the best set of features to approximate the source tasks and then reuses them in solving

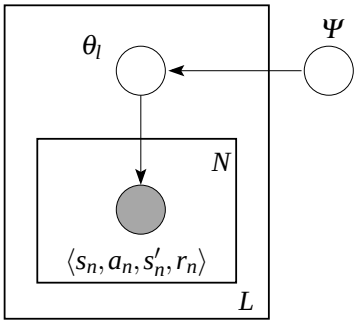

Fig. 5 Example of a generative model of a hierarchical Bayesian model (HBM). The observations $\langle s, a, s', r \rangle$ are generated according to an MDP parameterized by a parameter vector θ, while each task is generated according to a distribution defined by a set of hyper-parameters Ψ. Ψ_0 is a vector of parameters defining the prior over the hyper-parameters.

the target task. Similar to (Argyriou et al, 2008), the algorithm relies on the assumption that there exists a small subset of features which is useful to approximate the value functions of the tasks in $\mathscr{M}$ (*shared sparsity* assumption). The algorithm considers a set of features $\{\varphi_i^\theta\}_{i=1}^d$ parameterized by a parameter $\theta \in \Theta$ and the corresponding linear hypothesis space $\mathscr{H}_\theta = \{h(x,a) = \sum_{i=1}^d \alpha_i \varphi_i^\theta(x,a)\}$. The objective is to learn the parameter θ which defines a feature space such that only a small set of features are used to approximate the value functions of the tasks (i.e., features such that the corresponding optimal weights α have a small number of non-zero components). Empirical evaluation shows that whenever the optimal value functions of the tasks in $\mathscr{M}$ can be represented with a very small number of features, the algorithm is able to learn them and to obtain a significant convergence improvement.

4.4 Parameter Transfer

Unlike representation-transfer approaches, all parameter-transfer algorithms explicitly define a distribution Ω on the task space $\mathscr{M}$ and try to estimate the true distribution in order to build a prior over the space of hypotheses $\mathscr{H}$ so as to improve the initial performance (jumpstart improvement) and reduce the number of samples needed to solve any task in Ω. More formally, most of the parameter-transfer approaches share the following structure. Let $\mathscr{M} = \{M_\theta, \theta \in \Theta\}$ be a parameterized space of tasks with parameters θ and Ω_ψ a family of task probability distributions, where $\psi \in \Psi$ is a hyper-parameter vector. The main assumption is that all the task parameters θ are independently and identically distributed according to a specific task distribution Ω_{ψ^*}.

The hierarchical Bayesian model. The structure of this problem is usually represented as a hierarchical Bayesian model (HBM) as depicted in Figure 5. The trans-

fer algorithms take as input samples $K_{s_l} = \left\{ \{\langle s_n, a_n, s'_n, r_n \rangle\}_{n=1}^{N_s} \right\}$ from each of the source tasks M_{θ_l} ($l = 1, \ldots, L$) which are drawn from the true distribution Ω_{ψ^*} (i.e., $\theta_l \overset{iid}{\sim} \Omega_{\psi^*}$) whose true hyper-parameters are unknown. Given a prior over ψ, the algorithm solves the inference problem

$$\mathbb{P}\left(\psi | \{K_{s_l}\}_{l=1}^{L}\right) \propto \prod_{l=1}^{L} \mathbb{P}\left(K_{s_l} | \psi\right) \mathbb{P}(\psi), \tag{7}$$

where $\mathbb{P}\left(K_{s_l} | \psi\right) \propto \mathbb{P}\left(K_{s_l} | \theta\right) \mathbb{P}(\psi)$. The ψ with highest probability is usually transferred and used to initialize the learning process on the target task. Notice that the learning algorithm must be designed so as to take advantage of the knowledge about the specific task distribution Ω_ψ returned by the transfer phase. Bayesian algorithms for RL such as GPTD (Engel et al, 2005) are usually adopted (see Chapter **??**).

