Künstl Intell (2014) 28:7–14
DOI 10.1007/s13218-013-0286-8

DISCUSSION

Automated Transfer for Reinforcement Learning Tasks

Haitham Bou Ammar · Siqi Chen · Karl Tuyls · Gerhard Weiss

Published online: 9 January 2014

Abstract Reinforcement learning applications are hampered by the tabula rasa approach taken by existing techniques. Transfer for reinforcement learning tackles this problem by enabling the reuse of previously learned behaviours. To be fully autonomous a transfer agent has to: (1) automatically choose a relevant source task(s) for a given target, (2) learn about the relation between the tasks, and (3) effectively and efficiently transfer between tasks. Currently, most transfer frameworks require substantial human intervention in at least one of the previous three steps. This discussion paper aims at: (1) positioning various knowledge re-use algorithms as forms of transfer, and (2) arguing the validity and possibility of autonomous transfer by detailing potential solutions to the above three steps.

Keywords Transfer learning · Reinforcement learning · Markov decision processes · Inter-task mappings

H. Bou Ammar (✉)
Computer and Information Science Department,
University of Pennsylvania, Philadelphia,
PA 19104-6309, USA
e-mail: haitham.bouammar71@googlemail.com

S. Chen · G. Weiss
Department of Knowledge Engineering,
Maastricht University, Maastricht, Netherlands
e-mail: siqi.chen@maastrichtuniversity.nl

G. Weiss
e-mail: gerhard.weiss@maastrichtuniversity.nl

K. Tuyls
Department of Computer Science,
University of Liverpool, Liverpool, UK
e-mail: k.tuyls@liverpool.ac.uk

1 Introduction

In reinforcement learning (RL), an agent lives in an environment which it can perceive through sensory signals and affect by taking actions. To asses the behaviour of a certain action performed by an agent, a reward signal is used. This reward will punish the agent in case it has performed a "bad" action, and reward it for a "good" one. The agent then learns to maximise its total positive signal.

RL is typically formalised using Markov Decision Processes (MDPs). An MDP is a tuple of a state space, $\mathcal{S}$, an action space, $\mathcal{A}$, a reward function, $\mathcal{R}$, a transition probability function, $\mathcal{T}_{s,a}$, and a discount factor γ. $\mathcal{S}$ represents all the possible states of an environment, while $\mathcal{A}$ is the set of all possible actions the agent is allowed to execute. The reward function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$, quantifies the usefulness of the taken action on transitioning to a new state s'. This transition is dictated by the transition probability function $\mathcal{T}_{s,a}$. More specifically, given a current state $s \in \mathcal{S}$ and applying an action $a \in \mathcal{A}$, the transition probability, $\mathcal{T}_{s,a}$ determines the successor state $s' \sim \mathcal{T}_{s,a}$. The goal is then to maximise the total discounted rewards attained from the environment.

RL has become a popular framework for autonomous behaviour generation from limited feedback [4], but RL methods learn tabula rasa. In other words, agents when faced with a new task, start learning from scratch without assuming any prior knowledge. Due to such assumptions, RL agents learn slowly in large or complex environments.

Knowledge reuse algorithms, including but not limited to, reward shaping [10], apprenticeship learning [1], human trainer feedback [8], learning from demonstration [3], and inter-task transfer learning [2, 13] have been proposed to remedy these computational problems. Such techniques try

to improve learning in new tasks by incorporating additional knowledge from an external source. These vary depending on the assumptions imposed on the source and target tasks. In human trainer feedback, for instance, the human, assumed to be the expert, provides additional knowledge for the agent in terms of either action selection suggestions, or online feedback. These techniques, as discussed in Sect. 2, can be considered under suitable assumptions as special cases of the more general transfer framework. Namely, each of the above methods can be regarded as inter-task mapping transfers with varying degrees of autonomy.

Inter-task mapping transfer can be regarded as the most general transfer scenario since all MDPs' constituents are allowed to vary, mamely:

1. *Domain differences:* The state and action spaces of the two tasks not only can vary in size but also in dimensionality, and
2. *Task differences:* The transition probabilities, and the reward functions are also allowed to vary.

