Transfer Reinforcement Learning with Shared Dynamics

Romain Laroche, Merwan Barlier

Transfer Reinforcement Learning with Shared Dynamics

Romain Laroche

Orange Labs at Châtillon, France

Maluuba at Montréal, Canada

romain.laroche@m4x.org

Merwan Barlier

Orange Labs at Châtillon, France

Univ. Lille 1, UMR 9189 CRIStAL, France

merwan.barlier@orange.com

Abstract

This article addresses a particular Transfer Reinforcement Learning (RL) problem: when dynamics do not change from one task to another, and only the reward function does. Our method relies on two ideas, the first one is that transition samples obtained from a task can be reused to learn on any other task: an immediate reward estimator is learnt in a supervised fashion and for each sample, the reward entry is changed by its reward estimate. The second idea consists in adopting the *optimism in the face of uncertainty* principle and to use upper bound reward estimates. Our method is tested on a navigation task, under four Transfer RL experimental settings: with a known reward function, with strong and weak expert knowledge on the reward function, and with a completely unknown reward function. It is also evaluated in a Multi-Task RL experiment and compared with the state-of-the-art algorithms. Results reveal that this method constitutes a major improvement for transfer/multi-task problems that share dynamics.

1 Introduction

Reinforcement Learning (RL, (Sutton and Barto 1998)) is a framework for optimising an agent behaviour in an environment. It is generally formalised as a Markov Decision Process (MDP): $\langle \mathcal{S}, \mathcal{A}, R, P, \gamma \rangle$ where $\mathcal{S}$ the state space, and $\mathcal{A}$ the action space are known by the agent. $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, the Markovian transition stochastic function, defines the unknown dynamics of the environment. $R : \mathcal{S} \to \mathbb{R}$, the immediate reward stochastic function, defines the goal(s)[1]. In some settings such as dialogue systems (Laroche et al. 2009; Lemon and Pietquin 2012) or board games (Tesauro 1995; Silver et al. 2016), R can be inferred directly from the state by the agent, and in some others such as in robotics and in Atari games (Mnih et al. 2013; 2015), R is generally unknown. Finally, $\gamma \in [0, 1)$ the discount factor is a parameter given to the RL optimisation algorithm favouring short-term rewards. As a consequence, the RL problem consists in (directly or indirectly) discovering P, sometimes R, and planning.

Even when R is unknown, R is often simpler to learn than P: its definition is less combinatorial, R is generally sparse, only a mean estimation is required, R tends to be less stochastic than P, and finally it is frequently possible for the designer to inject expert knowledge: for instance, an adequate state space representation for R, the uniqueness of the state with a positive reward, its determinism or stochastic property, and/or the existence of R bounds: R_{min} and R_{max}.

Discovering (directly or indirectly) P and R requires collecting trajectories. In real world problems, trajectory collection is resource consuming (time, money), and Transfer Learning for RL (Taylor and Stone 2009; Lazaric 2012), through reuse of knowledge acquired from similar tasks, has proven useful in many RL domains: Atari games (Romoff, Bengio, and Pineau 2016), robotics (Taylor, Stone, and Liu 2007), or dialogue (Genevay and Laroche 2016). In this article, we address the problem of Transfer Reinforcement Learning with Shared Dynamics (TRLSD), *i.e.* the transfer problem when P is constant over tasks $\tau \in \mathcal{T}$, which thus only differ from each other by their reward functions R_τ. We include the Multi-Task RL variation of this problem under this denomination, *i.e.* when learning is made in parallel on several tasks. This family of problems may be encountered for instance in robotics, where the robot agent has to understand the complex shared environment dynamics in order to perform high level tasks that rely on this understanding.

In this article, we advocate that experience gathered on a task can be indirectly and directly reused on another task and that transfer can be made at the transition sample level. Additionally, the *optimism in the face of uncertainty* principle allows to guide the exploration efficiently on a new task, thanks to the dynamics knowledge transferred from the other tasks. The combination of those two principles allows us to define a general algorithm for TRLSD, on a continuous state space, enabling the injection of task related expert knowledge.

Section 2 offers an overview of the known studies related to TRLSD, and introduces the principles of transition sample sharing between tasks. Then, Section 3 recalls the *optimism in the face of uncertainty* principle, explains how to apply it to our setting, and explores different ways of computing this optimism, inspired from the UCRL algorithm. Finally, Section 4 presents various experiments illustrating and demonstrating the functioning of our algorithms in Transfer RL and Multi-Task RL experiments. The experimental results demonstrate the significant improvement brought by our approach,

[1] In this article, reward functions are defined on state representation $\mathcal{S}$, but all the results can be straightforwardly transposed to rewards received after performing an action in a given state, *i.e.* to reward function defined on $\mathcal{S} \times \mathcal{A}$.

in comparison with the state-of-the-art algorithms in TRLSD.