The inference problem in Equation (7) leverages on the knowledge collected on all the tasks at the same time. Thus, even if few samples per task are available (i.e., N_s is small), the algorithm can still take advantage of a large number of tasks (i.e., L is large) to solve the inference problem and learn an accurate estimate of ψ^*. As L increases the hyper-parameter ψ gets closer and closer to the true hyper-parameter ψ^* and ψ can be used to build a prior on the parameter θ for any new target task drawn from the distribution Ω_{ψ^*}. Depending on the specific definition of Θ and Ω_ψ and the way the inference problem is solved, many different algorithms can be deduced from this general model.

Inference for transfer. Tanaka and Yamamura (2003) consider a simpler approach. Although the MDPs are assumed to be drawn from a distribution Ω, the proposed algorithm does not try to estimate the task distribution but only a statistics about the action values is computed. The mean and variance of the action values over different tasks are computed and then used to initialize the Q-table for new tasks. Sunmola and Wyatt (2006) and Wilson et al (2007) consider the case where the MDP dynamics and reward function are parameterized by a parameter vector θ and they are assumed to be drawn from a common distribution Ω_ψ. The inference problem is solved by choosing appropriate conjugate priors over the hyper-parameter ψ. A transfer problem on POMDPs is consider in (Li et al, 2009). In this case, no explicit parameterization of the tasks is provided. On the other hand, it is the space of history-based policies $\mathscr{H}$ which is parameterized by a vector parameter $\theta \in \Theta$. A Dirichlet process is then used as a non-parametric prior over the parameters of the optimal policies for different tasks. Lazaric and Ghavamzadeh (2010) consider the case of a parameterized space of value functions by considering the space of linear functions spanned by a given set of features, $\mathscr{H} = \{h(x,a) = \sum_{i=1}^{d} \theta_i \varphi_i(x,a)\}$. The vector θ is assumed to be drawn from a multivariate Guassian with parameters ψ drawn from a normal-inverse-Wishart hyper-prior (i.e., $\theta \sim \mathscr{N}(\mu, \Sigma)$ and $\mu, \Sigma \sim \mathscr{N}\text{-}\mathscr{IW}(\psi)$). The inference problem is solved using an EM-like algorithm which takes advantage of the conjugate priors. This approach is further extended to consider the case in which not all the tasks are drawn from the same distribution. In order to cluster tasks into different classes, Lazaric and Ghavamzadeh (2010) place

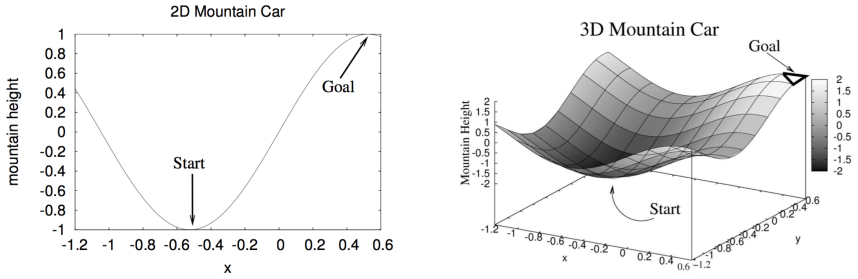

Fig. 6 Transfer from 2D to 3D mountain car (Taylor et al, 2008a).

a Dirichlet process on the top of the hierarchical Bayesian model and the number of classes and assignment of tasks to classes is automatically learned by solving an inference problem using a Gibbs sampling method. Finally, Mehta et al (2008) define the reward function as a linear combination of reward features which are common across tasks, while the weights are specific for each task. The weights are drawn from a distribution Ω and the transfer algorithm compactly stores the optimal value functions of the source tasks exploiting the structure of the reward function and uses them to initialize the solution in the target task.

5 Methods for Transfer from Source to Target Tasks with a Different State-Action Spaces

All the previous settings consider the case where all the tasks share the same *domain* (i.e., they have the same state-action space). In the most general transfer setting the tasks in $\mathcal{M}$ may also differ in terms of number or range of the state-action variables.

