State of the Art on: Transfer learning in reinforcement learning

Riccardo Poiani, poiani.riccardo@gmail.com

1. Introduction to the research topic [max 2 pages]

Machine learning (ML) is a subfield of artificial intelligence whose aim is the design of algorithms able to learn from data exploiting statistical tools. In *reinforcement learning* (RL) an agent acts in an unknown or not completely known environment with the goal of maximizing an external reward signal (Sutton and Barto, 1998). *Transfer learning* (TL) is another ML subfield that focuses on exploiting knowledge or structure derived from a learning problem to enhance learning on a related problem. Many approaches have been proposed concerning transfer learning in reinforcement learning (TLRL), which depend on the type of structure it is possible to exploit, what it is retained useful to transfer, and the goal of the transfer step. As already shown in different studies, the importance of these methodologies is that they can be used to speed up learning in a very effective way.

Looking at the expert's opinions and at the attendance trends, for what concerns the general topic of artificial intelligence, some relevant conferences are AAAI and IJCAI. NeurIPS and ICML are instead focused on machine learning. Important journals are instead *the Journal of Machine Learning Research* (Microtome), *Transactions on Pattern Analysis and Machine Intelligence* (IEEE), *Machine Learning* (Springer). All of these mentioned conferences and journals are also highly valued in ranking portals, such as [1].

1.1. Preliminaries

Reinforcement learning
The problem of RL is usually formalized as a *Markov decision process* (MDP) [32], that is defined as a tuple $\langle\ \mathcal{S},\ \mathcal{A},\ \mathcal{P},\ \mathcal{R},\ \gamma\ \rangle$, in which $\mathcal{S}$ is the state space in which the agent moves, $\mathcal{A}$ is the set of available actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathbb{P}(\mathcal{S})$ is a transition function, denoting the probability of reaching a given state taking an action in a state, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward signal beforementioned, and $\gamma \in [0, 1)$ is the discount factor. A stationary randomized control policy is a mapping $\pi : \mathcal{S} \to \mathbb{P}(\mathcal{A})$. At each time stamp (t) the agent chooses an action according to $a_t \sim \pi(\cdot|s_t)$. Solving an MDP means finding the *optimal policy* π^*, that is the policy that maximizes the accumulated discounted reward. RL comes into help when the dynamics of the environment are unknown or difficult to be modelled, or when the model of the environment is too complex to be solved exactly, so that approximate solutions are searched for.

The Markovian property, that is $\mathbb{P}(s_{t+1}|s_t, a_t) = \mathbb{P}(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ..., s_0, a_0)$, or, in a more informal way, the future is independent of the past given the present, is an important property of these stochastic processes.
The value function $V^\pi(s)$ and the action value function $Q^\pi(s, a)$ of a policy π are defined as the expected discounted reward starting from state *s* or action pair *(s,a)*, respectively, and following policy π. The value function and action value function associated to π^* are usually denoted as Q^* and V^*. As already mentioned, a typical goal of RL is maximizing the cumulated reward, learning π^*. An equivalent goal is the one of minimizing the *regret*, that is defined as the difference, in terms of cumulated rewards, between the optimal policy and the learnt policy.

RL algorithms applied to MDPs can be broadly categorized into model-based and model-free approaches: the former use estimates of the empirical reward and transition distributions to solve the problem, while the latter do not keep information about the model of the MDP.

In recent years, it is common to design both RL algorithms in terms of *sample complexity of exploration*, i.e. the number of timestamps t in which the policy is sub-optimal w.r.t. a fixed quantity $\epsilon > 0$, that is $V^t(s_t) < V^*(s_t) - \epsilon$.

This definition may change a bit w.r.t. the settings that the algorithms encounter. In other words, the goal is usually to design an algorithm that can transfer in such a way as to be able to trade-off exploration of the state action space and exploitation of the knowledge it has from source tasks and the knowledge it has collected in the target task.

Transfer learning
The core idea of TLRL is that the experience gained in learning to perform one or more RL task (*source tasks*) can help to improve learning performance in a related task (*target task*) [36]. Generally speaking, it can be formalized as follows: given a set of pairs (m, e_m), where m is a source task and e_m is the experience collected under m, and some performance metric p, the goal is to achieve better results according to p in a target task m_{new} w.r.t. the case where the beforementioned set was not present.

As in the standard RL case, the sample complexity is also used in transfer settings.
Although TL applied to RL has been shown to be able to speed up learning, it is not always possible to read in the literature guarantees on when the approach will fail. For instance, using only non-related source tasks will hardly provide benefits in the target tasks and more likely they will hurt performances. This problem is referred to as *negative transfer*. Designing algorithms that avoid negative transfer is an important aspect.