2 Background and Principle

To the authors knowledge at the time they write this article, only two recent works were dedicated to TRLSD. First, (Barreto et al. 2016) present the framework as a kind of hierarchical reinforcement learning, where composite and compoundable subtasks are discovered by generalisation over tasks. In order to do so, tasks share the successor features[2] of their policies, which are invariant from one task to another. Their decomposition of the reward function from the dynamics is unfortunately restricted to policies characterising the successor features. Additionally, the theoretical analysis depends on R_τ similarities, which is not an assumption that is made in this article. Second, (Borsa, Graepel, and Shawe-Taylor 2016) address the same problem in a Multi-Task RL setting by sharing the value-function representation: they build a transition sample set for all tasks and apply generalised versions of Fitted-Q iteration (Ernst, Geurts, and Wehenkel 2005) and Fitted Policy Iteration (Antos, Szepesvári, and Munos 2007) learning on those transitions as a whole. The generalisation amongst tasks occurs in the regularisation used in the supervised learning step of Fitted-Q iteration (and policy iteration/evaluation).

Instead of sharing successor features or value-function representations, we argue that transition samples can be shared across tasks. A transition sample (or sample in short) is classically defined as a 4-tuple $\xi = \langle s, a, r, s' \rangle$, where s is the state at the beginning of the transition, a is the action performed, r is the reward immediately received, and s' is the state reached at the end of the transition. For Transfer and Multi-Task RL, it is enhanced with task τ to keep in memory which task generated the sample: $\xi_\tau = \langle \tau, s, a, r, s' \rangle$. Formulated in another way, s is drawn according to a distribution depending on the behavioural policy π^τ, a according to the behavioural policy $\pi^\tau(s)$, r according to the reward function $R_\tau(s)$ of task τ and s' according to the shared dynamics $P(s, a)$.

As a consequence, with a transition sample set for all tasks $\Xi = \bigcup_{\tau \in \mathcal{T}} \Xi_\tau$, one can independently learn $\hat{P}$, an estimate of P, in a supervised learning way. In the same manner, with the sample set constituted exclusively of task τ transitions Ξ_τ, one can independently learn $\hat{R}_\tau$, an estimate of the reward function expected value $\mathbb{E}[R_\tau]$, in a supervised learning way. This is what model-based RL does (Moore and Atkeson 1993; Brafman and Tennenholtz 2002; Kearns and Singh 2002). In other words, if transition sample ξ_τ was generated on task τ, and if task τ' shares the dynamics P with τ, then ξ_τ can be used for learning the dynamics model of task τ'. The adaptation to non-stationary reward functions has been an argument in favour of model-based RL for twenty years. In particular, (Atkeson and Santamaria 1997) applies it successfully on a task transfer with shared dynamics and similar reward functions on the inverted pendulum problem.

Nevertheless, this approach has never been theorised nor applied to Transfer or Multi-Task RL. We also advocate that learning the dynamics model P is not necessary and that efficient policies can be learnt in a direct fashion: given a target task τ, any transition sample $\xi_{\tau'} = \langle \tau', s, a, r, s' \rangle$ from any other task $\tau' \neq \tau$ can be projected on task τ, just by modifying the immediate reward r with $\hat{R}_\tau(s)$, the estimate of the reward function expected value $\mathbb{E}[R_\tau]$: $\psi_{\hat{R}_\tau}(\xi_{\tau'}) = \langle \tau, s, a, \hat{R}_\tau(s), s' \rangle$. The approach consists thus in translating the transition sample set Ξ into $\hat{R}_\tau$ estimate: $\Psi_{\hat{R}_\tau}(\Xi) = \{\psi_{\hat{R}_\tau}(\xi_{\tau'})\}_{\xi_{\tau'} \in \Xi}$, and then in using any off-policy RL algorithm to learn policy π_τ on $\Psi_{\hat{R}_\tau}(\Xi)$. The off-policy characteristic is critical in order to remove the bias originated from the behavioural policies π^τ controlling the transition sample set Ξ generation. In our experiments, we will use Fitted-Q Iteration (Ernst, Geurts, and Wehenkel 2005). The following subsection recalls the basics.

Fitted-Q Iteration

The goal for any reinforcement learning algorithm is to find a policy π^* which yields optimal expected returns, *i.e.* which maximises the following Q-function:

$$Q^*(s_t, a_t) = Q^{\pi^*}(s_t, a_t) = \operatorname*{argmax}_\pi \mathbb{E}^\pi_{s_t, a_t} \left[\sum_{t' \geq 0} \gamma^{t'} r_{t'+t} \right].$$

The optimal Q-function Q^* is known to verify Bellman's equation:

$$Q^*(s, a) = \mathbb{E}\left[R(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right] \quad (1)$$

$$\Leftrightarrow \quad Q^* = T^* Q^*. \quad (2)$$

The optimal Q-function is thus the fixed point of Bellman's operator T^*. Since $\gamma < 1$, it is a contraction, and Banach's theorem ensures its uniqueness. Hence, the optimal Q-function can be obtained by iteratively applying Bellman's operator to some initial Q-function. This procedure is called Value Iteration.