5.1 Problem Formulation

Although here we consider the general case in which each task $M \in \mathcal{M}$ is defined as an MDP $\langle S_M, A_M, T_M, R_M \rangle$ and the environment $\mathcal{E}$ is obtained by defining a distribution Ω on $\mathcal{M}$, in practice only the source-to-target transfer setting has been considered. Thus, similarly to Section 3, we consider a task space $\mathcal{M} = \{M_s, M_t\}$, where $M_s = \langle S_s, A_s, T_s, R_s \rangle$ is the source task and $M_t = \langle S_t, A_t, T_t, R_t \rangle$ is the target task on which we want to improve the learning performance. According to the notation introduced in Section 2.1, $\mathcal{K}_s$ and $\mathcal{K}_t$ are now defined on different state-action spaces and the knowledge in $\mathcal{K}_s$ cannot be directly used by the learning algorithm to learn on the target task. Thus, the transfer phase implemented by $\mathcal{A}_{\text{transfer}}$ must return some knowledge $\mathcal{K}_{\text{transfer}}$ compatible with $\mathcal{K}_t$. This objective is usually achieved

following three different approaches: transform $\mathscr{K}_s$ into $\mathscr{K}_t$ through a *hand-coded* mapping, *learn a mapping* from M_s to M_t and transform $\mathscr{K}_s$ into $\mathscr{K}_t$, extract from $\mathscr{K}_s$ some *abstract knowledge* that can be reused to solve M_t. In the next section we still follow the different categories used in the previous sections but we will make explicit which approach to the problem of mapping is used.

Example 5. Let us consider the mountain car domain and the source and target tasks in Figure 6 introduced by Taylor et al (2008a). Although the problem is somehow similar (i.e., an under-powered car has to move from the bottom of the valley to the top of the hill), the two tasks are defined over a different state space and the action space contains a different number of actions. In fact, in the 2D mountain car task the state space is defined by the position and the velocity variables $(x, \dot{x})$ and the action space contains the actions $A = \{\text{Left}, \text{Neutral}, \text{Right}\}$. On the other hand, the 3D task has two additional state variables describing the position in y and its corresponding speed $\dot{y}$ and the action space becomes $A = \{\text{Neutral}, \text{West}, \text{East}, \text{South}, \text{North}\}$. The transfer approaches described so far cannot be applied here because the knowledge $\mathscr{K}_{\text{transfer}}$ they transfer from the source task would not be compatible with the target task. In this case, the transfer algorithm must define a suitable mapping between source and target state and action spaces, and then transfer solutions learned in the 2D mountain car to initialize the learning process in the 3D task. □

5.2 Instance Transfer

Similar to the method introduced by Lazaric et al (2008), Taylor et al (2008a) study the transfer of samples. Unlike Lazaric et al (2008), here only one source task is considered and no specific method for the selection of the samples is implemented. On the other hand, the two tasks do not share the same domain anymore, thus an explicit mapping between their state-action variables is needed. A hand-coded mapping is provided as input to the transfer algorithms which simply applies it to the source samples thus obtaining samples that can be used to learn on the target task. Following the inter-task mapping formalism introduced by Taylor et al (2007a) and used by Taylor et al (2008a), a hand-coded mapping is defined by two functionals χ_S and χ_A. Let the state spaces S_s and S_t be factorized in d_s and d_t state variables respectively (i.e., $S_s = S^1 \times \ldots \times S^{d_s}$ and $S_t = S^1 \times \ldots \times S^{d_t}$) and A_s and A_t be scalar spaces with values $A_s = \{a^1, \ldots, a^{k_s}\}$ and $A_t = \{a^1, \ldots, a^{k_t}\}$. The state mapping maps the index of a source state variable $1 \leq i \leq d_s$ to the index of a state variable $1 \leq j \leq d_t$ in the target state space, that is $\chi_S : \{1, \ldots, d_s\} \to \{1, \ldots, d_t\}$. With an abuse of notation we denote by $\chi_S(s) \in S_t$ the transformation of a state $s \in S_s$ into the state obtained by mapping each variable of s into a target variable according to χ_S. Similarly, the action mapping χ_A maps each source action in A_s to one of the target actions in A_t. As a result, if $\mathscr{K}_s$ is the space of source samples and $K_s \in \mathscr{K}_s$ is one specific realization of N_s samples, for any samples $\langle s_n, a_n, s'_n, r_n \rangle \in K_s$ the transfer algorithm returns new target samples $\langle \chi_S(s_n), \chi_A(a_n), \chi_S(s'_n), r_n \rangle$. While Lazaric et al (2008) define an