Different intertask mapping transfer techniques with varying degrees of autonomy have been proposed. Although successful, non of these are fully autonomous. Human intervention is needed to guide the target agent in either: (1) selecting the most relevant source task, (2) learning about the relation between the source and the target, and/or (3) transferring the source knowledge effectively to the target task.

Therefore, if transfer is to be fully automated an agent should be capable of performing each of the three previous steps autonomously [12]. More specifically, given a target task and a set of source tasks, a transfer agent should be able to automatically: (1) choose the most relevant source task, (2) infer about the relation between the two tasks, and (3) effectively transfer the source knowledge.

This paper, firstly, introduces a framework that allows the positioning of knowledge reuse algorithms under the context of transfer learning and secondly, discusses the possibility of creating fully autonomous transfer agents.

2 Reinforcement and Transfer Framework

This section introduces a framework that can be used to describe various knowledge reuse algorithms. Reinforcement learners are framed as mappings from knowledge to hypothesis sets. This constitutes the basis for the transfer framework that is then detailed.

2.1 Reinforcement Learners as Mappings

Reinforcement learning agents can be thought of as black-box mappings from a knowledge set, $\mathfrak{K}$, to hypotheses set $\mathfrak{H}$. The knowledge set, $\mathfrak{K}$, can be either given beforehand (e.g., offline reinforcement learning) or acquired through environmental interactions (e.g., online reinforcement learning). An agent/learner, $\mathfrak{L}$, maps the available knowledge to the hypothesis space (i.e., $\mathfrak{L}\ \mathfrak{K} \rightarrow \mathfrak{H}$). The definition of the hypothesis space, $\mathfrak{H}$, is mostly application oriented and depends on the reinforcement learning algorithm used[1]. For instance, in fitted-Q iteration (FQI), such a space is defined as a linear combination of basis functions used to approximate the Q-function. Therefore, the hypothesis space, is spanned by $\mathfrak{B} = \{\phi^{(i)}\}_{i=}^{\mathfrak{f}}$, where $\phi^{(i)} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. In words, $\phi^{(i)}$ is a function mapping states and actions to real numbers[2]. A hypothesis $h(\cdot,\cdot) \in \mathfrak{H}$, is then a linear combination of the basis functions constituting $\mathfrak{B}$. Namely $h(\cdot,\cdot) = \sum_{i=1}^{k} \gamma^{(i)}\phi^{(i)}(\cdot,\cdot)$. Given m samples, the goal of an FQI learner is to map the available knowledge $\mathfrak{K} = \left\{(\mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{R})^{\mathrm{m}}, \mathfrak{B}\right\}$ to the hypothesis space under certain criteria[3].

Various modifications of these definitions are definitely possible. For example, in some cases the basis functions can be defined over successor states too, or the hypothesis might be a nonlinear (or even nonparametric) approximation of states and actions. These can be included into the framework and thus RL agents can be seen as nothing but knowledge to hypothesis mappings.

2.2 Transfer and Knowledge Reuse Learners

Opposed to traditional reinforcement learning, in transfer, additional knowledge is available to the agent to facilitate learning. On a high level, two types of knowledge can be differentiated. The first is the so-called *source* knowledge, while the second is knowledge in the *target*. For instance, to learn the relation between source and target tasks, some level of knowledge about the target has to be available (e.g., target transitions might be required as explained in Sect. 4.1).

The source knowledge potentially resides in different realms to that of the target. This is, for example, the case in human trainer feedback, where the external knowledge is acquired from humans which are considered to be the experts. Such knowledge has to be *correctly* mapped before

[1] In policy iteration algorithms, for example, the policy space can be defined as the space of all possible policies that can be learnt. In other words, this space can be defined by a combination of basis functions and parameterisations spanning different policies.

[2] Such a setting is typical in continuous reinforcement learning. The reasons relate to: (1) Q-function, and (2) state and action space representations.

[3] A typical criterion used is to maximise the expected value of the total discounted pay-off signal.