Tools
According to the last data available on the web, *Python* is the most widely used programming language in the ML field. Several efficient Python libraries can be adopted to support the development of machine learning projects. Some of the most popular examples are: *scikit-learn*[31], *TensorFlow*[2], *theano*[38], *Keras*[16], and *Pytorch*[30].

For what concerns RL, the developer has usually to develop at least a simulated environment in which the agent should run the learning algorithm under consideration. Some frameworks that provides this possibility are *Arcade Learning Environment* [11], *OpenAIgym* [13] and *DeepMind Lab* [10]. It is important to note that, while for ML there exists standard datasets on which it is possible to test the algorithms, the RL research community has yet to adopt a unified benchmark of problems. However, there are a number of problems of different complexity that are often considered to test the performance of a learning algorithm. For instance, some common settings that consider continuous state-action spaces are *cartpole*, *acrobot* and *mountain car*. *Atari games* are also used to test the quality of a solution. For what concerns discrete spaces, it is possible to discretize continuous settings or, for example, use a *grid navigation world* with the purpose of reaching a goal state.
For what concerns transfer, to the authors knowledge, there is no particular framework or library that supports the development of TLRL algorithms, and the problem exposed for RL extends to this more specific setting. Again, problems mentioned above can be slightly modified and adapted to the transfer context. For instance, it is possible to study a grid navigation world that has a door whose position varies between the tasks, and the goal of the agent is to reach a certain goal state.

1.2. Research topic

RL algorithms have already shown their great capabilities in many application fields [26] [23] [15]. Nevertheless, two problems keep affecting the usage of these techniques, which are the sample complexity and safety. The former is mainly due to the fact that training the algorithms can be really a slow process and, even for not so complex problems, many samples are needed to reach good performance levels. The latter is because, at the beginning of the training phase, the agent may act in a dangerous way, taking actions that none of us would consider intelligent. This can be particularly dangerous for applications involving robots: for instance, considering the case of autonomous driving, if the agent takes very bad decision, life of other people can be at risk. Transfer learning applied to the field of reinforcement learning can help to address these classic problems speeding up the learning phase by re-using information and structure from previously solved tasks. Taking, for example, the case of a robot that has to learn how to swing a club of a certain mass m and a certain length l: if knowledge on how to swing clubs of different weight and length is available, the learning process can easily speed up by means of transfer.

2. Main related works [max 3 pages]

2.1. Classification of the main related works

In transfer learning there are many different possibilities and settings defined by: *the task difference assumptions*, *the source task selection*, that is if the agent has to select relevant source tasks or if they are already provided, *task mappings*, that is the knowledge that the agent has on how tasks are related to each other, *transferred knowledge*, that is the type of information that has to be transferred, and, finally, *the allowed learners*, which are possible restrictions that the agent may have in terms of algorithm to apply.
When dealing with TL, it is possible to encounter different evaluation metrics in the literature, such as the *total training complexity*, *the initial performance* (*jumpstart*), *the total reward* (or equivalently *regret*), *the transfer ratio* (i.e. the ratio between the total reward accumulated by the transfer learner divided by the reward accumulated by the non-transfer learner) or *the time to reach a pre-specified performance level*. However, in recent works, the most used seems to be the regret and the jumpstart performances.
There are also other differences between transfer learning approaches. Indeed, it is possible to consider different settings that the agent may have to face when dealing with tasks. For instance, the agent may face tasks sampled from a certain distribution one after the other (i.e. *lifelong RL* [3]), or it may have a set of sources tasks, finite or infinite, from which it can extract information.
Due to space limitation, in the rest of this document, the focus will be mainly on the dimensions of task difference, knowledge that is available in the source tasks, and the evaluation metric.
There are various steps that can be involved in the transfer: the agent should begin from the given source tasks and select the ones to be transferred; then it should learn the relationship between these selected tasks and the target task; and, finally, it should transfer knowledge from the sources to the target. Usually, research studies and algorithms focus only on one of these steps, while a unified framework that performs all these tasks at the same time is still missing. Moreover, the different performance metrics and settings lead to difficulties in comparing different approaches.

