Adaptive Probabilistic Policy Reuse

Yann Chevaleyre*, Aydano Machado Pamponet**

* Université Paris 13
yann.chevaleyre@lipn.univ-paris13.fr
** Universidade Federal de Alagoas
aydano.machado@gmail.com

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 (respectively nearly identical) to the current task, then the transfer process might slow down the learning (respectively 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?". In this paper, we show that this optimization problem is related to the continuum bandit problem. Based on this relation, we design an generic adaptive transfer method, which we evaluate on a grid-world task.

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 deals with this issue. Recently Strehl *et al.* [**?**] showed that nearly optimal strategies could be reached in as few as $\widetilde{O}(S \times A)$ time steps. However, with most real-world learning problems, the designer will face a huge state and action space, thus preventing any kind of exhaustive exploration.

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 utilized in different ways:

– By imitation: in particular, in a multi-agent environment, agents may observe traces of other agents and use this observation to learn the environment faster [**?**].

– By bootstrap: related tasks may have been previously learned by reinforcement [**?**] 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.
– By demonstration: a human tutor may provide some explicit knowledge. Other similar settings exist in the literature, among which are "advice taking" or "apprenticeship".

In this paper, we will focus on a simple version of the "bootstrap" transfer learning problem [**?**]: we will assume that a policy 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. Note that unlike in the "imitation" setting, in the boostrap setting no information about the transitions in the environment is available to the learning agent.

Given this knowledge, the learning agent faces a new dilemma: it has to balance between 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. However, if the available policy is unrelated to the current task, then exploring the environment by following the available policy could result in a slowdown of the learning process. This pathological behavior has been known in the transfer learning litterature as the *negative transfer* phenomenon [**?**]. Ideally, this amount should be tuned such that the transfer learner be *robust* with respect to the quality of the past policy : good policies should speed up the learner while bad ones should not slow it down significantly. Recently, a new approach has been proposed to solve this issue [**?**]. The main idea of this approach is to estimate the similarity between the two tasks, and then to use this estimate to parameterize 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* controlling 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 consisting of a wrapper around some standard transfer learning algorithm, and implementing a continuum-armed bandit algorithm.

We show that under some conditions, the regret of not having chosen from the beginning the optimal value of the transfer rate can be efficiently bounded. Experiments on a grid-world task validate our approach.

The paper is organized as follows. After some preliminaries, we introduce the continuous bandit problem and relate it to the optimization of the transfer rate. The following section introduces the generic transfer learner, which is then studied in deep. Finally a set of experiments assesses both the robustness and efficiency of our approach.

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 $V_M^\pi(s)$ denote the discounted value function for π in M from state s. More formally, $V_M^\pi(s) \triangleq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where $r_0, r_1, \ldots$ is the reward sequence obtained by following policy π from state s. Also, let $V_M^\pi \triangleq \mathbb{E}_{s \sim \mathcal{D}}\left[V_M^\pi(s)\right]$. 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 as in [**?**] 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.

3 Transfer Learners with Static Transfer Rates

In this section, we will present a state-of-the-art transfer learner, namely PPR (*Probabilistic Policy Reuse*, as well as PPR-decay, a variation on PPR [**?**]). These algorithms exhibit a parameter which controls the balance between the ongoing learned policy and $\bar{\pi}$. As in many transfer methods, PPR have been directly built 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*).

Let us see how PPR works. At each step, the PPR algorithm randomly chooses to follow the policy ϵ-greedy(π) or to follow $\bar{\pi}$, as depicted in Table **??**. Here, π refers to the policy induced 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 $0,95^t$. 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. Therefore, we introduce a variation on PPR-decay, namely PPR, in which φ is not decreased during the episode.

$ChooseAction(s_t, \bar{\pi}, \varphi)$	Name of transfer algorithm
$a_t = \begin{cases} \bar{\pi}(s_t) & w.proba.\varphi \times 0,95^t \\ \epsilon\text{-greedy}(\pi) & otherwise \end{cases}$	PPR-decay [**?**] (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)

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

Clearly, φ can be seen here as a parameter controling the *transfer rate*. 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 optimizing φ can be done during learning. In this section, φ was assumed to be a constant set before the learning process. In the next sections, we will show how φ can be optimized dynamically, and ajusted after each episode.