algorithm where the transferred samples are used in a batch model-free RL algorithm, Taylor et al (2008a) study how the model-based algorithm Fitted R-max can benefit from samples coming from the source task when transformed according to a hand-coded mapping from the source to the target task. In particular, the method is shown to be effective in the generalized mountain car problem in Figure 6.

5.3 Representation Transfer

As reviewed in the previous sections, many transfer approaches develop options that can be effectively reused in the target task. In this case, the main problem is that options learned on the source task are defined as a mapping from S_s to A_s and they cannot be used in a target task with different state-action variables. A number of transfer algorithms deal with this problem by considering abstract options that can be reused in different tasks. Ravindran and Barto (2003); Soni and Singh (2006) use the *homomorphism* framework to map tasks to a common abstract level. For instance, let us consider all the navigation problems in an empty squared room. In this case, it is possible to define one common abstract MDP and obtain any specific MDP by simply using operators such as translation, scaling, and rotation. In order to deal with this scenario, Ravindran and Barto (2003) introduce the concept of relativized options. Unlike traditional options, relativized options are defined on the abstract MDP, without an absolute frame of reference, and their policy is then transformed according to the specific target task at hand. In particular, a set of possible transformations is provided and the transfer phase needs to identify the most suitable transformation of the relativized options depending on the current target task. The problem is casted as a Bayesian parameter estimation problem and the transformation which makes the sequence of states observed by following the option more likely is selected. Konidaris and Barto (2007) define options at a higher level of abstraction and they can be used in the target task without any explicit mapping or transformation. In fact, portable options are defined in a non-Markovian *agent space* which depends on the characteristics of the agent and remains fixed across tasks. This way, even when tasks are defined on different state-action spaces, portable options can be reused to speed-up learning in any target task at hand. Finally, Torrey et al (2006) proposed an algorithm in which a set of skills is first identified using inductive logic programming and then reused in the target task by using a hand-coded mapping from source to target.

5.4 Parameter Transfer

While in the setting considered in Section 3, the transfer of initial solutions (e.g., the optimal source policy) from the source to the target task is trivial, in this case

the crucial aspect in making transfer effective is to find a suitable mapping from the source state-action space $S_s \times A_s$ to the target state-action space $S_t \times A_t$.

Most of the algorithms reviewed in the following consider hand-coded mappings and investigate how the transfer of different sources of knowledge (e.g., policies, value functions) influence the performance on the target task. The transformation through a hand-coded mapping and the transfer of the source value function to initialize the learning in the target task has been first introduced by Taylor and Stone (2005) and Taylor et al (2005) and its impact has been study in a number of challenging problems such as the simulated keep-away problem (Stone et al, 2005). Banerjee and Stone (2007) also consider the transfer of value functions in the context of general games where different games can be represented by a common abstract structure. Torrey et al (2005) learn the Q-table in the target task by reusing *advices* (i.e., actions with higher Q-values in the source task) which are mapped to the target task through a hand-coded mapping. While the previous approaches assume that in both source and target task the same solution representation is used (e.g., a tabular approach), Taylor and Stone (2007) consider the problem of mapping a solution (i.e., a value function or a policy) to another solution when either the approximation architecture (e.g., CMAC and neural networks) or the learning algorithm itself (e.g., value-based and policy search methods) changes between source and target tasks. Using similar mappings as for the state-action mapping, they show that transfer is still possible and it is still beneficial in improving the performance on the target task. Finally, Taylor et al (2007b) study the transfer of the source policy where a hand-coded mapping is used to transform the source policy into a valid policy for the target task and a policy search algorithm is then used to refine it.