2.2. Brief description of the main related works

Model-free approaches
Most of the transfer approaches (see Table 1) present in the literature are model-free.
In many model-free approaches, value functions are subject to transfer. For instance, in [39] the agent is given a finite set of source tasks sampled from some distribution and a parametric approximation to their optimal value functions is available. The source tasks are used to learn a prior distribution over the optimal value functions and provide a variational approximation of the corresponding posterior in a new target task. The proposed method allows tasks to have different transition matrices and rewards, and works online using regret as a performance index. The approach investigated in [42] focuses on providing theoretical guarantees, under some assumptions, for the convergence rate of transferring action-value functions to a target task solved via Q-learning [43]. They again take into consideration the case in which the transition matrix and rewards can be different, and consider the regret as the performance metric.

A different approach is based on transferring samples from the source tasks to the target task. When only the reward function is different, [25] suggests that the transitions samples obtained from a task can be reused in any other task, and combine this idea with the *optimism in the face of uncertainty* principle [20]. Another approach that considers the possibility of transferring samples is the one presented in [6]: they consider an approach that applies to a lifelong learner that operates in an adversarial setting and has to learn multiple tasks online while enforcing safety constraints on the learned policy. Here, both the reward and the transition matrix can differ.

Other methods such as [27], instead, implement a mechanism for re-useing previously learned policies: their algorithm is able to learn when and which source policy is best to re-use, as well as when to terminate its reuse. The RLPA algorithm proposed in [8] always considers to reuse policies, but it exploits a set of input policies

and choose the best one in the set. The authors prove that the regret is sub-linear in $\sqrt{T}$ and independent of the size of the state and action space. The work exposed in [3], instead, identifies the initial policy that optimizes the expected performance over the distribution of tasks for increasingly complex classes of policy and task distributions. Simultaneously, they consider an initialization method for R_{max} [12] and Delayed Q-learning [34] algorithms that preserves the PAC guarantees. The work proposed in [29], instead, is based on clustering and considers the lifelong RL setting: when the number of MDPs grows too much, the transfer becomes ineffective because the agent should take too long to test old policies. So, the algorithm clusters them and chooses as source tasks the representatives of each cluster. Policies from source tasks are re-used following an algorithm that takes inspiration from EXP-3 [7]. In the case of deep RL [24], an approach that considers policies is exposed in [37], where multiple agents are trained together while being forced to stay close to a policy that is shared among tasks. Other algorithms studied to empower deep RL algorithms have also been investigated recently, for instance in [9] and [44]. Another example is presented in [17], which shows that neural network policies can be decomposed into "task-specific" and "robot-specific" modules: the first can be shared among robots, while the latter can be shared between multiple tasks in a robot.

When any task difference is allowed, [4] develops a method to optimize a shared repository of transferable knowledge and learns projection matrices that specialize that knowledge to different task domains. The information available to carry out the transfer is sample. The sample-based approach has been adopted also in [35]: here samples are selected to be transferred to a target task that is being learned with a model-based RL algorithm. Other studies, such as [5], use an unsupervised manifold alignment to learn inter-task mappings and transfer samples between different task domains. [22] shows, instead, that an agent can learn a portable shaping function from experience in a sequence of tasks.

Model-based approaches
For what concerns the model-based category, instead, [41] and [40] consider transferring all the samples collected from the source tasks, by estimating the relevance of each datum to solve the target task. They differ from previous approaches, in which only some samples were selected, but then the discrepancies between tasks were no longer considered. This approach turns out to be empirically very robust to negative transfer. Another algorithm able to avoid negative transfer is presented in [14]; the authors show that the sample complexity is significantly reduced in many cases. A different model-based approach is presented in [18], which introduces a framework to parametrize a family of related dynamical systems with a low-dimensional set of latent factors by using Gaussian processes [33] and Indian buffet processes [19]. However, non-linear mechanics could not be captured by their model: a problem that has been overcome with the study discussed in [21], which substitutes Gaussian processes with Bayesian neural networks [28]. This choice empowers both the capabilities of the model and increases the computational efficiency. For what concerns the transfer of policies, the work in [45] formally defines a setting where multiple teacher agents can provide advice to students, and introduces an algorithm to leverage advices and exploration. The authors are also able to quantify the negative transfer.