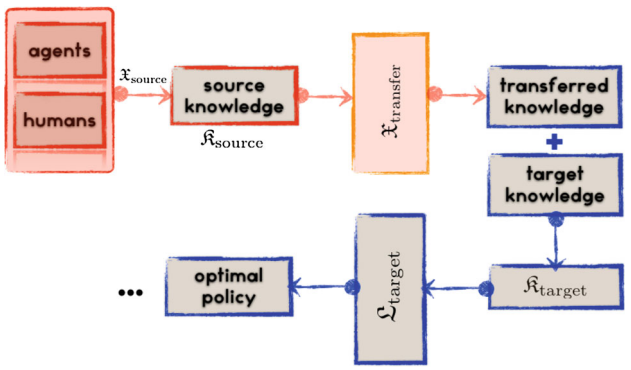

Fig. 1 The overall transfer learning framework. The source knowledge, $\mathfrak{K}_{\text{source}}$, is passed through the intertask mapping $\mathfrak{X}_{\text{target}}$ to produce a part of the target knowledge $\mathfrak{K}_{\text{target}}$. This target knowledge is then passed to the target learner $\mathfrak{L}_{\text{target}}$ to produce an optimal target policy π_{target}

being used in a target. Such form of mappings are typically referred to as *inter-task* mappings, $\mathfrak{X}_{\text{transfer}}$.

Figure 1 shows the overall transfer framework that applies to various knowledge reuse algorithms. Firstly, source knowledge is available from other agents (in different source tasks) or from humans. If this knowledge is available in different formats it needs to be mapped using $\mathfrak{X}_{\text{source}}$ to an acceptable form before being used by the learning framework. For instance, if human advice was adopted, human knowledge has to be converted to action advices for example. $\mathfrak{K}_{\text{source}}$ is then passed through an intertask mapping[4] , $\mathfrak{X}_{\text{transfer}}$ $\mathfrak{K}_{\text{source}} \times \mathfrak{K}_{\text{target}} \rightarrow \mathfrak{K}_{\text{target}}$, with the goal of producing a part of the target knowledge $\mathfrak{K}_{\text{target}}$. Additional target knowledge plus that of transfer, together constitute $\mathfrak{K}_{\text{target}}$. This is then used by $\mathfrak{L}_{\text{target}}$ $\mathfrak{K}_{\text{target}} \rightarrow \mathfrak{H}_{\text{target}}$ to generate the optimal target mapping (i.e., π_{target}).

Using this framework, knowledge reuse algorithms can be framed in the context of transfer with varying assumptions. In learning from demonstration, for instance, the source knowledge is available within the same target domain (i.e., state and action spaces) in the form of trajectories, near-optimal policies, and/or parameters, etc. For reward shaping, such a source knowledge can either be attained from humans or learned from other tasks to design a modification to the target's reward function.

Having introduced the transfer framework, next autonomous inter-task mapping transfer is pursued. The derivations and discussions presented next correspond to the most general case.

[4] $\mathfrak{X}_{\text{transfer}}$ can either be: (1) hand-coded (see [13]), or (2) learned through source and target samples [2].

3 Problem Formulation

The problem of creating autonomous transfer agents for RL can be split into three sub-questions. The first is how to relate and infer about two given tasks (i.e., a source and a target). Having this relation, the second question is how to exploit this learnt relation in order to successfully and effectively conduct transfer. The final challenge to be tackled is how to construct a framework in which an agent is automatically capable of choosing (a) relevant source task(s) to a given target. According to [12] automated transfer is thus achieved.

Rather than focusing on discrete reinforcement learning, the aim in this paper is to tackle a more generic RL framework. Namely, continuous states discrete actions RL is in focus. This setting covers a broader scope of applications compared to discrete RL.

To tackle each of the above subproblems, we assume two MDPs, $\mathcal{M}_S = \langle \mathcal{S}_S, \mathcal{A}_S, \mathcal{R}_S, \mathcal{T}_S, \gamma_S \rangle$, and $\mathcal{M}_T = \langle \mathcal{S}_T, \mathcal{A}_T, \mathcal{R}_T, \mathcal{T}_T, \gamma_T \rangle$. Furthermore, the following is also available:

1. *Source knowledge:* In the source task, a set of m random samples, and k basis functions, $\mathfrak{B}_{\text{source}}$, are available. Namely:

$$\mathfrak{K}_{\text{source}} = \left\{ (\mathcal{S}_S \times \mathcal{A}_S \times \mathcal{S}_S \times \mathcal{R}_S)^m, \mathfrak{B}_{\text{source}} \right\}$$