Unlike the previous approaches, some transfer methods automatically identify the most suitable mapping between source and target tasks. Taylor et al (2008b) introduce the MASTER algorithm. The objective is to identify among all the possible mappings from source to target state variables, the one which guarantees the best prediction of the dynamics of the environment. The algorithm receives as input a relatively large number of samples from the source task and few target samples. Let X be the set of all the possible mappings between S_s and S_t, $K_s \in \mathscr{K}_s$ and $K_t \in \mathscr{K}_t$ be specific realization of source and target samples. From K_t the algorithm first computes an approximation of the target dynamics $\hat{T}_t$. Each sample $\langle s_n, a_n, s'_n, r_n \rangle$ in K_s is then transformed with one of the possible mappings $\chi_S \in X$ and the state $\chi_S(s'_n)$ is compared to the state predicted by $\hat{T}_t(\chi_S(s_n,), a)$. The mapping χ_S which is the most accurate in predicting the transitions of the samples in K_s is then used to transfer the solution of the source task. Talvitie and Singh (2007) first learn a policy for the source task, and a target policy is then obtained by mapping the source policy according to each of the possible mappings from the source to the target state variables. The problem of selecting the most appropriate mapping is then translated into the problem of evaluating the best policy among the target policies. The problem is then solved using an expert-like algorithm (Cesa-Bianchi and Lugosi, 2006).

6 Conclusions and Open Questions

In this chapter we defined a general framework for the transfer learning problem in the reinforcement learning paradigm, we proposed a classification of the different approaches to the problem, and we reviewed the main algorithms available in the literature. Although many algorithms have been already proposed, the problem of transfer in RL is far from being solved. In the following we single out a few open questions that are relevant to the advancement of the research on this topic. We refer the reader to the survey by Taylor and Stone (2009) for other possible lines of research in transfer in RL.

Theoretical analysis of transfer algorithms. Although experimental results support the idea that RL algorithms can benefit from transfer from related tasks, no transfer algorithm for RL has strong theoretical guarantees. Recent research in transfer and multi-task learning in the supervised learning paradigm achieved interesting theoretical results identifying the conditions under which transfer approaches are expected to improve the performance over single-task learning. Crammer et al (2008) study the performance of learning reusing samples coming from different classification tasks and they prove that when the sample distributions of the source tasks do not differ too much compared to the target distribution, then the transfer approach performs better than just using the target samples. Baxter (2000) studies the problem of learning the most suitable set of hypotheses for a given set of tasks. In particular, he shows that, as the number of source tasks increases, the transfer algorithm manages to identify a hypothesis set which is likely to contain *good* hypotheses for all the tasks in $\mathscr{M}$. Ben-David and Schuller-Borbely (2008) consider the problem of learning the best hypothesis set in the context of multi-task learning where the objective is not to generalize on new tasks but to achieve a better average performance in the source tasks. At the same time, novel theoretical results are now available for a number of popular RL algorithms such as fitted value iteration (Munos and Szepesvári, 2008), LSTD (Farahmand et al, 2008; Lazaric et al, 2010), and Bellman residual minimization (Antos et al, 2008; Maillard et al, 2010). An interesting line of research is to take advantage of theoretical results of transfer algorithms in the supervised learning setting and of RL algorithms in the single-task case to develop new RL transfer algorithms which provably improve the performance over single-task learning.

Transfer learning for exploration. The objective of learning speed improvement (see Section 2.2) is often achieved by a better use of the samples at hand (e.g., by changing the hypothesis set) rather than by the collection of more *informative* samples. This problem is strictly related to the exploration-exploitation dilemma where the objective is to trade-off between the exploration of different strategies and the exploitation of the best strategy so far. Recent works by Bartlett and Tewari (2009); Jaksch et al (2010) studied optimal exploration strategies for single-task learning. Although most of the option-based transfer methods implicitly bias the exploration strategy, the problem of how the exploration on one task should be adapted on the basis of the knowledge of previous related tasks is a problem which received little attention so far.

