Experiments with Adaptive Transfer Rate in Reinforcement Learning

Yann Chevaleyre[1], Aydano Machado Pamponet[2], and Jean-Daniel Zucker[3]

[1] Universit Paris-Dauphine
yann.chevaleyre@lamsade.dauphine.fr
[2] Universit Paris 6
aydano.machado@lip6.fr
[3] IRD UR Godes
Centre IRD de l'Ile de France, Bondy, France
jean-daniel.zucker@ird.fr

Abstract. Transfer algorithms allow the use of knowledge previously learned on related tasks to speed-up learning of the current task. Recently, many complex reinforcement learning problems have been successfully solved by efficient transfer learners. However, most of these algorithms suffer from a severe flaw: they are implicitly tuned to transfer knowledge between tasks having a given degree of similarity. In other words, if the previous task is very dissimilar (resp. nearly identical) to the current task, then the transfer process might slow down the learning (resp. might be far from optimal speed-up). In this paper, we address this specific issue by explicitly optimizing the transfer rate between tasks and answer to the question : "can the transfer rate be accurately optimized, and at what cost ?". We show that this optimization problem is related to the continuum bandit problem. We then propose a generic adaptive transfer method (AdaTran), which allows to extend several existing transfer learning algorithms to optimize the transfer rate. Finally, we run several experiments validating our approach.

1 Introduction

In the reinforcement learning problem, an agent acts in an unknown environment, with the goal of maximizing its reward. All learning agents have to face the exploration-exploitation dilemma: whether to act so as to explore unknown areas or to act consistently with experience to maximize reward (exploit). Most research on reinforcement learning deal with this issue. Recently Strehl et al. [17] showed that near optimal strategies could be reached in as few as $\widetilde{O}(S \times A)$ time steps. However, in many real-world learning problems, the state space or the action space have an exponential size.

One way to circumvent this problem is to use previously acquired knowledge related to the current task being learned. This knowledge may then be used to guide exploration through the state-action space, hopefully leading the agent towards areas in which high rewards can be found. This knowledge can be acquired in different ways:

- By imitation: in particular, in a multi-agent environment, agents may observe the behavior of one another and use this observation to improve their own strategy [13].
- By bootstrap: related tasks may have been previously learned by reinforcement [14,1,10] and the learned policy may be used to bootstrap the learning.
- By abstraction: a simplified version of the current task could have been generated to quickly learn a policy which could be used as a starting point for the current task [18,21].
- By demonstration: a human tutor may provide some explicit knowledge. Other similar settings exist in the literature, among which "advice taking" [19] or "apprenticeship" [20].

In this paper, we will focus on a simple version of the "bootstrap" transfer learning problem [1]: we will assume that a policy (or a Q-value function) is available to the learner, and that this policy has been learned on a past task which shares the *same state-action space* as that of the current task.

Given this knowledge, the learning agent faces a new dilemma: it has to balance among following the ongoing learned policy and exploring the available policy. Most transfer learners do not tackle this dilemma explicitly: the amount of exploration based on the available policy does not depend on its quality. Ideally, this amount should be tuned such that the transfert learner be *robust* w.r.t. the quality of the past policy : good policies should speedup the learner while bad ones should not slow it down significantly. Recently, a new approach has been proposed to solve this issue [10,11]. The main idea of this approach is to estimate such similarity between the two tasks, and then to use this estimate as a parameter of the transfer learning process, balancing between ongoing and past policies. However, measuring this similarity is a costly process in itself and moreover there are no guarantee that this similarity optimizes the transfer learning process.

In this paper, we show that a parameter called the *transfer rate* controling the balance between past policy and the ongoing policy can be optimized efficiently *during* the reinforcement learning process. For this purpose, we first show in which way this optimization problem is related to the *continuum-armed bandit* problem. Based on this relation, we propose a generic adaptive transfer learning method (AdaTran) consisting of a wrapper around some standard transfer learning algorithm, and implementing a continuum-armed bandit algorithm.