2. *Target knowledge:* In the target, a set of $n << m$ random samples, and l basis function, $\mathfrak{B}_{\text{target}}$ are available:

$$\mathfrak{K}_{\text{target}} = \left\{ (\mathcal{S}_T \times \mathcal{A}_T \times \mathcal{S}_T \times \mathcal{R}_T)^n, \mathfrak{B}_{\text{target}} \right\}$$

Please note that not only the transition probabilities and reward functions may vary between the tasks, but also the dimensionality and the size of each of the state and action spaces. Using the above notation the three subproblems leading to autonomous transfer are:

1. *Inter-task mapping:* Given $\mathfrak{K}_{\text{source}}$ and $\mathfrak{K}_{\text{target}}$, learn $\mathfrak{X}_{\text{transfer}}$ $\mathfrak{K}_{\text{source}} \rightarrow \mathfrak{K}_{\text{target}}$. The reasons behind adding successor states to the definition of the intertask mapping, is that if an algorithm is to be able to analyse and reason about the relation between two tasks, it has to be able to have some information about the relation between the transition models of the two tasks. Learning such a mapping is a challenge as it is almost impossible for a human to relate source and target triplets (i.e., state-action-successor state) manually, especially if the tasks had different state and/or action space dimensionality.
2. *Effective transfer:* Given an inter-task mapping $\mathfrak{X}_{\text{transfer}}$, source knowledge $\mathfrak{K}_{\text{source}}$, and additional target knowledge learn an optimal policy $\pi^{\star}_{\text{target}}$.

3. *Source task choice:* For automatic source task selection, a measure quantifying the similarity as well as transfer performance between different tasks is required. Precisely, given the source and target task knowledge, $\mathfrak{K}_{\text{source}}$ and $\mathfrak{K}_{\text{target}}$, respectively, learn a measure $\mathfrak{d}\left(\mathfrak{K}_{\text{source}}, \mathfrak{K}_{\text{target}}\right)$.

4 Possible Solutions

In this section, details of possible solutions for each of the above questions are presented. Firstly, details on automatically learning an intertask mapping (i.e., the relation between the source and target tasks) are described. Secondly, two potential solutions for how to exploit this knowledge to effectively transfer is explained. Thirdly, a potential future research direction to solve the last challenge is introduced.

4.1 Automated Intertask Mapping Transfer

The goal of the inter-task mapping $\mathfrak{X}_{\text{transfer}}$ is to relate source and target knowledge $\mathfrak{K}_{\text{source}}$ and $\mathfrak{K}_{\text{target}}$, respectively. Depending on the available knowledge, an inter-task mapping relates source and target state-action spaces, reward functions and/or transition probabilities.

Learning an intertask mapping between two tasks is one of the challenging problems for the general transfer learning case. Different inter-task mapping based transfer algorithms with varying degrees of autonomy have been proposed [13]. Most either assume that the mapping is given by a designer, or require an exponential complexity to find a relevant one.

In [2] a fully automated technique to learn the intertask mapping is proposed. This method does not impose any restrictive assumptions on the variation between the source and target task MDPs. The state, and action spaces, transition probabilities, reward functions, and the discount factors can vary. The problem definition targeted in [2] can be stated as follows:

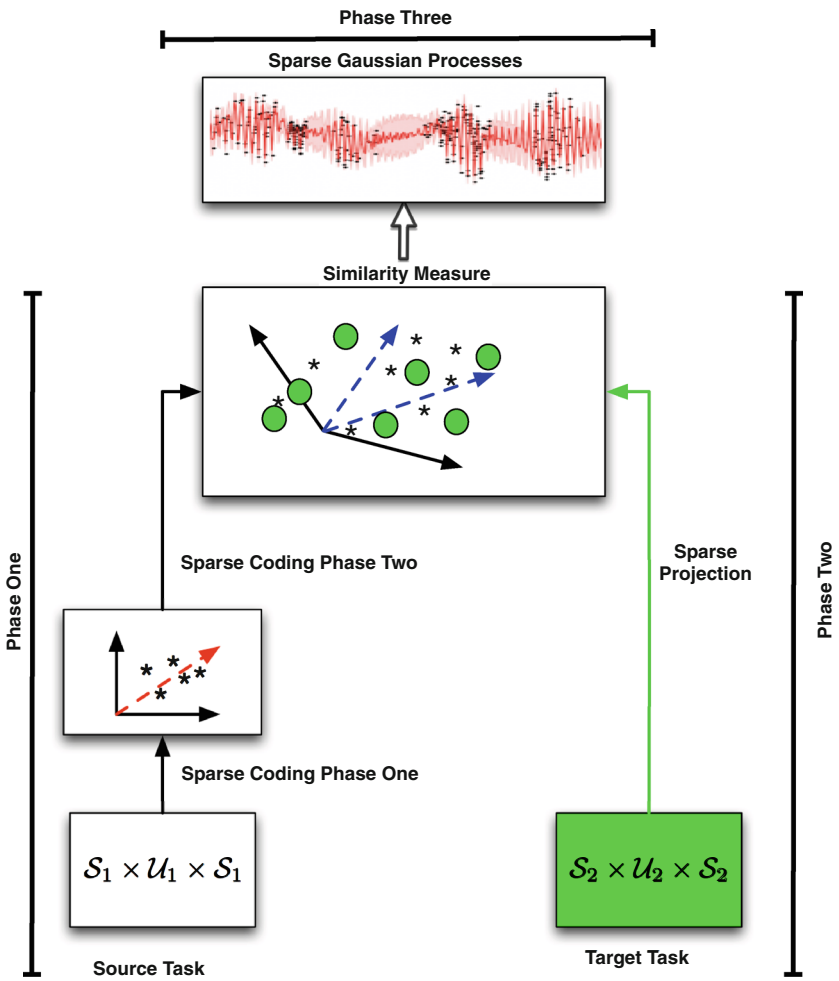

Fig. 2 A high level schematic of the overall approach in [2]. It consists of three major phases. In the first, high level features are detected in the source task MDP. In the second, samples from the target task are projected to that space. Finally, in the last phase (i.e., Phase Three), sparse Gaussian processes are used in order to learn the intertask mapping

Inter-Task Mapping Learner: Given $\mathfrak{K}_{\text{source}} = \{(\mathcal{S}_S \times \mathcal{A}_S \times \mathcal{S}_S)^m, \mathfrak{B}_{\text{source}}\}$ and $\mathfrak{K}_{\text{target}} = \{(\mathcal{S}_T \times \mathcal{A}_T \times \mathcal{S}_T)^n, \mathfrak{B}_{\text{target}}\}$, with $n \ll m$, learn $\mathfrak{X}_{\text{transfer}} : \mathfrak{K}_{\text{soruce}} \rightarrow \mathfrak{K}_{\text{target}}$.

The overall procedure is shown in Fig. 2. The easiest way to approach the problem of learning the intertask mapping is using supervised learning. However, according to the previous definitions, $\mathfrak{X}_{\text{target}}$ maps state-action-successor state triplets in the source to these in the target. Therefore, any supervised learning algorithm will require a data set that has source transitions as inputs and target transitions as outputs. Unfortunately, it is nearly impossible for a human to manually determine which transitions in the source correspond to which in the target. Thus, this problem has also to be approached automatically. The simplest way to determine correspondence is to use a distance measure and search for the transition in the target that is closest to that in the source. However, these transitions might potentially belong to different dimensions. Consequently, before seeking any correspondence, the dimensions of these transitions should be unified.

To unify the dimensions, sparse coding (SC) [9] is used. SC is an unsupervised feature extraction algorithm that tries to discover succinct representations unanticipated in the original set. Given two data sets $\mathcal{D}_S = \{\langle s_S^{(i)}, a_S^{(i)}, s_S^{(i)\prime}\rangle\}_{i=1}^m$ and $\mathcal{D}_T = \{\langle s_T^{(j)}, u_T^{(j)}, s_T^{(j)\prime}\rangle\}_{j=1}^n$, where m and n represent the number of transitions in the source and the target[5], the first step makes use of sparse coding to unify the dimensions of the source to the target. This is, denoted by "Sparse Coding Phase One" in Fig. 2, is followed. Namely, the transitions of the source are sparse coded to attain the same number of dimensions as these of the target. Mainly, the idea is to discover new features in the state-action spaces not anticipated by original variables. However, the number of these new dimensions is limited to be the same as these of the target task.

