Transfer of Samples in Batch Reinforcement Learning

Alessandro Lazaric LAZARIC@ELET.POLIMI.IT
Marcello Restelli RESTELLI@ELET.POLIMI.IT
Andrea Bonarini BONARINI@ELET.POLIMI.IT
DEI, IIT-Lab, Politecnico di Milano, P.za Leonardo da Vinci, 32, I-20133, Milan, Italy

Abstract

The main objective of transfer in reinforcement learning is to reduce the complexity of learning the solution of a target task by effectively reusing the knowledge retained from solving a set of source tasks. In this paper, we introduce a novel algorithm that transfers samples (i.e., tuples $\langle s, a, s', r \rangle$) from source to target tasks. Under the assumption that tasks have similar transition models and reward functions, we propose a method to select samples from the source tasks that are mostly similar to the target task, and, then, to use them as input for batch reinforcement-learning algorithms. As a result, the number of samples an agent needs to collect from the target task to learn its solution is reduced. We empirically show that, following the proposed approach, the transfer of samples is effective in reducing the learning complexity, even when some source tasks are significantly different from the target task.

1. Introduction

The main objective of transfer in Reinforcement Learning (RL) is to reduce the learning time. In fact, the solution of a set of *source* tasks can provide useful information about how to solve a related *target* task, thus reducing the amount of experience needed to solve it. In order to design an effective transfer algorithm, two aspects must be taken into account: *what* to transfer, that is the knowledge retained from the source tasks, and *when* to transfer, that is the identification of tasks from which transfer is likely to be effective.

There exists a much empirical evidence about the effectiveness of techniques such as *task decomposition*, *options*, *shaping rewards*, *exploration strategies*, in improving the learning speed of RL algorithms in single-task problems. Many studies focus on extending such techniques to the transfer scenario. In particular, hierarchical solutions are often used (Şimşek et al., 2005; Mehta et al., 2005) to augment the action space with policies suitable for the solution of a wide range of tasks sharing the same dynamics, but with different goals. In (Konidaris & Barto, 2007), a set of options is learned in an *agent space* defined by a set of features shared across the tasks, thus making the options reusable even in tasks with different state spaces. The improvement of learning speed can also be obtained through direct transfer of solutions from source to target task. In this scenario, the main issue is to map the solution learned in a source task to the state-action space of the target task, thus initializing the learning algorithm to a convenient solution. Different aspects of a learning algorithm can be initialized, such as value functions, policies, and approximator structure (Taylor et al., 2007, and references therein).

Although these approaches study how the transfer of different elements from source to target tasks can impact on the performance of an RL algorithm, they often rely on the assumption that the tasks are strictly related and they do not address the problem of *negative transfer* (Rosenstein et al., 2005). In fact, transfer may bias the learning process towards solutions that are completely different from the optimal one, thus worsening the learning performance. Some works focus on the definition of measures of relatedness between tasks that can be used to select from which source tasks transfer is actually convenient. An experimental analysis of measures that estimate the expected speed-up on the basis of information such as policy overlapping, Q-values, and reward structure is reported in (Carroll & Seppi, 2005). Unfortunately, it is often difficult to compute these measures before actually solving the target task, and, thus, they can be used only to analyze the effectiveness of a transfer

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

method. In (Ferns et al., 2004), different metrics for the distance between tasks are proposed and theoretical bounds on the difference between the corresponding optimal value functions are derived.

In this paper, we focus on a perspective that received little attention so far, the transfer of samples. We propose a mechanism that selectively transfers samples from source to target tasks on the basis of the similarity of source tasks with the samples collected in the target task. We introduce a criterion to select from which sources transfer should occur, and, within each task, which samples are more likely to speed-up the learning process. As a result, through selective transfer of samples, it is possible to reduce the number of samples needed to solve the target task.

The paper is organized as follows. In Section 2 we introduce notation and we briefly review batch RL. In Section 3 we propose a novel mechanism for transfer of samples in batch RL algorithms. In Section 4 we report the experimental results of sample transfer. In Section 5 we relate our work with other transfer-learning approaches. Finally, in Section 6 we draw conclusions, and we propose directions for future works.

2. Batch Reinforcement Learning

In RL, the interaction between the agent and the environment is modeled as a discrete-time Markov Decision Process (MDP). An MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Pi(\mathcal{S})$ is the transition model that assigns to each state-action pair a probability distribution over $\mathcal{S}$, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \Pi(\mathbb{R})$ is the reward function that assigns to each state-action pair a probability distribution over $\mathbb{R}$, $\gamma \in [0, 1)$ is the discount factor. At each time step, the agent chooses an action according to its current *policy* $\pi : \mathcal{S} \to \Pi(\mathcal{A})$, which maps each state to a probability distribution over actions. The goal of an RL agent is to maximize the expected sum of discounted rewards, that is to learn an optimal policy π^* that leads to the maximization of the value function in each state. The optimal action-value function $Q^*(s, a)$ is defined by the Bellman equations

$$Q^*(s,a) = \sum_{s'} \mathcal{P}(s'|s,a) \left[R(s,a) + \gamma \max_{a'} Q^*(s',a') \right],$$

where $R(s, a) = E[\mathcal{R}(s, a)]$ is the expected reward.