Concept drift and continual learning. One of the main assumptions of transfer learning is that a clear distinction between the tasks in $\mathscr{M}$ is possible. Nonetheless, in many interesting applications there is no sharp division between source and target tasks while it is rather the task itself that changes in time. This problem, also known as *concept drift*, is also strictly related to the continual learning and lifelong learning paradigm (Silver and Poirier, 2007) in which, as the learning agent autonomously discovers new regions of a non-stationary environment, it also increases its capability to solve tasks defined on that environment. Although tools coming from transfer learning probably could be reused also in this setting, novel approaches are needed to deal with the non-stationarity of the environment and to track the changes in the task at hand.

References

Antos A, Szepesvári C, Munos R (2008) Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. Machine Learning Journal 71:89–129

Argyriou A, Evgeniou T, Pontil M (2008) Convex multi-task feature learning. Machine Learning Journal 73(3):243–272

Asadi M, Huber M (2007) Effective control knowledge transfer through learning skill and representation hierarchies. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07), pp 2054–2059

Banerjee B, Stone P (2007) General game learning using knowledge transfer. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07), pp 672–677

Bartlett PL, Tewari A (2009) Regal: a regularization based algorithm for reinforcement learning in weakly communicating mdps. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI-09), AUAI Press, Arlington, Virginia, United States, pp 35–42

Baxter J (2000) A model of inductive bias learning. Journal of Artificial Intelligence Research 12:149–198

Ben-David S, Schuller-Borbely R (2008) A notion of task relatedness yiealding provable multiple-task learning guarantees. Machine Learning Journal 73(3):273–287

Bernstein DS (1999) Reusing old policies to accelerate learning on new mdps. Tech. rep., University of Massachusetts, Amherst, MA, USA

Bonarini A, Lazaric A, Restelli M (2006) Incremental skill acquisition for self-motivated learning animats. In: Proceedings of the Ninth International Conference on Simulation of Adaptive Behavior (SAB-06), Lecture Notes in Artificial Intelligence, vol 4095, Springer Verlag, Berlin, pp 357–368

Cesa-Bianchi N, Lugosi G (2006) Prediction, Learning, and Games. Cambridge University Press

Crammer K, Kearns M, Wortman J (2008) Learning from multiple sources. Journal of Machine Learning Research 9:1757–1774

Drummond C (2002) Accelerating reinforcement learning by composing solutions of automatically identified subtasks. Journal of Artificial Intelligence Research 16:59–104

Engel Y, Mannor S, Meir R (2005) Reinforcement learning with Gaussian processes. In: Proceedings of the 22nd International Conference on Machine Learning (ICML-05), pp 201–208

Ernst D, Geurts P, Wehenkel L (2005) Tree-based batch mode reinforcement learning. Journal of Machine Learning Research 6:503–556

Farahmand AM, Ghavamzadeh M, Szepesvári C, Mannor S (2008) Regularized policy iteration. In: Proceedings of the Twenty-Second Annual Conference on Advances in Neural Information Processing Systems (NIPS-2008), pp 441–448

Fawcett T, Callan J, Matheus C, Michalski R, Pazzani M, Rendell L, Sutton R (eds) (1994) Constructive Induction Workshop at the Eleventh International Conference on Machine Learning

Ferguson K, Mahadevan S (2006) Proto-transfer learning in markov decision processes using spectral methods. In: Workshop on Structural Knowledge Transfer for Machine Learning at the Twenty-Third International Conference on Machine Learning

Ferns N, Panangaden P, Precup D (2004) Metrics for finite markov decision processes. In: Proceedings of the 20th conference on Uncertainty in Artificial Intelligence (UAI-2004), pp 162–169

Ferrante E, Lazaric A, Restelli M (2008) Transfer of task representation in reinforcement learning using policy-based proto-value functions. In: Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-08), pp 1329–1332

Foster DJ, Dayan P (2002) Structure in the space of value functions. Machine Learning Journal 49(2-3):325–346