When the state space is continuous (or very large) it is impossible to use Value-Iteration as such. The Q-function must be parametrised. A popular choice is the linear parametrisation of the Q-function (Sutton and Barto 1998; Chandramohan, Geist, and Pietquin 2010):

$$Q(s, a) = \theta^\top \Phi(s, a), \quad (3)$$

where $\Phi(s, a) = \{\mathbb{1}_{a=a'} \Phi(s)\}_{a' \in \mathcal{A}}$ is the feature vector for linear state representation, $\mathbb{1}_{a=a'}$ is the indicator function, $\Phi(s)$ are the features of state s, and $\theta = \{\theta_a\}_{a \in \mathcal{A}}$ is the parameter vector that has to be learnt. Each element of θ_a represents the influence of the corresponding feature in the Q-function.

The inference problem can be solved by alternately applying Bellman's operator and projecting the result back onto the space of linear functions, and iterating these two steps until convergence.

$$\theta^{(i+1)} = (X^\top X)^{-1} X^\top y^{(i)}, \quad (4)$$

where, for a transition sample set $\Xi = \{\xi_j\}_{j \in [\![1, |\Xi|]\!]} = \{\langle s_j, a_j, s'_j, r_j \rangle\}_{j \in [\![1, |\Xi|]\!]}$, X is the observation matrix, which lines are the s_j feature vectors: $(X)_j = \Phi(s_j, a_j)$, and $y^{(i)}$ is a vector with elements $(y^{(i)})_j = r_j + \gamma \max_{a'} \theta^{(i)} \Phi(s'_j, a')$.

[2] A successor feature, *a.k.a.* feature expectation in (Ng and Russell 2000), is a vector summarising the dynamics of the Markov chain induced by a fixed policy in a given environment.

Data: Ξ: transition sample set on various tasks
Data: $\Xi_\tau \subseteq \Xi$: transition sample set on task τ
Learn on Ξ_τ an immediate reward proxy: $\tilde{R}_\tau$;
Cast sample set Ξ on task τ: $\Psi_{\tilde{R}_\tau}(\Xi)$;
Learn on $\Psi_{\tilde{R}_\tau}(\Xi)$ a policy for task τ: π_τ;

Algorithm 1: Transition reuse algorithm

3 Optimism in the Face of Uncertainty

The batch learning presented in last section proves to be inefficient in online learning: using an estimate $\hat{R}_\tau$ of R_τ is inefficient in early stages, when only a few samples have been collected on task τ and reward has never been observed in most states, because the algorithm cannot decide if it should exploit or explore further. We generalise our approach to a reward proxy $\tilde{R}_\tau$ in Algorithm 1.

In order to guide the exploration, we adopt the well-known *optimism in the face of uncertainty* heuristic, which can be found in Prioritized Sweeping (Moore and Atkeson 1993), R-MAX (Brafman and Tennenholtz 2002), UCRL (Auer and Ortner 2007), and VIME (Houthooft et al. 2016). In the optimistic view, $\tilde{R}_\tau$ is the most favourable plausible reward function. Only UCRL and VIME use an implicit representation of the exploration mechanism that is embedded into the transition and the reward functions. The way UCRL separates the dynamics uncertainty from the immediate rewards uncertainty makes it more convenient to implement and the following of the article is developed with UCRL solution, but any other optimism-based algorithm could have been considered in its place. (Lopes et al. 2012) and (Osband, Roy, and Wen 2016) are also proposing interesting alternative options for guiding the exploration.

Upper Confidence Reinforcement Learning

UCRL algorithm keeps track of statistics for rewards and transitions: the number of times $N(s,a)$ action a in state s has been performed, the average immediate reward $\hat{r}(s)$ in state s, and the observed probability $\hat{p}(s,a,s')$ of reaching state s' after performing action a in state s. Those statistics are only an estimate of their true values. However, confidence intervals may be used to define a set $\mathcal{M}$ of plausible MDPs in which the true MDP belongs with high probability. As said in last paragraph, UCRL adopts the *optimism in the face of uncertainty* principle over $\mathcal{M}$ and follows one of the policies that maximise the expected return in the most favourable MDP(s) in $\mathcal{M}$. The main idea behind the optimism exploration is the fact that mistakes will be eventually observed and knowledge of not doing it again will be acquired and realised through a narrowing of the confidence interval.

One UCRL practical problem is the need for searching the optimal policy inside $\mathcal{M}$ (Szepesvári 2010), which is complex and computer time consuming. In our case, we can however consider that $\hat{P}$ is precise enough in comparison with $\hat{R}_\tau$ and that dynamics uncertainty should not guide the exploration. Therefore, the optimal policy on $\mathcal{M}$ is necessarily the optimal policy of the MDP with the highest reward function inside the confidence bounds, *i.e.* $\tilde{R}_\tau(s)$ defined by the following equation:

$$\tilde{R}_\tau(s) = \hat{R}_\tau(s) + CI_\tau(s), \tag{5}$$

where $CI_\tau(s)$ is the confidence interval of reward estimate $\hat{R}_\tau$ in state s. Afterwards, the optimal policy can be directly learnt on data $\Psi_{\tilde{R}_\tau}(\Xi)$ with Fitted-Q Iteration.