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 speed up 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 *continuum 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 this setting, at each time step t, a learner chooses a real number $X_t \in [0, 1]$ and receives a reward depending on the sequence $X_1 \ldots X_t$. The goal of the learner is to maximize the total sum of rewards, or to minimize the regret as stated formally below:

Definition 1 ***(The continuum armed-bandit problem)*** *Let $P(. \mid x, t)$ be an unknown distribution 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 randomly drawn from $P(. \mid X_t, t)$. Let $b_t(x) = \mathbb{E}[Y_t \mid X_t = x, t]$. The agent's goal is to minimize its expected regret $\mathbb{E}\left[\sum_t b_t(x^*) - \sum_t Y_t\right]$, given that $x^* = \sup_{x \in [0,1]} \sum_{t=1}^{n} b_t(x)$.*

This definition is a slight generalization of that found in [**?**]. Still in [**?**], Kleinberg designs an algorithm called *CAB1* solving this continuum armed bandit problem with the following guaratees:

Corollary 2 *If the function b_t is L-lipschitz (i.e., $\mid b_t(x) - b_t(x') \mid \leq L \mid x - x' \mid$ for all $x, x' \in [0,1]$), then using CAB1 yields an expected regret bounded by $O(Ln^{\frac{2}{3}} \log^{\frac{2}{3}} n)$.*

In the next section, we will describe AdaTran, an algorithm using CAB1 as a subroutine. Thus, corollary **??** will later be useful to derive a regret bound on AdaTran.

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 refered 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

```
Init()
t ← 0
φ ← φ_0
for each episode h do
    set the initial state s
    while (end of episode not reached) do
        a_t = ChooseAction(s_t, π̄, φ)
        Take action a_t, observe r_{t+1}, s_{t+1}
        Learn(s_t, a_t, r_{t+1}, s_{t+1})
        t ← t + 1
    end while
    φ ← UpdateContBandit(π̄, φ, ⟨r_1, r_2, ...⟩)
end for
```

Depending of the function used for *ChooseAction* (e.g. one of Table **??**), *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) and AdaTran(PPR-decay) Let us now show how the bound on the regret of CAB1 can be applied. Let t_i be the time at which the i^{th} episode begins. Suppose $V_t(\varphi)$ satisfies the L-lipschitz condition. Let φ_t refer to the parameter chosen by CAB1 at time t. Then on the n first episodes, we have $\sum_{i=1}^{n} V_{t_i}(\varphi^*) - V_{t_I}(\varphi_t) \leq O(Ln^{\frac{2}{3}} \log^{\frac{2}{3}} n)$ iff the following assumption holds:

Assumption 3 *At any given time step t, the value functions $V_t(\varphi)$ does not depend on previous actions in the MDP.*

Equivalently, we might assume that the sequence of functions $V_t(\varphi)$ is fixed in advance. Note that this type of assumption has been widely discussed in the multi-armed bandit setting. Also, there has been some attempts to overcome this assumption in the bandit literature, in particular **[?]**. These attempts usually rely on non standard definition of the regret and/or on strong assumptions on the type of non-stationarity of the environment, which does not suit our setting. Nevertheless, in our case, the assumption **??** seems reasonable since in most situations, choosing a sub-optimal exploration strategy for a given episode will not jeopardize the whole learning process.

We have seen in this section that optimizing φ with bounded regret may be possible, given that $V_t(\varphi)$ satisfies the Lipschitz condition. This remains to be proven. We will show this in detail for AdaTran(PPR).

6 Properties of the Value Function

In order to bound the regret of AdaTran, we now need to study the properties of the value function $V(\varphi)$ (the parameter t will be omitted). We will conduct this analysis in detail for AdaTran(PPR).

First, we will show that without any restrictions, $V(\varphi)$ cannot be optimized in the worst case. This is due to the fact that the function $V(\varphi)$ can be made arbitrarily close to any continuous function. To show this, we must first recall what Bernstein polynomials are. Without loss of generality, we will assume that the probability distribution $\mathcal{D}$ of starting states is equal to one on a given state s_0 and is null elsewhere.