2.3. Discussion

Transferring knowledge is something that is embedded in every intuitive concept of intelligence; humans, indeed, learn most of the tasks by looking at how others solve them. For instance, when someone is learning to drive a car, he/she already knows a lot of useful information that he/she has collected over time. This is because he/she has already spent a lot of time in a car with someone else driving, and he/she has prior knowledge of what happens by pressing the different pedals, steering, and so on. Moreover, he/she also has a good understanding of the physics of the world: in the end, what he/she has to learn are very few things. Being able to transfer knowledge between tasks is an important problem for reinforcement learning agents because it can effectively increase performances. Despite the success of recent work, the problem is still far from being solved: humans are able to do that very quickly, especially for similar tasks. Moreover, they are able to perform all the chain of the transfer process effectively, sometimes without even reasoning about the phenomena: they can select relevant source tasks, understand how they relate to the target, and finally transfer the knowledge needed to behave well

Transfer learning for reinforcement learning							
Paper	**P**	**R**	**All**	**Knowledge**	**Metric**	**M-free**	**M-based**
Tirinzoni et al. [2018] [39]	X	X		V	Regret	X	
Wang et al. [2018]2[42]	X	X		Q	Regret	X	
Laroche [2017] [25]		X		Samples	Regret	X	
Ammar [2015] [6]	X	X		Samples	Regret	X	
Li et al. [2018][27]	X	X		Policy	Regret	X	
Azar et al. [2013][8]	X	X		Policy	Regret	X	
Abel et al. [2018][3]		X		V/Q	Jumpstart/Regret	X	
Mahmud et al [2013][29]	X	X		Policy	Regret	X	
Teh et al. [2017][37]	X	X		Shared Policy	Regret	X	
Barreto et al. [2019][9]		X		Policy/V	Jumpstart	X	
Yin et al. [2017] [44]	X	X		Samples	Regret	X	
Devin et al. [2016] [17]	X	X		Policy	Jumpstart	X	
Ammar et al. [2015] [4]			X	Samples	Regret	X	
Taylor et al. [2008] [35]			X	Samples	Regret	X	
Ammar et al. [2015] [5]			X	Policy	Jumpstart/Regret	X	
Konidaris et al. [2012] [22]			X	Policy/Value	Regret	X	
Tirinzoni et al. [2018] [41]	X	X		Samples	Regret		X
Tirinzoni et al. [2019 [40]	X	X		Samples	Regret		X
Brunskill et al. [2013] [14]	X	X		Samples	Regret		X
Doshi-Velez et al. [2013] [18]	X			Samples	Regret		X
Killian et al. [2017] [21]	X			Samples	Regret		X
Zhan et al. [2016] [45]				Policy	Regret		X

Table 1: Table summarizing the recent works mentioned in this document on the transfer learning in reinforcement learning settings. The dimensions used to categorize the algorithms are the allowed differences between the tasks (P for the transition matrix, R for the reward, All when everything can change), the available knowledge (Knowledge), the metric that has been used to evaluate the algorithms (Metric), and whether the approach is model-free (M-free) or model-based (M-based).

in a new task.

The variety of possible settings, which can create some confusion in the beginning, is a sign that many roads are still open for future research. For instance, although recently some algorithms are robust w.r.t. negative transfer, it is still not clear how is it possible to select the tasks a priori. Indeed, commonly in the literature the algorithms are showed to not perform worse than the case where no task is available: an important improvement direction is to understand a priori if the tasks are useful. This, for instance, would help a lot in selecting a proper algorithm for a given problem.

Moreover, providing algorithms that are able to carry out tasks transferring knowledge and requiring lower sample complexity is still possible in many settings. In many cases, a lower bound is not even present. Having lower bounds and algorithms that match them is an important contribution since it is useful both for worst-case efficiency reasons and for measuring progress in the research area.

References

[1] Core. http://www.core.edu.au/conference-portal. Accessed: 2019-10-30.

[2] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] Abel, D., Jinnai, Y., Guo, S. Y., Konidaris, G., and Littman, M. Policy and value transfer in lifelong reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning* (Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018), J. Dy and A. Krause, Eds., vol. 80 of *Proceedings of Machine Learning Research*, PMLR, pp. 20–29.

[4] Ammar, H. B., Eaton, E., Luna, J. M., and Ruvolo, P. Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning. In *Proceedings of the 24th International Conference on Artificial Intelligence* (2015), IJCAI'15, AAAI Press, pp. 3345–3351.

[5] Ammar, H. B., Eaton, E., Ruvolo, P., and Taylor, M. Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment.

[6] Ammar, H. B., Tutunov, R., and Eaton, E. Safe policy search for lifelong reinforcement learning with sublinear regret, 2015.

[7] Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing 32*, 1 (2002), 48–77.

[8] Azar, M. G., Lazaric, A., and Brunskill, E. Regret bounds for reinforcement learning with policy advice, 2013.

[9] Barreto, A., Borsa, D., Quan, J., Schaul, T., Silver, D., Hessel, M., Mankowitz, D., Žídek, A., and Munos, R. Transfer in deep reinforcement learning using successor features and generalised policy improvement, 2019.

[10] Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S. Deepmind lab, 2016.