We choose two representative transfer approaches, namely *Probabilistic Policy Reuse* (PPR, [1]) and *Memory-guided Exploration* (MGE, [12]) which, wrapped inside AdaTran, will be referred to as AdaTran(PPR) and AdaTran(MGE). We show experimentally on a grid-world task that AdaTran is much more robust than non adaptive transfer algorithms.

The paper is organized as follows. After some preliminaries and a state of the art on transfer learners, we introduce the continuous bandit problem and relate it to the optimization of the transfer rate. The following section introduces the AdaTran framework and its instantitation AdaTran(PPR) and AdaTran(MGE). Then, a set of experiments in which AdaTran is compared to standard transfer learners assesses both its robustness and efficiency.

2 Preliminaries

Reinforcement learning problems are typically formalized using Markov Decision Processes (MDPs). An MDP M is a tuple $\langle S, A, T, r, \gamma \rangle$ where S is the set of all states, A is the set of all actions, T is a state transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$, r is a reward function $r : S \times A \rightarrow \mathbb{R}$, and $0 \leq \gamma < 1$ is a discount factor on rewards. From a state s under action a, the agent receives a stochastic reward r, which has expectation $r(s,a)$, and is transported to state s' with probability $T(s,a,s')$. A policy is a strategy for choosing actions. If it is also deterministic, a policy can be represented by a function $\pi : S \rightarrow A$. As in most transfer learning settings, we assume that the learning process is divided into episodes : at the beginning of an episode, the agent is placed on a starting state sampled from a distribution $\mathcal{D}$. The episode ends when the agent reaches a special absorbing state (the goal), or when a time limit is reached.

For any policy π, let $V^{\pi}_{M}(s)$ denote the discounted value function for π in M from state s. More formally, $V^{\pi}_{M}(s) \triangleq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where $r_0, r_1, \ldots$ is the reward sequence obtained by following policy π from state s. Also, let $V^{\pi}_{M} \triangleq \mathbb{E}_{s \sim \mathcal{D}}[V^{\pi}_{M}(s)]$. To evaluate the quality of an action under a given policy, the Q-value function $Q^{\pi}(s,a) \triangleq r(s,a) + \gamma \mathbb{E}_{s' \sim T(s,a,.)}\left[V^{\pi}(s')\right]$ is generally used (Here, as there are no ambiguity, M has been omitted). The optimal policy π^* is the policy maximizing the value function. The goal of any reinforcement learning algorithm is to find a policy such that the agent's performance approaches that of π^*.

To speed up learning on a new task, transfer learners exploit knowledge previously learned on a past task. Here, we will assume an in [1] that the past task and the current task have the same state-action space. We study the case where the available knowledge has the form of a policy $\bar{\pi}$ learned on the past task or of a Q-value function $\bar{Q}$. Both cases lead to different transfer learners, such as PPR and MGE.

3 Transfer Learners with Static Transfer Rates

In this section, we will present two state-of-the-art transfer learners, namely PPR (*Probabilistic Policy Reuse*, as well as PPR-decay, a variation on PPR [1]) and MGE (Memory-Guided Exploration [12]), exhibiting a parameter which controls the balance between the ongoing learned policy and $\bar{\pi}$. As most transfer methods, PPR and MGE have been directly build on a standard Q-learner, and thus share the same structure. The only difference with a Q-learner lies in the action selection method (referred here as *ChooseAction*).