Definition 4 *For any function f on $[0,1]$, the associated Bernstein polynomial is defined as follows: $B_n(f,x) \triangleq \sum_{k=0}^{n} f(\frac{k}{n}) b_{k,n}(x)$, where $b_{k,n}(x) \triangleq \binom{n}{k} x^k (1-x)^{n-k}$.*

Unfortunately, the set of all possible function $V(\varphi)$ includes the set of all Bernstein polynomials:

Lemma 5 *Let f be any continuous function of $[0,1]$ and $d \in \mathbb{N}$. Then there exist an MDP such that $V(\varphi) = B_d(f,\varphi)$*

Proof. Let A^d be a deterministic MDP having the structure of a binary tree of depth d. Let the root state be s_0. At each state (node in the tree), A^d allows two actions *left* and *right* leading respectively to the left and right child states. Let the rewards of all state-actions be null, except those between depth $d-1$ and d. Let us allocate rewards to the 2^d states-actions pairs at depth $d-1$ of the tree in the following way: if a state-action pair can be reached from the root by l steps left and $d-l$ steps right (in any order), then its reward must be $\gamma^{1-d} \times f(\frac{l}{d})$. Suppose on each state s, the standard policy is $\pi(s) = right$, whereas the transfer policy is $\bar{\pi}(s) = left$. Thus, a learner exploring this tree and starting from the root node would walk down the tree, choosing randomly left and right branches with probability φ (respectively $1-\varphi$), and collecting a reward $\gamma^{1-d} \times f(\frac{i}{d})$ at the bottom of the tree. Let $V^{A^d}(\varphi) = \mathbb{E}[r_1 + \gamma r_2 + \ldots]$ be the value of the root node, parameterized by φ. Clearly, the probability that an agent chooses l times the *left* action and $d-l$ times the *right* action in a given order is $\varphi^l(1-\varphi)^{d-l}$. Thus, the probability of choosing l times *left* in any order is $b_{l,d}(\varphi) = \binom{d}{l}\varphi^l(1-\varphi)^{d-l}$. Therefore, we have $V^{A^d}(\varphi) = \frac{1}{\gamma^{d-1}} \sum_{l=0}^{d} \gamma^{1-d} \times f(\frac{l}{d}) \times b_{l,d}(\varphi)$.

Recall now that the Weierstrass theorem states that for any continuous function f on $[0,1]$, $B_n(f,x)$ converges uniformly to $f(x)$ as $n \to \infty$. An immediate corollary is:

Corollary 6 *For any continuous function f on $[0,1]$, for any $\epsilon > 0$, there exists a deterministic MDP such that $|V(\varphi) - f(\varphi)| < \epsilon$ for all φ.*

This has important implications in our setting: this corollary tells us that $V(\varphi)$ can be arbitrarily close to any continuous function, provided the appropriate MDP. Thus, optimizing $V(\varphi)$ without further restrictions is hopeless.

However, by upper-bounding the rewards, we will now show that $V(\varphi)$ finally satisfies the lipschitz condition. Let us first bound the derivative of Bernstein polynomials.

Lemma 7 *Let f be a real-valued function on $\{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}$. Then we have $\sup_{x\in[0,1]} | \frac{d}{dx} B_n(f,x) | \leq 2n \sup | f(x) |$.*

Due to space constraints, the proof of this technical lemma is omitted.

Applying this lemma on the tree MDPs used in lemma **??**, we get the bound $| \frac{d}{dx} V^{A_d}(\varphi) | \leq 2d\gamma^{d-1} r_{max}$, given that all rewards are bounded by r_{max}. Finally, we generalize this result (again, the proof is omitted).

Proposition 8 *For any MDP M in which rewards are bounded by r_{max} , any policies π and $\bar{\pi}$, and a starting state s_0, we have $\left| \frac{d}{d\varphi} V(\varphi) \right| \leq \frac{2r_{max}}{(1-\gamma)^2}$.*

Finally, combining the above result with the regret bound of corollary **??**, we can now show the following:

Corollary 9 *The cumulative regret per episode of AdaTran(PPR) is $O\left(\frac{r_{max} h^{\frac{2}{3}} \log^{\frac{2}{3}} h}{(1-\gamma)^2}\right)$, where h is the episode number.*

Again, note that the regret does not depend on the number of states, so the MDP might be huge here.

7 Experiments

In this section, we evaluate AdaTran on a standard benchmark for transfer learning: the grid-world problem [**?**,**?**]. The reason behind our choice of this learning task lies in its simplicity. 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. 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 (the exit of the maze), 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 $\gamma = 0,95$.