One of the main drawbacks of online RL algorithms (e.g., Q-learning) when applied to real-world problems is the large amount of experience needed to solve a task. In order to overcome this drawback, batch approaches have been proposed. The main idea is to distinguish between the exploration strategy that collects samples of the form $\langle s, a, s', r \rangle$ (*sampling* phase), and the offline learning algorithm that, on the basis of the samples, computes the approximation of the action-value function (*learning* phase). In this paper, we focus on fitted algorithms, although the proposed transfer mechanism can be applied to any batch RL algorithm. The idea underlying fitted solutions (Ernst et al., 2005, and references therein) is to reformulate the learning of the value function as a sequence of regression problems. Given a set of samples, Fitted Q-Iteration (FQI)(Ernst et al., 2005) estimates the optimal action-value function by iteratively extending the optimization horizon. At the first iteration, the algorithm defines a regression problem for a 1-step problem, in which the action-value function is equal to the reward function. An approximation is computed running a chosen regression algorithm on the available samples. Thereafter, at each iteration k, corresponding to a k-step horizon, a new regression problem is stated, in which the training samples are computed exploiting the approximation of the action-value function at the previous iteration.

3. Transfer of Samples in Batch Reinforcement Learning

We formulate the transfer problem as the problem of solving a target task given a set of source tasks drawn according to a given probability distribution defined on a set of tasks which differ in either the transition model or the reward function, or both, but share the same state-action space.

Definition 1 *A* task T *is an MDP defined by the tuple* $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}_T, \mathcal{R}_T, \gamma \rangle$*, in which the transition model* $\mathcal{P}_T$ *defines the* dynamics*, and the reward function* $\mathcal{R}_T$ *defines the* goal*.*

Definition 2 *An* environment E *is defined by the tuple* $\langle \mathcal{T}, \Omega \rangle$*, where* $\mathcal{T}$ *is the* task space *and* Ω *is the* task distribution *that provides the probability of a task* $T \in \mathcal{T}$ *to occur.*

In batch RL algorithms, the element that mainly affects the learning performance is the set of samples used to feed the learning algorithm, the more informative the samples the better the approximation. We focus on the way this set of samples can be augmented by the inclusion of samples drawn from a set of source tasks. The basic intuition underlying this idea is that, since tasks are related through the task distribution Ω, some of the source tasks are likely to contain samples similar to those in the target task. Therefore, we expect the transfer of samples to improve performance of

batch RL algorithms even when a very limited number of samples have been actually collected from the target task. This improvement is particularly important in domains where sampling is slow and expensive (e.g., robotic applications).

More formally, we consider the scenario in which a set of n source tasks $\{S_k\}$, with $S_k \in \mathcal{T}$ and $k \in \mathbb{N}_n$, drawn from Ω are available. From each source task m samples have been collected, while only $t \ll m$ samples are available from the target task T. Let $\{\widehat{S}_k\}$ and $\widehat{T}$ be the sample sets for the source and target tasks respectively. The transfer algorithm selects a set of samples from the source tasks that are used to augment $\widehat{T}$, thus building a new set of samples $\widetilde{T}$. Finally, samples in $\widetilde{T}$ are used as input for the learning algorithm.

3.1. Task Compliance

The main problem of transferring samples across tasks is to avoid negative transfer, that is the transfer of samples from source tasks that are significantly different from the target task. Therefore, we need to identify which source tasks are more likely to have samples similar to those in the target task. Alternatively, this problem can be stated as a *model identification problem*. Let us consider the following scenario: The task space $\mathcal{T}$ contains n tasks, and m samples have been already collected from each task. Let T be a new task drawn according to Ω and $\widehat{T}$ the set of samples collected from it, with $|\widehat{T}| = t \ll m$. Since the transfer of samples from all the tasks in $\mathcal{T}$ may worsen the performance in T, we need to identify which of the previously solved tasks is actually T according to the available samples. Starting from a uniform prior over the tasks in $\mathcal{T}$, we compute the posterior distribution as the probability of a task to be the model from which samples in $\widehat{T}$ are drawn. As the number of samples t increases, the posterior distribution is updated accordingly until the total probability mass concentrates on the task equal to T. Then, the m samples previously collected in the task equal to T can be added to $\widehat{T}$ and used to feed the batch RL algorithm, thus improving its learning performance.

In the general case in which $\mathcal{T}$ is infinite or contains many tasks, the probability to have one source task identical to the target task is negligible. Thus, instead of the probability of a source task to generate *all* the samples collected in the target task, we compute its *compliance* with T as the *average* probability of generating the samples in $\widehat{T}$. Then, we transfer samples from source tasks proportionally to their compliance with the target task.