The most widely used transfer method probably is Q-reuse ([14], also sometimes referred to as *direct transfer*). Based on a Q-learner, this method simply uses $\bar{Q}$ to initialize the Q-values of the current task. Caroll *et al* note in [12] that with this approach, "if the tasks are too dissimilar, the agent will spend too much time unlearning...". To overcome this drawback, they propose an alternative to Q-reuse, namely *Memory-Guided Exploration*, a standard Q-learner in which the action selection procedure has been replaced as shown in table 1. Here, ξ is a

Table 1. Examples of $ChooseAction(s_t, \bar{\pi}, \varphi)$ functions in static transfer learners

$ChooseAction(s_t, \bar{\pi}, \varphi)$	Name of transfer algorithm
$a_t = argmax_a a \left\{(1-\varphi)Q_t(s_t,a) + \varphi \bar{Q}_t(s_t,a) + \xi\right\}$ where ξ is some real-valued random variable.	MGE [12] (Memory Guided Exploration)
$a_t = \begin{cases} \bar{\pi}(s_t) & w.proba.\varphi \times 0,95^t \\ \epsilon\text{-greedy}(\pi) & otherwise \end{cases}$	PPR-decay [1] (PPR with exponential decay)
$a_t = \begin{cases} \bar{\pi}(s_t) & with\, proba.\, \varphi \\ \epsilon\text{-greedy}(\pi) & with\, proba\, 1-\varphi \end{cases}$	PPR (Probabilistic Policy Reuse)

random variable (e.g. normal distributed with zero mean) used to add random exploration. Clearly, the parameter φ influences the procedure: if $\varphi \to 0$, then the algorithm is similar to a standard Q-learner, and if $\varphi \to 1$, it always follow $\bar{\pi}$.

Recently, Fernandez and Veloso proposed a completely different probabilistic approach to transfer learning (PPR-decay [1]) also based on a Q-learner. At each step, the algorithm randomly chooses to follow the policy ϵ-greedy(π) or to follow $\bar{\pi}$, as depicted in table 1. Here, π refers to the policy induces by the Q-values ($\pi(s) = argmax_a Q_t(s,a)$) and ϵ-greedy(π) refers to the policy obtained by choosing π with probability $1-\epsilon$, or a random action with probability ϵ. Fernandez *et al.* proposed arbitrarily to initialize φ to one at the beginning of each episode, and to decrease its influence at each step t by $0,95^t$. As for MGE, PPR-decay mimics a Q-learner when $\varphi = 0$, but does not follow $\bar{\pi}$ at each step when $\varphi = 1$, because of the decay. Therefor, we introduce a variation on PPR-decay, namely PPR, in which φ is not decreased during the episode.

Clearly, φ can be seen here as a parameter controlling the *transfer rate*, although it does not have exactly the same role in PPR and MGE. It is not hard to see that this rate should be dependent on the similarity between the past and the current task. Computing such a similarity is difficult in the general case, and that optimizing φ can be done during learning, as shown in the next sections.

4 Optimization of the Transfer Rate as a Stochastic Continuum-Armed Bandit Problem

Consider a transfer method such as one of those discussed above, in which a parameter $\varphi \in [0,1]$ controls the transfer rate, in such a way that if $\varphi = 0$, the policy $\bar{\pi}$ is not being used, and if $\varphi = 1$, the agent follows exclusively $\bar{\pi}$. Let us consider the problem of optimizing φ, in order to improve the speedup learning. For the sake of simplicity, adjustment of φ will occur only after each episode, thus exploiting the sequence of rewards gathered during the last episode.

Consider a learning episode starting at time t. Before the episode begins, the agent has to choose a value of φ, which ideally would yield the highest expected gain $V_t(\varphi) \triangleq \mathbb{E}\left[r_t + \gamma r_{t+1} + \ldots \mid \varphi\right]$. At the end of the episode, the agent can compute $\sum_k r_{t+k}\gamma^k$ which is an unbiased estimator of $V_t(\varphi)$. Choosing the best value for φ is challenging, as gradient methods which require the knowledge of

$\frac{\partial V_t}{\partial \varphi}$ might not be applicable. It turns out that this problem is a typical *multi-armed bandit problem.*