Another UCRL limitation is that it does not accommodate continuous state representations. If a continuous state representation $\Phi(\mathcal{S}) = \mathbb{R}^d$ needs to be used for estimating $\tilde{R}_\tau$, UCCRL (Ortner and Ryabko 2012) or UCCRL-KD (Lakshmanan, Ortner, and Ryabko 2015) have been considered. But, they suffer from two heavy drawbacks: they do not define any method for computing the optimistic plausible MDP and the respective optimal policy; and they rely on the definition of a discretisation of the state representation space, which is exponential on its dimension, therefore intractable in most case, and in our experimental setting more particularly.

Confidence intervals for continuous state space

We decided to follow the same idea and compute confidence intervals around the regression $\hat{R}_\tau$. The natural way of computing such confidence intervals would be to use confidence bands. Holm–Bonferroni method (Holm 1979) consists in defining a band of constant diameter around the learnt function such that the probability of the true function to be outside of this band is controlled to be sufficiently low. Unfortunately, this method does not take into account the variability of confidence in different parts of the space, and this variability is exactly the information we are looking for. Similarly, Scheffé's method (Scheffe 1999) studies the contrasts between the variables, and although its uncertainty bound is expressed in function of the state, it is only dependent on its distance to the sampling mean, not on the points density near the point of interest. Both methods are indeed confidence measures for the regression, not for the individual points.

Instead, we propose to use the density of neighbours in Ξ_τ around the current state to estimate its confidence interval. In order to have a neighbouring definition, one needs a similarity measure $\mathscr{S}(s_1, s_2)$ that equals 1 when $s_1 = s_2$ and tends towards 0 when s_1 and s_2 get infinitely far from each other. In this article, we use the Gaussian similarity measure relying on the Euclidean distance in the state space $\mathcal{S}$ or its linear representation $\Phi(\mathcal{S})$:

$$\mathscr{S}_\mathcal{S}(s_1, s_2) = e^{-\|s_1 - s_2\|^2 / 2\sigma^2}, \tag{6}$$

$$\mathscr{S}_\Phi(s_1, s_2) = e^{-\|\Phi(s_1) - \Phi(s_2)\|^2 / 2\sigma^2}, \tag{7}$$

where parameter σ denotes the distance sensitivity of the similarity. Once a similarity measure $\mathscr{S}(s_1, s_2)$ has been chosen, the next step consists in computing the neighbouring weight in a sample set Ξ_τ around a state s:

$$W_\tau(s) = \sum_{\langle \tau, s_j, a_j, s'_j, r_j \rangle \in \Xi_\tau} \mathscr{S}(s, s_j). \tag{8}$$

Similarly to UCB, UCRL and UCCRL upper confidence, the confidence interval can be obtained thanks to the neighbouring weight with the following equation:

$$CI_\tau(s) = \kappa \sqrt{\frac{\log(|\Xi_\tau|)}{W_\tau(s)}}, \tag{9}$$

where parameter κ denotes the *optimism* of the agent.

This confidence interval definition shows several strengths: contrarily to Holm-Bonferroni and Scheffé's methods, it is locally defined, and it works with any regression method computing $\hat{R}_\tau$. But it also has two weaknesses: it relies on two parameters σ and κ, and it does not take into account the level of agreement between the neighbours. UCRL and UCCRL set κ values for which theoretical bounds are proven. Experiments usually show that lower κ values are generally more efficient in practice. The empirical sensibility to κ and σ values is evaluated in our experiments. The definition of a better confidence interval is left for further studies.

The estimates $\hat{R}_\tau$ of the rewards can be computed with any regression algorithm, from linear regression to neural nets. In our experiments, in order to limit computations, we use linear regression with a Tikhonov regularisation and $\lambda = 1$ (Tikhonov 1963), which, in addition to standard regularisation benefits, enables to find regression parameters before reaching a number of examples equal or higher to the dimension d of $\Phi(\mathcal{S})$. As in UCRL, the current optimal policy is updated as soon as the confidence interval in some encountered state has been divided by 2 since the last update.

Using expert knowledge to cast the reward function into a simpler discrete state space

Since, in our setting, the optimism principle is only used for the reward confidence interval, we can dissociate the continuous linear parametrisation $\Phi(\mathcal{S})$ used for learning the optimal policy and a simpler[3] discrete representation for estimating $\hat{R}_\tau$. If $\hat{R}_\tau$ is estimated by averaging on this discrete representation, its confidence interval CI_τ might be computed in the same way as UCB or UCRL. Confidence intervals are defined in the following way:

$$CI_\tau(s) = \kappa \sqrt{\frac{\log(|\Xi_\tau|)}{N_\tau(s)}}, \tag{10}$$

where parameter κ denotes the *optimism* of the learning agent, and $N_\tau(s)$ is the number of visits of the learning agent in state s under task τ, and therefore the number of received rewards in this state.

The possibility to use a different state representation for estimating P and $\tilde{R}_\tau$ is a useful property since it enables to include expert knowledge on the tasks: structure, bounds, or priors on R_τ, which may drastically speed up the learning convergence in practice. In particular, the possibility to use priors is very interesting when the task distribution is known or learnt from previously encountered tasks.