Essentially, at this stage any transfer learning algorithm can be adopted. However, this new dimensional space might not be informative enough to conduct transfer. To ensure that transfer is performed in a highly informative space, another step of sparse coding is performed to discover "richer" features in the source ("Sparse Coding Phase Two" in Fig. 2). Here the number of new dimensions can be set to a high value. Interestingly, due to the sparsity condition "unneeded" bases will end up attaining close to zero activations and therefore will not contribute in the transfer procedure.

This new space represents informative features in the source. However, it does not relate yet to the target state and action spaces. Therefore, to use the similarity measure to determine the data set needed to approximate the intertask mapping, target triplets need to be projected to this new space. Projection means finding activations in the new bases that represent the original target transitions. This is performed using sparse projection as shown by "Phase Two" of Fig. 2.

Now, the similarity measure to correspond the source and target triplets can be used. A minimum distance search between the source and the projected target triplets is conducted to attain the data set needed by the regressor to learn the intertask mapping. Any supervised learning algorithm can be used. However, the intertask mapping might be a highly complex relation and therefore, the nonparametric sparse Gaussian processes framework [11] (denoted by Phase Three in Fig. 2) is adopted.

This intertask mapping can now be used to transfer samples from the source to the target task. More specifically, starting from different initial states and using the source task's optimal policy $\pi_1^\star$, samples in form of state-action-successor states can be attained. These can be passed through the inter-task mapping to have an initial batch of samples in the target. Starting from these batches, a sample-based RL algorithm can be used to attain the target's optimal behavior.

4.2 Effective and Efficient Transfer

Having learned such an intertask mapping, the next stage is to make use of this mapping in order to aid the target agent when learning in a new task. Various methods and techniques have been proposed to deal with this problem as described in [12]. Indeed the algorithm will depend on the type of knowledge to be used. Transferring low level samples however, can also be considered most general as no abstraction assumptions are required. The problem definition can be stated as follows:

[5] Typically, $n_2 << n_1$ where only few transitions are available from the target task.

Effective & Efficient Transfer: Given an inter-task mapping $\mathfrak{X}_{\text{transfer}}$, source knowledge $\mathfrak{K}_{\text{source}}$, and additional target knowledge learn an optimal policy $\pi^{\star}_{\text{target}}$.

It is worth noting, that for optimal behaviour additional target knowledge is required. The transferred knowledge can aid the target agent by providing a good starting point. Further note, that no restrictive assumptions on the target basis $\mathfrak{B}_{\text{target}}$ are additionally imposed. In [2], two new and efficient algorithms for transferring samples were proposed. The two algorithms are based on sample efficient RL algorithms, namely Fitted-Q Iteration (FQI) and Least Squares Policy Iteration (LSPI).

Given the above, it is now possible to perform two steps autonomously. The first, is learning intertask mapping between source and target RL tasks without imposing any restrictive assumptions, while in the second knowledge is efficiently exploited to improve learning in the target task.

Aiming at a fully automated framework, one more question needs to be answered.

4.3 Choosing the Relevant Source Task

The question at this stage is given a target task and a bag of source tasks, how should an agent decide on what task to use?

Automatically choosing the relevant source task might be one of the toughest challenges in transfer learning. For an agent to choose a source task, a notion of distance between MDPs should be introduced.

The problem definition can be stated as follows:

Similarity Measure: Given the source and target task knowledge, $\mathfrak{K}_{\text{source}}$ and $\mathfrak{K}_{\text{target}}$, respectively, learn a measure $\mathfrak{d}\left(\mathfrak{K}_{\text{source}}, \mathfrak{K}_{\text{target}}\right)$.

To the best of our knowledge, there has been little progress on this goal in the literature thus far. Most relevant are works that use bisimulation metrics [5–7]. However, non of the aforementioned techniques can operate in: (1) continuous state spaces, and/or (2) between MDPs with different state and/or action spaces.

To remedy these problems, an extension of the algorithm proposed in [2] can be developed. Two additional steps are required. These are detailed next.

4.3.1 Density Estimation

After having source and target samples projected to the common space (i.e., Phase One and Phase Two in Fig. 2), a sparse density estimator can be used on the projected samples. The result of this estimation will be two probability density functions describing each of the source and target transitions in a *unified* space. This high level encoding can now be used to determine a distance between two MDPs even if the original state and/or action spaces differed.