The *continuum armed-bandit* problem which belongs to the well known family of multi-armed bandit problems, is a particularly appropriate setting for the optimization of $V_t(\varphi)$. In [8], a stochastic version of this problem is presented, for which the specific algorithm UCBC was designed. In [7], an algorithm called CAB1 is described, in order to solve the *adversarial* version of this problem. In our setting, the gain $\sum_k r_{t+k}\gamma^k$ is stochastic, and changing in time. Thus, we will need a generalization of these two settings, namely the *stochastic adversarial continuum armed-bandit problem*, which can be described as follows

Definition 1. (The stochastic adversarial continuum armed-bandit problem) *Assume the existence of an unknown distribution family $P(. \mid x, t)$ indexed by $x \in [0,1]$ and $t \in \{1 \dots n\}$. At each trial t, the learner chooses $X_t \in [0,1]$ and receives return $Y_t \sim P(. \mid X_t, t)$. Let $b_t(x) \triangleq \mathbb{E}[Y_t \mid X_t = x, t]$. The agent's goal is to minimize its expected regret $\mathbb{E}[\sum_t b_t(x^*) - \sum_t Y_t]$, given that $x^* = \sup_{x\in[0,1]} \sum_{t=1}^n b_t(x)$.*

Although this setting seems more general than the adversarial case, the regret guarantees of CAB1 still hold here. Investigating regret bounds is beyond the scope of this paper. However, the reader can refer to theorem 3.1 of [7], which proof can easily be generalized to the stochastic adversarial setting.

5 AdaTran: A Generic Adaptive Transfer Framework

We now present a generic adaptive transfer learning algorithm, which can be seen as a wrapper around a transfer learner, optimizing the transfer rate φ using a *stochastic adversarial continuum armed-bandit* algorithm referred to as *UpdateContBandit*. This leads to the *AdaTran* wrapper, a generic adaptive transfer algorithm in which many transfer learners can be implemented. Note that even though most transfer learners do not have such a parameter, they can often be modified so as to make φ appear explicitly.

Algorithm 1. AdaTran

```
1: Init()
2: $t \leftarrow 0$
3: $\varphi \leftarrow \varphi_0$
4: for each episode $h$ do
5:     set the initial state $s$
6:     while (end of episode not reached) do
7:         $a_t = ChooseAction(s_t, \bar{\pi}, \varphi)$
8:         Take action $a_t$, observe $r_{t+1}, s_{t+1}$
9:         $Learn(s_t, a_t, r_{t+1}, s_{t+1})$
10:        $t \leftarrow t + 1$
11:    end while
12:    $\varphi \leftarrow UpdateContBandit(\bar{\pi}, \varphi, \langle r_1, r_2, \dots \rangle)$
13: end for
```

Depending of the function used for *ChooseAction* (e.g. one of table 1), *Learn* (e.g. a TD update of a model-based learning step) and *UpdateContBandit* (e.g. CAB1), the AdaTran will lead to different types of transfer learners. In particular, the experimental section will evaluate AdaTran(PPR), AdaTran(PPR-decay), and AdaTran(MGE).

Let us now show how CAB1 can be applied. Let t_i be the time at which the i^{th} episode begins. Let φ_t refer to the parameter chosen by CAB1 at time t. Then on the n first episodes, CAB1 will try to minimize the regret $\sum_{i=1}^{n} V_{t_i}(\varphi^*) - V_{t_I}(\varphi_t)$, which consists in finding the best transfer rate.