4 Experiments and results

We consider a TRLSD navigation toy problem, where the agent navigates in a 2D maze world as depicted by Figures 1-4. The state representation $\mathcal{S}$ is the agent's real-valued coordinates $s_t = \{x_t, y_t\} \in (0, 5)^2$, and the set of 25 features $\Phi(s_t)$ is defined with 5*5 Gaussian radial basis functions placed at $s_{ij} = \{i - 0.5, j - 0.5\}$ for $i, j \in [\![1, 5]\!]$, computed with the $\mathscr{S}_\mathcal{S}$ similarity with $\sigma = 0.2$:

$$\phi_{ij}(s_t) = \mathscr{S}_\mathcal{S}(s_t, s_{ij}). \tag{11}$$

At each time step, the agent selects an action among four possibilities: $\mathcal{A} = \{\text{NORTH}, \text{WEST}, \text{SOUTH}, \text{EAST}\}$. P is defined as follows for the NORTH action: $x_{t+1} \sim x_t + \mathcal{N}(0, 0.25)$ and $y_{t+1} \sim y_t - 1 + \mathcal{N}(0, 0.5)$, where $\mathcal{N}(\mu, \nu)$ is the Gaussian distribution with centre μ and standard deviation ν. This works similarly with the other three directions. Then, wall and out-of-grid events intervene in order to respect the dynamics of the maze. When a wall is encountered, a *rebound* is drawn according to $\mathcal{U}(0.1, 0.5)$ is applied, where $\mathcal{U}(\cdot, \cdot)$ denotes the uniform distribution.

The stochastic reward function $R_{\tau_{ij}}$ is corrupted with a strong noise and is defined for each task τ_{ij} with $i, j \in [\![1, 5]\!]$ as follows:

$$R_{\tau_{ij}}(s_t) \sim \begin{cases} 1 + \mathcal{N}(0, 1) \text{ if } \begin{cases} i = \lceil x_t \rceil, \\ j = \lceil y_t \rceil, \end{cases} \\ \mathcal{N}(0, 1) \text{ otherwise.} \end{cases} \tag{12}$$

Transfer Reinforcement Learning experiments

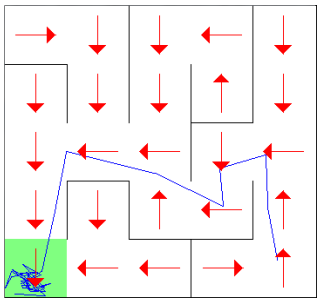
Transfer Learning experiments unfold as follows. First, 25000 transitions are generated with a random policy and stored in Ξ. After each transition, the trajectory has a 2% chance to be terminated, then, the next state is reset to a uniformly drawn state. Those transitions are considered enough to construct a perfect representation of P. Whatever the reward function has been used during this data collection, the reward information is discarded, such that any target task τ would be regarded as new and undiscovered during the transfer phase.

In the first experiment, Task τ is assumed to be known (*i.e.* the reward function R_τ is known). It is the case when the agent is instructed to perform a specific task. In this setting, the reward estimator $\hat{R}_\tau$ equals R_τ and the uncertainty is null. Therefore, $\tilde{R}_\tau = R_\tau$. An optimal policy can directly be computed from $\Psi_{R_\tau}(\Xi)$ with Fitted-Q Iteration. Figure 1 shows that the agent immediately follows a rational policy and heads towards the reward slot.

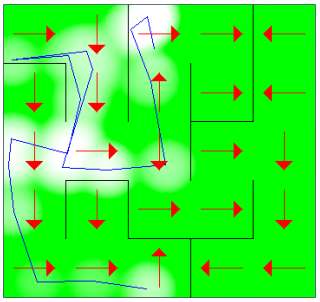
In the other transfer learning learning experiments, where R_τ is partially unknown, the transfer learning phase consists in the gathering of 1,000 transitions on the target task.

In the second experiment, 25 tasks are considered. Two transition reuse settings are compared: the *continuous* and the *discrete* settings. In the *continuous* setting, nothing is assumed to be known about the task τ: state representation $\Phi(\mathcal{S})$ is used to learn R_τ. As explained in the confidence interval section, we use linear regression with Tikhonov regularisation and $\lambda = 1$. The computation of the confidence interval $CI_\tau(s)$ is made according to Equation 9, which is dependent on two parameters: σ and κ. Figures 2a, 2b, and 2c show the exploration-exploitation trade-off of typical trajectories in the *continuous* setting. Initially, the agent is attracted by unknown areas, but as the task is discovered, it exploits more and more its knowledge, and exploration is eventually limited to less visited places when the agents passes by them. In the *discrete* setting, it is assumed to be known that task

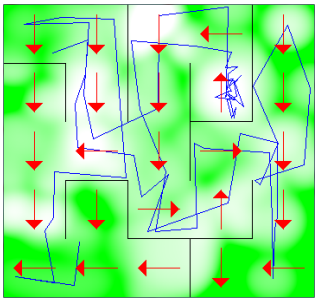
[3] In the sense, that it can be inferred from $\Phi(S)$.