4.3.2 Measure Between Densities

The problem of finding a distance between MDPs is now reduced to finding the distance between two probability density functions. Any symmetric measure between densities can now be used.

5 Overall Autonomous Framework

The overall framework is better explained using the following example, shown in Fig. 3. The target task is the so-called cart-pole problem. In this problem, there is a pole attached to a cart that can translate on a flat surface. The goal of the agent is to execute the correct linear actions so to control the pole in an upright position.

Furthermore, the target agent has access to a collection of source tasks. In Fig. 3, four different tasks are shown. The first is the inverted pendulum, where the goal is to choose the correct rotational actions such that the pole is controlled in an upright position. The difficulty in the task is that the actions are can not upswing the pole in one shot, rather it has to oscillate around its equilibrium position gaining enough momentum to swing upwards. The second task, is the mountain car. Starting at the bottom of the valley, the goal of the agent is to drive the car up the hill. However, the thrust of the engine is not enough for the car to reach the top of the hill in one shot. It rather has to oscillate to gain enough momentum to achieve the goal. The third task is the simple mass system. In this task the goal is to position the mass at a certain pre-specified value. Finally, in the fourth (i.e., the double mass system) the goal

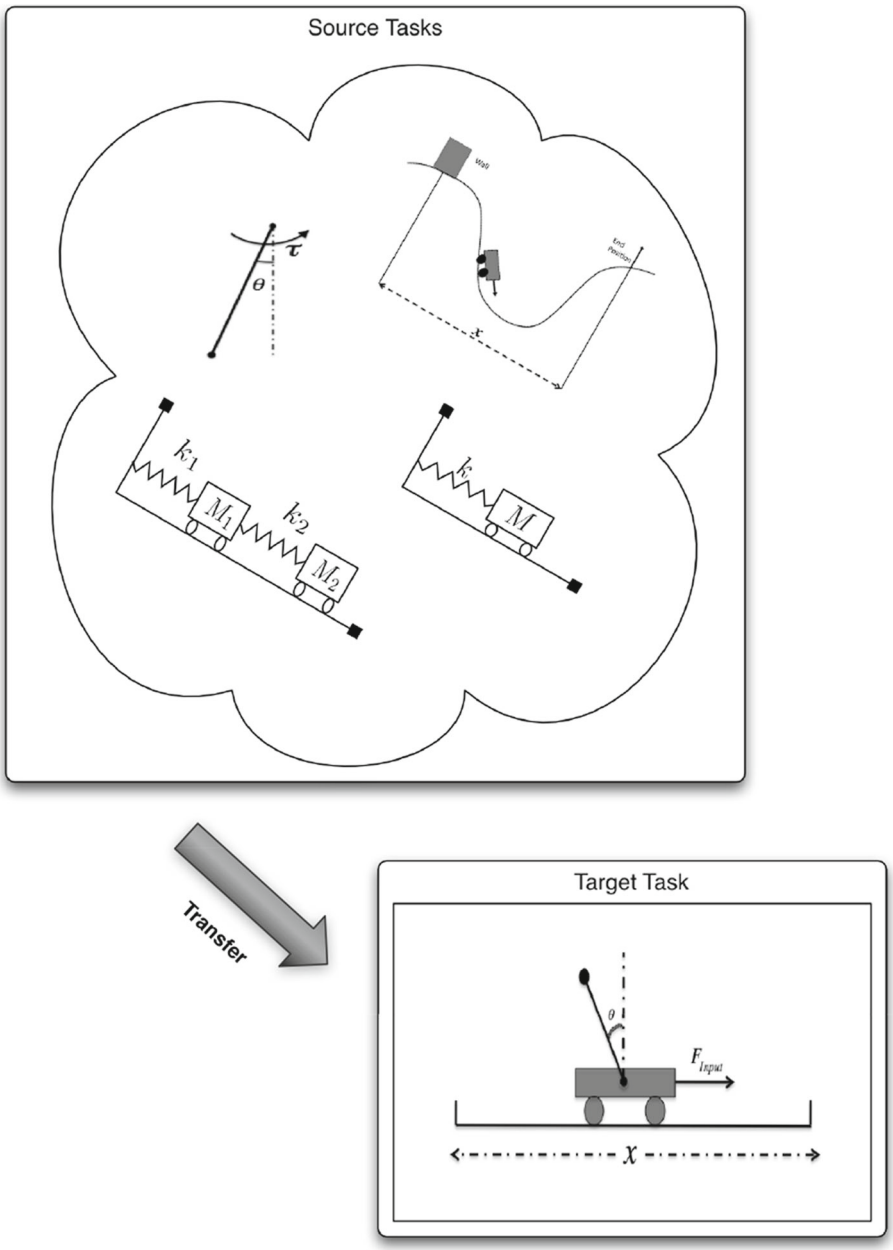

Fig. 3 Transfer Learning Example. The *upper panel* shows a bag of different source tasks, while the lowers shows a specific target task. To autonomously transfer, the agent has to: (1) choose a relevant source task(s), (2) reason about the relations between the tasks, and (3) effectively use the source task knowledge

is to control the first mass at a certain position while only being able to apply actions that affect the second mass.

To automatically transfer between one of these source tasks and the cart pole, the previous three steps need to be successfully executed. Firstly, the transfer agent has to decide on the source task to transfer from. This can be done using the approach described in Sect. 4.3

The agent has then to reason about the relation between the source and target task in order to successfully conduct transfer. This can be achieved using what is explained in Sect. 4.1 It is worth noting, that this transfer will not provide the optimal behavior in the target task, the agent has still to improve using normal reinforcement techniques to attain the optimal behavior. However, the hope is that the transferred knowledge can provide a "good" starting prior that can be used to improve the learning performance.