6 Experiments

In this section, we evaluate AdaTran on a standard benchmark for transfer learning: the grid-world problem [1,6]. The reason behind our choice of this learning task lies in its simplicity. As the state space is discrete, and no function approximation method is required, experimental results are more likely to evaluate the transfer approaches per se instead and not so much the whole machinery needed to make a complex learning problem tractable. In this learning task, an agent moves in a 25×25 two-dimensional maze. Each cell of this grid-world is a state and it may be surrounded by zero to four walls. Each cell has two coordinates (x, y), and the cell at the center of the grid has coordinates $(0, 0)$. At each time step, the agent can choose to move from its current position to one of the reachable contiguous north/east/west/south cell. If a wall lies in between, the action fails. Otherwise, the move succeeds with probability 90%, and with probability 10%, the agent is randomly placed on one of the reachable cells contiguous to the current cell. At the beginning of each episode, the agent is randomly and uniformly placed on the maze. As the agent reaches the goal state, it is given a reward of 1, and the episode is ended. All other rewards are null and the discount factor is arbitrarily set to $\gamma = 0,95$. At the beginning of each episode, the learning agent is place at the center of the grid, at position $(0, 0)$. The goal of the current task T is to reach the bottom right corner at $(12, -12)$. We generated three tasks T_1, T_2 and T_3 having three different goal positions (12,-10),(12,12),(-12,12).

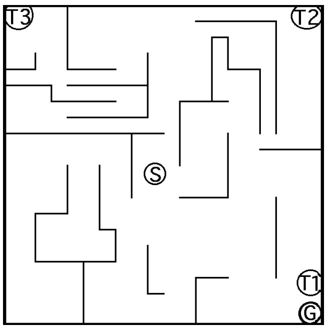

Fig. 1. grid-world

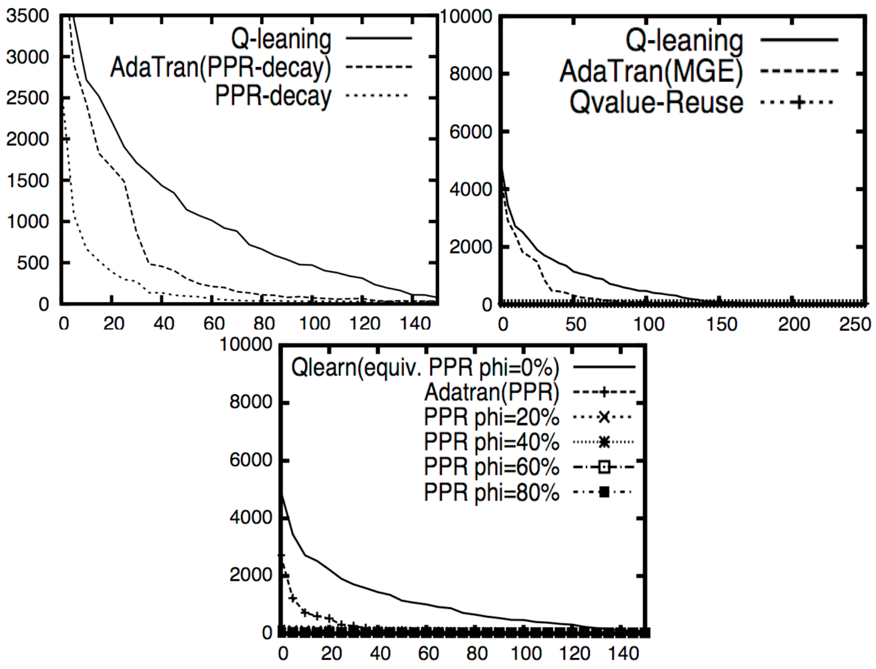

Fig. 2. Evaluation of AdaTran(PPR-decay), AdaTran(PPR-decay) and AdaTran(PPR-decay) against other learners, using T_1 as the transfer task

The optimal policies computed on each of these three tasks will serve as transfer knowledge to solve T. The goals of T_1 and T are very close to each other. Thus, transfer between both might be highly valuable. On the opposite, the goals in T_3 and T are very dissimilar to one another, and transfer is likely to be less valuable. In between, T and T_2 can be seen as "orthogonal" to each other: moving towards the goal of T_2 does not make the goal of T closer or farther. This will allow us to evaluate the robustness of AdaTran compared to other algorithms. Each of the following curves have been averaged over 100 runs. The x-axis represents the episodes, and the y-axis is the average episode length, given that episode are limited to 10000 steps.