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

[12] Brafman, R. I., and Tennenholtz, M. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res. 3* (Mar. 2003), 213–231.

[13] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

[14] Brunskill, E., and Li, L. Sample complexity of multi-task reinforcement learning, 2013.

[15] Cai, Q., Filos-Ratsikas, A., Tang, P., and Zhang, Y. Reinforcement mechanism design for fraudulent behaviour in e-commerce.

[16] Chollet, F., et al. Keras. https://keras.io, 2015.

[17] Devin, C., Gupta, A., Darrell, T., Abbeel, P., and Levine, S. Learning modular neural network policies for multi-task and multi-robot transfer, 2016.

[18] Doshi-Velez, F., and Konidaris, G. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations, 2013.

[19] Griffiths, T. L., and Ghahramani, Z. The indian buffet process: An introduction and review. *J. Mach. Learn. Res. 12* (July 2011), 1185–1224.

[20] Kamiura, M., and Sano, K. Optimism in the face of uncertainty supported by a statistically-designed multi-armed bandit algorithm. *Biosystems 160* (08 2017).

[21] Killian, T., Daulton, S., Konidaris, G., and Doshi-Velez, F. Robust and efficient transfer learning with hidden-parameter markov decision processes, 2017.

[22] Konidaris, G., Scheidwasser, I., and Barto, A. G. Transfer in reinforcement learning via shared features. *J. Mach. Learn. Res. 13* (May 2012), 1333–1371.

[23] Lange, S., Riedmiller, M., and Voigtlander, A. Autonomous reinforcement learning on raw visual input data in a real world application. pp. 1–8.

[24] Lapan, M. *Deep Reinforcement Learning Hands-On*. Packt Publishing, Birmingham, UK, 2018.

[25] Laroche, R., and Barlier, M. Transfer reinforcement learning with shared dynamics. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (2017), AAAI'17, AAAI Press, pp. 2147–2153.

[26] Li, C. Y., Liang, X., Hu, Z., and Xing, E. P. Hybrid retrieval-generation reinforced agent for medical image report generation, 2018.

[27] Li, S., Gu, F., Zhu, G., and Zhang, C. Context-aware policy reuse, 2018.

[28] MacKay, D. J. C. A practical bayesian framework for backpropagation networks. *Neural Comput. 4*, 3 (May 1992), 448–472.

[29] Mahmud, M., Hawasly, M., Rosman, B., and Ramamoorthy, S. Clustering markov decision processes for continual transfer.

[30] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop* (2017).

[31] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[32] Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[33] Rasmussen, C. E., and Williams, C. K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[34] Strehl, A. L., Li, L., Wiewiora, E., Langford, J., and Littman, M. L. Pac model-free reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning* (New York, NY, USA, 2006), ICML '06, ACM, pp. 881–888.

[35] Taylor, M. E., Jong, N. K., and Stone, P. Transferring instances for model-based reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases* (Berlin, Heidelberg, 2008), W. Daelemans, B. Goethals, and K. Morik, Eds., Springer Berlin Heidelberg, pp. 488–505.

[36] Taylor, M. E., and Stone, P. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res. 10* (Dec. 2009), 1633–1685.

[37] Teh, Y. W., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distral: Robust multitask reinforcement learning, 2017.

[38] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints abs/1605.02688* (May 2016).

[39] Tirinzoni, A., Rodriguez Sanchez, R., and Restelli, M. Transfer of value functions via variational methods. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 6179–6189.

[40] Tirinzoni, A., Salvini, M., and Restelli, M. Transfer of samples in policy search via multiple importance sampling. In *Proceedings of the 36th International Conference on Machine Learning* (Long Beach, California, USA, 09–15 Jun 2019), K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97 of *Proceedings of Machine Learning Research*, PMLR, pp. 6264–6274.

[41] Tirinzoni, A., Sessa, A., Pirotta, M., and Restelli, M. Importance weighted transfer of samples in reinforcement learning, 2018.

[42] Wang, Y., Meng, Q., Cheng, W., Liug, Y., Ma, Z.-M., and Liu, T.-Y. Target transfer q-learning and its convergence analysis, 2018.

[43] Watkins, C. J. C. H., and Dayan, P. Q-learning. *Machine Learning 8*, 3 (May 1992), 279–292.

[44] Yin, H., and Pan, S. J. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (2017), AAAI'17, AAAI Press, pp. 1640–1646.

[45] Zhan, Y., Ammar, H. B., and taylor, M. E. Theoretically-grounded policy advice from multiple teachers in reinforcement learning settings with applications to negative transfer, 2016.