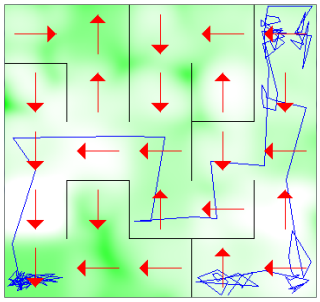
(1) Reward function is known: 20 first transitions.



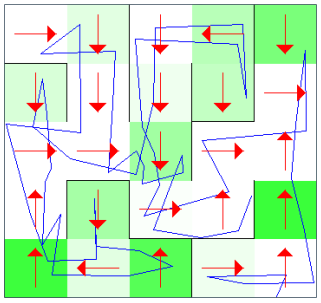
(2a) Reward function is unknown: 15 first transitions.



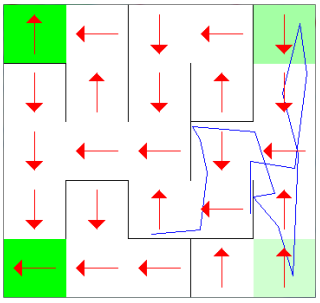
(2b) Reward function is unknown: after 150 transitions.



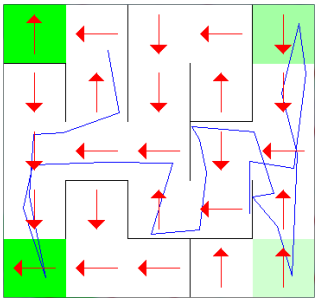
(2c) Reward function is unknown: after 650 transitions.



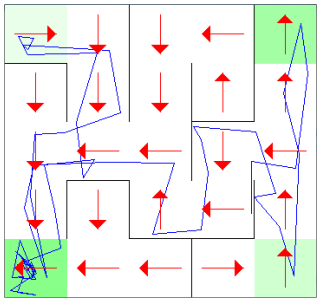
(3) Rewards are known to be defined by cell: 50 first transitions.



(4a) Rewards are known to be at the corners: 15 first transitions.



(4b) Rewards are known to be at the corners: 30 first transitions.



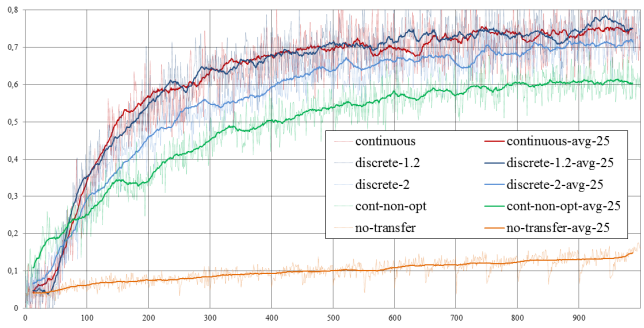
(4c) Rewards are known to be at the corners: 60 first transitions.

Figures 1-4: Trajectories directed by policies learnt from 25,000 transition samples with various reward function representations. Some of the policies are motivated by the knowledge of the estimated reward function and some are motivated by curiosity and optimism. The walls are displayed in black and are constant over tasks, the reward function representation are displayed in green, the current trajectory is the broken line in blue and the policy at the centre of cells is coarsely exhibited with the red arrows. In all these screenshots, the real reward function is the one rewarding the bottom left cell.

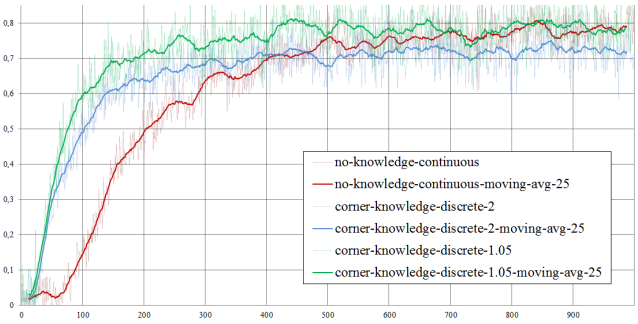
τ is defined as follows: $R_\tau : [\![1,5]\!]^2 \to \mathbb{R}$ and the reward function is cast into this simpler discrete space. The expert knowledge consists in the model of estimator $\hat{R}_\tau$ and since it is discrete, Equation 10, which is dependent on the single parameter κ, can be used for computing the confidence intervals. Figure 3 shows the first trajectory in the *discrete* setting, exploring methodically the 25 slots.

The results of the second experiment are displayed on Figure 5: it compares the collected rewards averaged on 500 independent runs (*i.e.* 20 times each task) with the best parametrisation of *continuous* and *discrete* settings, with the non-optimistic version (*i.e.* with $\kappa = 0$) which corresponds to the method in (Atkeson and Santamaria 1997), and with a learning that would not use any transfer. Without transfer, the time scale (only 1000 transitions) is too short to converge to any efficient policy because the dynamics are too complex. The non-optimistic version is better at the very beginning because it exploits right away what it finds, but also regularly fails to converge because of this premature exploitation trend. But the most interesting result that can be observed is the fact that the *continuous* setting dominates the *discrete* one if the updates are made every time a confidence interval is divided by 2. Actually, the *continuous* setting triggers twice more policy updates (80 vs. 40 on average) than the *discrete* setting. By setting the update parameter to 1.2 instead of 2, the number of updates becomes similar as well as the performance curves. After all, the knowledge of the reward function shape does not help: there is the same number of unknown values (25), and our continuous version of confidence bounds demonstrate good properties. Figures 7a and 7b compare the cumulative rewards after the first 100, first 200, first 500, all 1000, and last 500 episodes for the *continuous* setting (the same results are obtained in the *discrete* setting) according to their parameter values. One can notice that the best ones depend on the horizon of the learning, and that the efficiency is much more sensitive to κ values than to σ values.