Let us first consider the experiments with T_1 as a transfer task (fig. 2). As expected, the Q-learner performs worst, unable to exploit a task very similar to the current task. Also as expected, Qvalue-Reuse and "PPR phi" with $\varphi \geq 20\%$ perform extremely well, as they are biased to follow the transfer policy often. Even though the performance of AdaTran on this task are not as good as that of these biased transfer learners, compared to the Q-learner, AdaTran performs much better. For example, after 40 episodes, the average duration of a Q-learning episode is three times that of an AdaTran(PPR), AdaTran(PPR-decay) or AdaTran(MGE) episode.

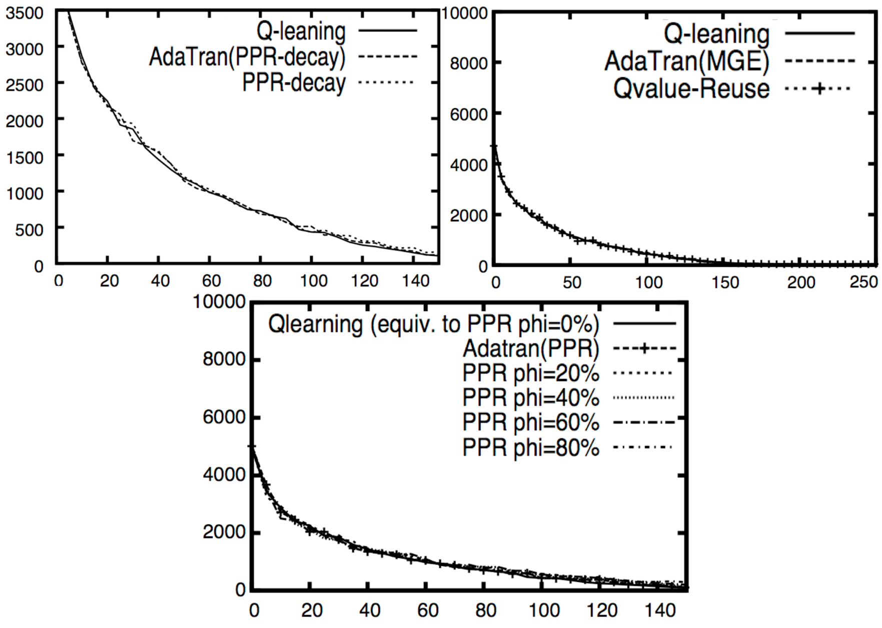

Fig. 3. Evaluation of AdaTran(PPR-decay), AdaTran(PPR-decay) and AdaTran(PPR-decay) against other learners, using T_2 as the transfer task

Concerning the transfer between T_2 and T, the "orthogonality" of both tasks appears in the curves (fig. 3). No transfer method achieves better or worse results than Q-learning.

Concerning the transfer from T_3 to T, which are very dissimilar tasks, we note that Q-learning always achieve best results, as it is never mislead by the transfer policy. Also note that AdaTran is quite close to Q-learning, which is the best it could do : the bandit in AdaTran must learns as fast as possible not to follow the transfer policy. It can be seen on figure 4 that PPR-decay is the best of the non-adaptive transfer learners here. In fact, PPR-decay follows the transfer policy only at the beginning of each episode, and is thus less penalized than others by bad transfer policies. Also, the Qvalue-reuse algorithm, which is one of the most used transfer method in reinforcement learning, performs very badly : as seen on figure 4, it requires approximately 150 episodes to "unlearn" the transfer policy. On the contrary, AdaTran(MGE) which also reuses the Q-values selects low values of φ very quickly, thus not using the transfer knowledge much. Lastly, note that PPR-phi are the worst non-adaptive transfer learners, as the episode length (y-axis) always reaches it maximal value. Clearly, AdaTran is shown to be robust to dissimilar tasks (T_3) unlike the other transfer methods studied here, and transfer successfully a high amount of knowledge on similar tasks (T_1).