The goal of the current task (the exit) is to reach the bottom right corner. We generated two other tasks based on the exact same maze but different goals. The first task refered to as the "similar task" has its goal located two cells away from the bottom right corner, whereas the second task, refered to as the "dissimilar task", has its goal located on the opposite corner (upper left corner).

The optimal policies computed on each of these two tasks will serve as transfer knowledge to solve the current task. The goals of the "similar task" and the current task are very close to each other. Thus, transfer between both might be highly valuable. On the opposite, the goals of the "dissimilar task" and that of the current task are very dissimilar to one another, and transfer is likely to be less valuable.

AdaTran is compared to other algorithms on figure **??** and **??**. Each of these 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.

When transferring from the "similar task", PPR performs extremely well, and AdaTran performs much better than Q-learning, as it quickly finds that tasks are similar, but a bit worse that PPR as expected. When transferring from the "dissimilar task", PPR with various levels of φ perform the worse: all episodes reach 10000 steps in average, and the exit of the maze is nearly never found. The best learner here is Q-learning, which ignores the transfer policy. In between both lies AdaTran, which quickly detects that the transfer policy should not be trusted.

Clearly, AdaTran is shown to be robust to dissimilar tasks unlike most other transfer methods, and it is shown to transfer successfully a high amount of knowledge on similar tasks.

8 Conclusion

In this paper, we have presented a new framework for explicitly optimizing the transfer rate in reinforcement learning. We have shown how this frame-

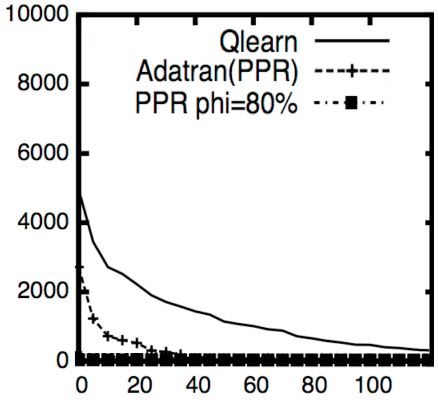

Fig. 1. Similar transfer tasks

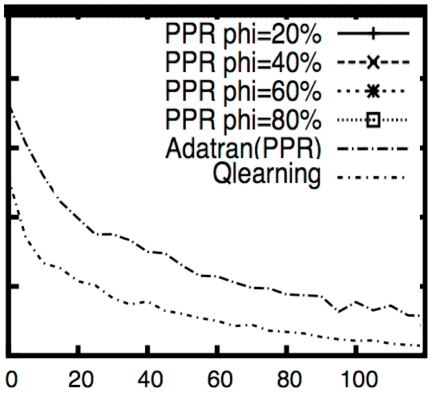

Fig. 2. Dissimilar transfer tasks

work could be applied on a well known transfer learner to make the transfer rate auto-adaptable, namely the *PPR* method. Moreover, by bounding the maximum reward, we showed that the average regret converged towards zero.

References

1. Fernandez, F., and 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. Price, B. and Boutilier, C. (2003) "*Accelerating Reinforcement Learning through Implicit Imitation*", Journal of Artificial Intelligence Research, Volume 19, pages 569-629.
3. Robert Kleinberg. "*Nearly tight bounds for the continuum-armed bandit problem*". In Advances in Neural Information Processing Systems 17 (NIPS 2004), pp. 697-704.
4. James L. Carroll, Kevin Seppi "*Task Similarity Measures for Transfer in Reinforcement Learning Task Libraries*" in The 2005 Int Joint Conference on Neural Networks, (IJCNN 2005).
5. Strehl, A.L., Li, L., Wiewiora, E., Langford, J., and Littman, M.L. "*PAC model-free reinforcement learning*", in ICML-06, p.881–888 (2006).
6. Pucci de Farias, D., Megiddo, N. *Combining expert advice in reactive environments.* Journal of the ACM, 53(5). pp762-799. 2006.
7. Rosenstein, M. T., Marx, Z., Kaelbling, L. P., Dietterich, T. G. *To transfer or not to transfer.* NIPS 2005 Workshop on Transfer Learning, Whistler, BC.
8. Lazaric, A., Restelli, M. and Bonarini A. *Transfer of Samples in Batch Reinforcement Learning.* Proceedings of the Twenty-Fifth International Conference on Machine Learning, Helsinki, Finland, June 5-9, 2008.