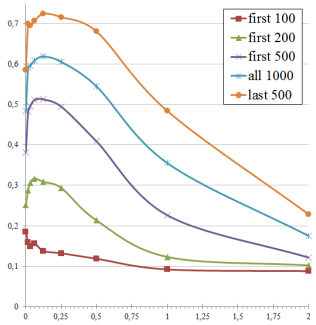
In the third experiment, only corner tasks $\tau_{11}, \tau_{15}, \tau_{51}$, and τ_{55} are considered. Once again, we compare the *continuous* setting and the *discrete* setting where it is known that rewards are exclusively distributed at the four corners. More formally, task τ is known to be defined as follows: $R_\tau : \{1,5\}^2 \to \mathbb{R}$. It corresponds to the case when strong expert knowledge may be injected into the system in order to speed up learning. Figures 4a, 4b, and 4c show that exploration is well targeted at the corners and that unnecessary exploration is avoided. Figure 6 compares both settings and one can notice that initial knowledge boosts significantly early convergence. Once again, one needs to pay attention that less than 10 updates on average are realised after the 1000 fist transitions with the update parameter set to 2, and the performance of online learning gets impaired. The update parameter set to 1.05 reaches 50 updates and a better convergence curve.



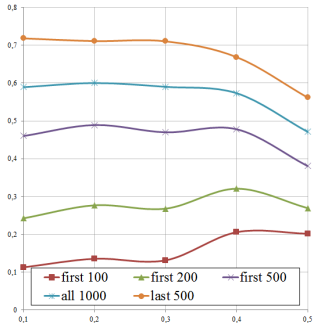
(5) 25 tasks: average immediate rewards received in function of transition index. Comparison of no-transfer learning and non-optimistic and optimistic in *continuous* and *discrete* settings.



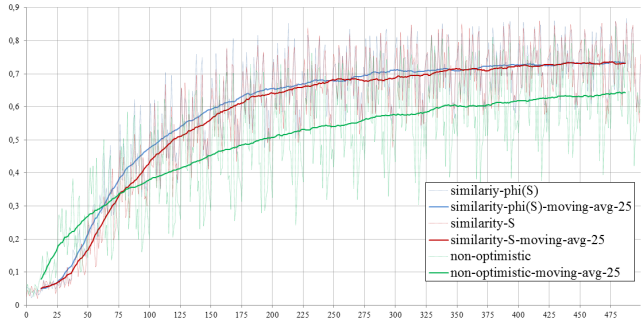
(6) 4 tasks (in corners): average immediate rewards received in function of transition index. Comparison of optimistic in *continuous* setting and in *discrete* setting with injection of strong expert knowledge.



(7a) Sensitivity to κ: average reward received at given transition ranges for several values of κ.



(7b) Sensitivity to σ: average reward received at given transition ranges for several values of σ.



(8) Multi-Task RL: average immediate rewards received in function of trajectory index. Comparison of non-optimistic with the two similarity definitions in Equations 6 and 7 in the *continuous* setting.

Figures 5-8: Performance plots for Transfer RL and Multi-Task RL experiments under *continuous* and *discrete* settings.

Multi-Task Reinforcement Learning experiment

The Multi-Task experiment unfolds as follows. The 25 tasks are run in turns: one trajectory of 50 transitions of each task is generated with the current policy and is stored in Ξ_τ, until collecting 20 trajectories from each task, *i.e.* 1,000 transitions from each task and 25,000 in total. In a multi-task setting, one cannot anymore consider that P is known. Indeed, P will be learnt at the same time as R_τ. Still, in order to show the efficiency of our approach, the same algorithm will be applied whatever the size of Ξ, and estimator $\hat{P}$ is considered as the true P. The updates of the policies will be made independently from one task to another, and they will be entirely driven by the confidence interval reductions, as in the Transfer RL experiments.

Figure 8 displays the 200-run average performance over time under the *continuous* setting. It compares the non-optimistic version and the two similarity measures defined in Equations 6 and 7. It reveals that both similarity definitions are almost equally efficient and that they significantly outperform the non-optimistic setting. The regular shape of the curves is explained by the fact that the task is deterministically chosen and that some tasks are easier than others.