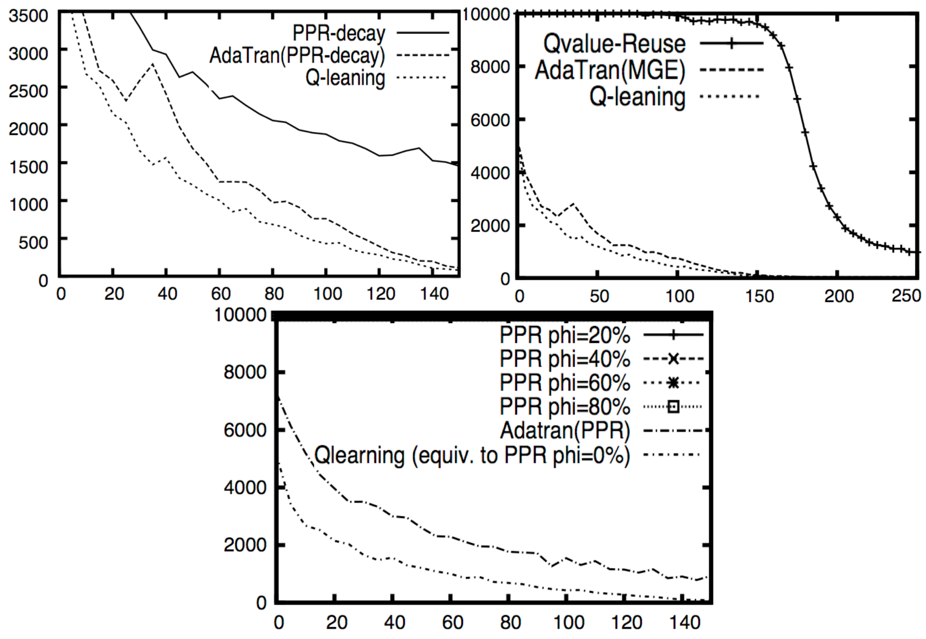

Fig. 4. Evaluation of AdaTran(PPR-decay), AdaTran(PPR-decay) and AdaTran(PPR-decay) against other learners, using T_3 as the transfer task

7 Conclusion and Future Work

In this paper, we have presented a new framework for explicitly optimizing the transfer rate in reinforcement learning. We have shown how this framework could be applied on two representative transfer learners to make the transfer rate auto-adaptable, namely the *probabilistic policy reuse* methods and *MGE* method, related to the well known Qvalue-reuse.

We have shown experimentally that AdaTran is robust to misleading transfer knowledge: when the transfer task is similar to the current task, AdaTran's performance will be close to non-adaptive transfer methods, but when the transfer task is very dissimilar to the current task, AdaTran will not spend a large amount of time forgetting transfer knowledge, unlike non-adaptive transfer learners.

There are four main directions for future work. First, the performance of these algorithms must be evaluated on real-world learning tasks. Some preliminary experiments on various grid-world tasks suggest that AdaTran makes transfer learners much more robust but it remains to be precisely characterized. Another important issue concerns the optimization criterion. In this paper, we proposed to optimize $V(\varphi)$. However, this implies trying to maximize the short term gain, which is heavily biased towards exploitation. Because of the close relation between our approach and gradient policy search methods, we are studying the criteria used in the latter [2].

Also, theoretical work relating the regret of the bandits to the expected performance of AdaTran remains to be done. Finally, we could design bandit algorithms taking the specificity of our problem into account. In fact, the adversarial setting may be excessive for our application, and the function $V(\varphi)$ probably has some properties (such as some kind of monotonicity through time) which may be exploited favorably.