Gentner D, Loewenstein J, Thompson L (2003) Learning and transfer: A general role for analogical encoding. Journal of Educational Psychology 95(2):393–408

Gick ML, Holyoak KJ (1983) Schema induction and analogical transfer. Cognitive Psychology 15:1–38

Hauskrecht M (1998) Planning with macro-actions: Effect of initial value function estimate on convergence rate of value iteration. Tech. rep., Department of Computer Science, University of Pittsburgh

Hengst B (2003) Discovering hierarchy in reinforcement learning. PhD thesis, University of New South Wales

Jaksch T, Ortner R, Auer P (2010) Near-optimal regret bounds for reinforcement learning. Journal of Machine Learning Research 11:1563–1600

Kalmar Z, Szepesvari C (1999) An evaluation criterion for macro-learning and some results. Tech. Rep. TR-99-01, Mindmaker Ltd.

Konidaris G, Barto A (2006) Autonomous shaping: knowledge transfer in reinforcement learning. In: Proceedings of the Twenty-Third International Conference on Machine Learning (ICML-06), pp 489–496

Konidaris G, Barto AG (2007) Building portable options: Skill transfer in reinforcement learning. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07), pp 895–900

Langley P (2006) Transfer of knowledge in cognitive systems. Talk, workshop on Structural Knowledge Transfer for Machine Learning at the Twenty-Third International Conference on Machine Learning

Lazaric A (2008) Knowledge transfer in reinforcement learning. PhD thesis, Poltecnico di Milano

Lazaric A, Ghavamzadeh M (2010) Bayesian multi-task reinforcement learning. In: Proceedings of the Twenty-Seventh International Conference on Machine Learning (ICML-2010), submitted to the

Lazaric A, Restelli M, Bonarini A (2008) Transfer of samples in batch reinforcement learning. In: Proceedings of the Twenty-Fifth Annual International Conference on Machine Learning (ICML-2008), pp 544–551

Lazaric A, Ghavamzadeh M, Munos R (2010) Finite-sample analysis of lstd. In: Proceedings of the Twenty-Seventh International Conference on Machine Learning (ICML-2010)

Li H, Liao X, Carin L (2009) Multi-task reinforcement learning in partially observable stochastic environments. Journal of Machine Learning Research 10:1131–1186

Madden MG, Howley T (2004) Transfer of experience between reinforcement learning environments with progressive difficulty. Artificial Intelligence Review 21(3-4):375–398

Mahadevan S, Maggioni M (2007) Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. Journal of Machine Learning Research 38:2169–2231

Maillard OA, Lazaric A, Ghavamzadeh M, Munos R (2010) Finite-sample analysis of bellman residual minimization. In: Proceedings of the Second Asian Conference on Machine Learning (ACML-2010)

McGovern A, Barto AG (2001) Automatic discovery of subgoals in reinforcement learning using diverse density. In: Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)

Mehta N, Natarajan S, Tadepalli P, Fern A (2008) Transfer in variable-reward hierarchical reinforcement learning. Machine Learning Journal 73(3):289–312

Menache I, Mannor S, Shimkin N (2002) Q-cut - dynamic discovery of sub-goals in reinforcement learning. In: Proceedings of the Thirteen European Conference on Machine Learning, pp 295–306

Munos R, Szepesvári C (2008) Finite time bounds for fitted value iteration. Journal of Machine Learning Research 9:815–857

Pan SJ, Yang Q (2010) A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22(22):1345–1359

Perkins DN, Salomon G, Press P (1992) Transfer of learning. In: International Encyclopedia of Education, Pergamon Press

Perkins TJ, Precup D (1999) Using options for knowledge transfer in reinforcement learning. Tech. rep., University of Massachusetts, Amherst, MA, USA

Phillips C (2006) Knowledge transfer in markov decision processes. http://www.cs.mcgill.ca/ martin/usrs/phillips.pdf, McGill School of Computer Science

Ravindran B, Barto AG (2003) Relativized options: Choosing the right transformation. In: Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003), pp 608–615