6 Conclusion and Future Work

In this discussion paper a unifying framework for various knowledge reuse algorithms has been introduced. Furthermore, the possibility of autonomous transfer was detailed. For autonomous transfer a set of three problems need to be solved. Learning an inter-task mapping and effective transfer can be achieved as discussed in the paper. To choose a relevant source task, however, a similarity

measure needs to be introduced. This can be achieved via density estimators in the unified knowledge space.

In the future work, we plan on studying such a measure in more details to enable full autonomy.

References

1. Abbeel P, Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the 21st international conference on Machine learning, ICML '04, ACM, New York, NY, USA
2. Ammar HB, Taylor ME, Tuyls K, Driessens K, Weiss G (2012) Reinforcement learning transfer via sparse coding (full paper). In: Proceedings of the 11th conference on Autonomous Agents and Multiagent Systems (AAMAS), Valencia
3. Argall BD, Chernova S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robot Auton Syst, 57(5):469–483
4. Buşoniu L, Babuška R, De Schutter B, Ernst D (2010) Reinforcement learning and dynamic programming using function approximators. CRC Press, Boca Raton
5. Castro PS, Precup D (2010) Using bisimulation for policy transfer in mdps. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1, AAMAS '10, International Foundation for Autonomous Agents and Multiagent Systems, Richland, pp 1399–1400
6. Ferns N, Panangaden P, Precup D (2004) Metrics for finite markov decision processes. In: Chickering DM, Halpern JY, (eds), UAI, AUAI Press pp 162–169
7. Ferns N, Panangaden P, Precup D (2011) Bisimulation metrics for continuous markov decision processes. SIAM J Comput, 40(6):1662–1714
8. Knox WB, Stone P, Breazeal C (2013) Teaching agents with human feedback: a demonstration of the tamer framework. In: IUI Companion, pp 65–66
9. Lee H, Battle A, Raina R, Ng AY (2007) Efficient sparse coding algorithms. In: In NIPS, NIPS pp 801–808
10. Ng AY, Harada D, Russell S (1999) Policy invariance under reward transformations: theory and application to reward shaping. In: In Proceedings of the 16th International Conference on Machine Learning, Morgan Kaufmann, pp 278–287
11. Snelson E, Ghahramani Z (2006) Sparse gaussian processes using pseudo-inputs. In: Advances in Neural Information Processing Systems, MIT press, pp 1257–1264
12. Taylor ME, Stone P (2009) Transfer learning for reinforcement learning domains: a survey. J Mach Learn Res, 10:1633–1685
13. Taylor ME, Stone P, Liu Y (2007) Transfer learning via inter-task mappings for temporal difference learning. J Mach Learn Res 8(1):2125–2167