Unfortunately, comparison with state-of-the-algorithm is limited to the non optimistic setting. (Barreto et al. 2016) and (Borsa, Graepel, and Shawe-Taylor 2016) studies apply to the Multi-Task RL experiment, but direct comparison was still difficult for the following reasons:

In addition to the shared dynamics, (Barreto et al. 2016) assumes that the reward function parameters are similar among tasks, which is not an assumption that is made in our experiments. Still, one can notice that their experiment dynamics were simpler: no wall and reduced action noise, and that their reward function is deterministic. Despite dealing with a much easier problem, the reward function changed every 5000 transitions in their most complex setting. Our problem of 25 tasks is near solved after 300 episodes of 50 transitions for a total of 15,000 transitions, *i.e.* 600 transition per task. Additionally, they needed 30,000 transitions over 6 tasks to find a good transfer model. It is two times more than our approach on 25 tasks.

(Borsa, Graepel, and Shawe-Taylor 2016) does not make any assumption of this kind, but their algorithm cannot be directly applied to an online learning problem. As for (Barreto et al. 2016), their experiment is much simpler: deterministic dynamics and reward functions, and despite this, the required sample budget was of 500 transitions per task, when we only need 600 in our very noisy setting.

As a conclusion, our approach to Multi-Task RL with shared dynamics shows a breakthrough in performance. Its implementation with UCRL probably satisfies convergence bounds that are in the same range as UCRL ones, but this part is left for further studies.

References

Antos, A.; Szepesvári, C.; and Munos, R. 2007. Value-iteration based fitted policy iteration: learning with a single trajectory. In *Proceedings of the 1st IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL)*, 330–337. IEEE.

Atkeson, C. G., and Santamaria, J. C. 1997. A comparison of direct and model-based reinforcement learning. In *Proceedings of the 14th International Conference on Robotics and Automation*, 3557–3564. IEEE Press.

Auer, P., and Ortner, R. 2007. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NIPS)*, volume 19, 49.

Barreto, A.; Munos, R.; Schaul, T.; and Silver, D. 2016. Successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1606.05312*.

Borsa, D.; Graepel, T.; and Shawe-Taylor, J. 2016. Learning shared representations in multi-task reinforcement learning. *arXiv preprint arXiv:1603.02041*.

Brafman, R. I., and Tennenholtz, M. 2002. R-max: a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research* 3(Oct):213–231.

Chandramohan, S.; Geist, M.; and Pietquin, O. 2010. Optimizing spoken dialogue management with fitted value iteration. In *Proceedings of the 10th Annual Conference of the International Speech Communication Association (Interspeech)*, 86–89.

Ernst, D.; Geurts, P.; and Wehenkel, L. 2005. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research* 503–556.

Genevay, A., and Laroche, R. 2016. Transfer learning for user adaptation in spoken dialogue systems. In *Proceedings of the 15th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. International Foundation for Autonomous Agents and Multiagent Systems.

Holm, S. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics* 65–70.

Houthooft, R.; Chen, X.; Duan, Y.; Schulman, J.; De Turck, F.; and Abbeel, P. 2016. Curiosity-driven exploration in deep reinforcement learning via bayesian neural networks. *arXiv preprint arXiv:1605.09674*.

Kearns, M., and Singh, S. 2002. Near-optimal reinforcement learning in polynomial time. *Machine Learning* 49(2-3):209–232.

Lakshmanan, K.; Ortner, R.; and Ryabko, D. 2015. Improved regret bounds for undiscounted continuous reinforcement learning. In *Proceedings of The 32nd International Conference on Machine Learning (ICML)*, volume 37, 524–532.

Laroche, R.; Putois, G.; Bretier, P.; and Bouchon-Meunier, B. 2009. Hybridisation of expertise and reinforcement learning in dialogue systems. In *Proceedings of the 9th Annual Conference of the International Speech Communication Association (Interspeech)*, 2479–2482.

Lazaric, A. 2012. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*. Springer. 143–173.

Lemon, O., and Pietquin, O. 2012. *Data-Driven Methods for Adaptive Spoken Dialogue Systems: Computational Learning for Conversational Interfaces* . Springer.

Lopes, M.; Lang, T.; Toussaint, M.; and Oudeyer, P.-Y. 2012. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS)*, 206–214.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Moore, A. W., and Atkeson, C. G. 1993. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning* 13(1):103–130.

Ng, A., and Russell, S. 2000. Algorithms for inverse reinforcement learning. In *in Proceedings of the 17th International Conference on Machine Learning (ICML)*, 663–670. Morgan Kaufmann.

Ortner, R., and Ryabko, D. 2012. Online regret bounds for undiscounted continuous reinforcement learning. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS)*, 1763–1771.

Osband, I.; Roy, B. V.; and Wen, Z. 2016. Generalization and exploration via randomized value functions. In *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016*, 2377–2386.

Romoff, J.; Bengio, E.; and Pineau, J. 2016. Deep conditional multi-task learning in atari.

Scheffe, H. 1999. *The analysis of variance*, volume 72. John Wiley & Sons.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press.

Szepesvári, C. 2010. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning* 4(1):1–103.

Taylor, M. E., and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10:1633–1685.

Taylor, M. E.; Stone, P.; and Liu, Y. 2007. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research* 8(Sep):2125–2167.

Tesauro, G. 1995. Td-gammon: A self-teaching backgammon program. In *Applications of Neural Networks*. Springer. 267–285.

Tikhonov, A. N. 1963. Regularization of incorrectly posed problems. Soviet Mathematics Doklady.