Sherstov AA, Stone P (2005) Improving action selection in MDP's via knowledge transfer. In: Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)

Silver D (2000) Selective transfer of neural network task knowledge. PhD thesis, University of Western Ontario

Silver DL, Poirier R (2007) Requirements for machine lifelong learning. In: IWINAC (1), pp 313–319

Simsek O, Wolfe AP, Barto AG (2005) Identifying useful subgoals in reinforcement learning by local graph partitioning. In: Proceedings of the Twenty-Second International Conference of Machine Learning (ICML 2005)

Singh S, Barto A, Chentanez N (2004) Intrinsically motivated reinforcement learning. In: Proceedings of the Eighteenth Annual Conference on Neural Information Processing Systems (NIPS-2004)

Soni V, Singh SP (2006) Using homomorphisms to transfer options across continuous reinforcement learning domains. In: Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06)

Stone P, Sutton RS, Kuhlmann G (2005) Reinforcement learning for RoboCup-soccer keepaway. Adaptive Behavior 13(3):165–188

Sunmola FT, Wyatt JL (2006) Model transfer for markov decision tasks via parameter matching. In: Proceedings of the 25th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG 2006)

Sutton RS, Barto AG (1998) Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA

Sutton RS, Precup D, Singh S (1999) Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. Artificial Intelligence 112:181–211

Talvitie E, Singh S (2007) An experts algorithm for transfer learning. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07), pp 1065–1070

Tanaka F, Yamamura M (2003) Multitask reinforcement learning on the distribution of mdps. In: IEEE International Symposium on Computational Intelligence in Robotics and Automation, vol 3, pp 1108–1113

Taylor ME, Stone P (2005) Behavior transfer for value-function-based reinforcement learning. In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-05), pp 53–59

Taylor ME, Stone P (2007) Representation transfer for reinforcement learning. In: AAAI 2007 Fall Symposium on Computational Approaches to Representation Change during Learning and Development

Taylor ME, Stone P (2009) Transfer learning for reinforcement learning domains: A survey. Journal of Machine Learning Research 10(1):1633–1685

Taylor ME, Stone P, Liu Y (2005) Value functions for RL-based behavior transfer: A comparative study. In: Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)

Taylor ME, Stone P, Liu Y (2007a) Transfer learning via inter-task mappings for temporal difference learning. Journal of Machine Learning Research 8:2125–2167

Taylor ME, Whiteson S, Stone P (2007b) Transfer via inter-task mappings in policy search reinforcement learning. In: Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-07)

Taylor ME, Jong NK, Stone P (2008a) Transferring instances for model-based reinforcement learning. In: Proceedings of the European Conference on Machine Learning (ECML-08), pp 488–505

Taylor ME, Kuhlmann G, Stone P (2008b) Autonomous transfer for reinforcement learning. In: Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-08), pp 283–290

Thorndike EL, Woodworth RS (1901) The influence of improvement in one mental function upon the efficiency of other functions. Psychological Review 8

Torrey L, Walker T, Shavlik JW, Maclin R (2005) Using advice to transfer knowledge acquired in one reinforcement learning task to another. In: Proceedings of the European Conference on Machine Learning (ECML-05), pp 412–424

Torrey L, Shavlik JW, Walker T, Maclin R (2006) Skill acquisition via transfer learning and advice taking. In: Proceedings of the European Conference on Machine Learning (ECML-06), pp 425–436

Utgoff P (1986) Shift of bias for inductive concept learning. Machine Learning 2:163–190

Walsh TJ, Li L, Littman ML (2006) Transferring state abstractions between mdps. In: ICML Workshop on Structural Knowledge Transfer for Machine Learning

Watkins C, Dayan P (1992) Q-learning. Machine Learning 8:279–292

Wilson A, Fern A, Ray S, Tadepalli P (2007) Multi-task reinforcement learning: a hierarchical bayesian approach. In: Proceedings of the Twenty-Forth International Conference on Machine learning (ICML-07), pp 1015–1022