References

1. Fernandez, F., Veloso, M.: Probabilistic Policy Reuse in a Reinforcement Learning Agent. In: The Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS (2006)
2. Kakade, S.: On The Sample Complexity of Reinforcement Learning. Phd Thesis, University College London (2003)
3. Taylor, M.E., Stone, P.: Behavior Transfer for Value-Function-Based Reinforcement Learning. In: The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, July 2005, pp. 53–59. ACM Press, New York (2005)
4. Taylor, M.E., Whiteson, S., Stone, P.: Transfer via Inter-Task Mappings in Policy Search Reinforcement Learning. In: The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2007 (May 2007)
5. Perkins, T.J., Precup, D.: Using options for knowledge transfer in reinforcement learning. Technical report, University of Massachusetts, Amherst (1999)
6. Price, B., Boutilier, C.: Accelerating Reinforcement Learning through Implicit Imitation. Journal of Artificial Intelligence Research 19, 569–629 (2003)
7. Kleinberg, R.: Nearly tight bounds for the continuum-armed bandit problem. In: Advances in Neural Information Processing Systems 17 (NIPS 2004), pp. 697–704 (2004)
8. Auer, P., Ortner, R., Szepesvri, C.: Improved Rates for the Stochastic Continuum-Armed Bandit Problem. In: Bshouty, N.H., Gentile, C. (eds.) COLT. LNCS, vol. 4539, pp. 454–468. Springer, Heidelberg (2007)
9. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: Gambling in a Rigged Casino: The adversarial multi-armed bandit problem. In: Proceedings of the 36th Annual Symposium on Foundations of Computer Science (1998)
10. Carroll, J.L., Peterson, T.S.: Fixed vs. Dynamic Sub-transfer in Reinforcement Learning. In: Arif Wani, M. (ed.) ICMLA 2002, Las Vegas Nevada, USA, June 24-27, 2002. CSREA Press (2002)
11. Carroll, J.L., Seppi, K.: Task Similarity Measures for Transfer in Reinforcement Learning Task Libraries. In: The 2005 International Joint Conference on Neural Networks, IJCNN 2005 (2005)
12. Carroll, J.L., Peterson, T.S., Owens, N.E.: Memory-guided Exploration in Reinforcement Learning. In: The 2001 International Joint Conference on Neural Networks, IJCNN 2001 (2001)
13. Price, B., Boutilier, C.: A Bayesian Approach to Imitation in Reinforcement Learning. In: IJCAI 2003, pp. 712–720 (2003)
14. Taylor, M.E., Stone, P., Liu, Y.: Transfer Learning via Inter-Task Mappings for Temporal Difference Learning. Journal of Machine Learning Research 8(1), 2125–2167 (2007)
15. Zhou, D.-X.: A note on derivatives of Bernstein polynomials. Journal of Approximation Theory 78(1), 147–150 (1994)
16. Lorentz, G.G.: Bernstein Polynomials. Chelsea, New York (1986)

17. Strehl, A.L., Li, L., Wiewiora, E., Langford, J., Littman, M.L.: PAC model-free reinforcement learning. In: ICML 2006: Proceedings of the 23rd international conference on Machine learning, pp. 881–888 (2006)
18. Madden, M.G., Howley, T.: Transfer of Experience Between Reinforcement Learning Environments with Progressive Difficulty. Artif. Intell. Rev. 21(3-4), 375–398 (2004)
19. Torrey, L., Walker, T., Shavlik, J., Maclin, R.: Knowledge Transfer Via Advice Taking. In: Proceedings of the Third International Conference on Knowledge Capture, KCAP 2005 (2005)
20. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the Twenty-first International Conference on Machine Learning (2004)
21. Konidaris, G.D.: Autonomous Shaping: Knowledge Transfer in Reinforcement Learning. In: Proceedings of the Twenty Third International Conference on Machine Learning (ICML 2006), Pittsburgh, PA (June 2006)
22. Kearns, M., Singh, S.: Finite-Sample Rates of Convergence for Q-Learning and Indirect Methods. In: Advances in Neural Information Processing Systems 11, pp. 996–1002. The MIT Press, Cambridge (1999)