Let us consider a source task S and the set of target samples $\widehat{T}$. Given a state-action pair $\langle s, a\rangle$, the probability of S to be the model from which the target samples in $\langle s, a\rangle$ are extracted, that is the likelihood of the model in $\langle s, a\rangle$, can be simply computed by applying the Bayes theorem as [1]

$$\begin{aligned} P\left(S|\widehat{T}_{\langle s,a\rangle}\right) &\propto P\left(\widehat{T}_{\langle s,a\rangle}|S\right) P\left(S\right) \\ &= \prod_{\tau_i \in \widehat{T}_{\langle s,a\rangle}} P\left(\tau_i|S\right) P\left(S\right) \\ &= \prod_{\tau_i \in \widehat{T}_{\langle s,a\rangle}} \mathcal{P}_S(s'_i|s_i,a_i)\mathcal{R}_S(r_i|s_i,a_i) P\left(S\right), \end{aligned} \quad (1)$$

where $\widehat{T}_{\langle s,a\rangle} = \{\tau_i \in \widehat{T} | s_i = s, a_i = a\}$, $P\left(S\right)$ is the *prior* on the source task S, and $P(S|\widehat{T}_{\langle s,a\rangle})$ is the *posterior* distribution over the source tasks in $\langle s, a\rangle$.

Unfortunately, the posterior probability cannot be immediately computed without the exact model of S. On the other hand, we have a set of m samples $\widehat{S}$ previously collected in S, from which an approximation of the continuous model can be computed. In the following, with an abuse of notation, with $\widehat{T}$ and $\widehat{S}$ we denote both the sets of samples and the model approximations built on them. Let $\tau_i = \langle s_i, a_i, s'_i, r_i\rangle$ be a sample in $\widehat{T}$, the probability of this sample to be generated by S given the set of source samples $\widehat{S}$ is

$$P(\tau_i|\widehat{S}) = \mathcal{P}_{\widehat{S}}(s'_i|s_i,a_i)\mathcal{R}_{\widehat{S}}(r_i|s_i,a_i),$$

where $\mathcal{P}_{\widehat{S}}$ and $\mathcal{R}_{\widehat{S}}$ are the approximated transition and reward models respectively. Since in continuous spaces the probability to have samples in the same state-action pair is negligible, it is necessary to use an approximation that generalizes over all the samples close to $\langle s_i, a_i\rangle$. In particular, we follow the kernel-based approximation proposed in (Jong & Stone, 2007).

Let $\varphi(\cdot)$ be a kernel function (e.g., a Gaussian kernel $\varphi(x) = \exp(-x^2/\delta)$ with bandwidth δ) applied to a given distance metric d (e.g., Euclidean or Mahalanobis distance). First of all, we define the similarity (*compliance* in the following) between the experience tuple τ_i and the experience tuples $\sigma_j \in \widehat{S}$ in terms of dynamics and reward. We define the compliance of τ_i with respect to σ_j for the transition model as

$$\lambda^{\mathcal{P}}_{ij} = w_{ij} \cdot \varphi\left(\frac{d(s'_i, s_i + (s'_j - s_j))}{\delta_{s'}}\right),$$

where

$$w_{ij} = \frac{\varphi\left(\frac{d(\langle s_i,a_i\rangle,\langle s_j,a_j\rangle)}{\delta_{sa}}\right)}{\sum_{l=1}^{m} \varphi\left(\frac{d(\langle s_i,a_i\rangle,\langle s_l,a_l\rangle)}{\delta_{sa}}\right)}.$$

[1] We assume that samples are mutually independent.

While the first term (w_{ij}) of $\lambda^{\mathcal{P}}_{ij}$ is a weight that takes into consideration the relative closeness of the two samples in the state-action space, the second term measures the similarity of the outcome. In particular, under the assumption that the transition model is continuous in the state-action space, it measures the distance between s'_i and the state obtained by applying the state transition $(s'_j - s_j)$ of σ_j to state s_i (see Jong & Stone, 2007). Therefore, the dynamics of τ_i is highly compliant with that of σ_j when they are close and their state transitions are similar.
Similarly, the compliance of the reward in τ_i with respect to that of σ_j is defined as

$$\lambda^{\mathcal{R}}_{ij} = w_{ij}\varphi\left(\frac{|r_i - r_j|}{\delta_r}\right).$$

The approximated transition and reward models are the average of the compliance between τ_i and all the samples in $\widehat{S}$

$$\mathcal{P}_{\widehat{S}}(s'_i|s_i,a_i) = \frac{1}{Z^{\mathcal{P}}}\sum_{j=1}^{m}\lambda^{\mathcal{P}}_{ij};\;\; \mathcal{R}_{\widehat{S}}(r_i|s_i,a_i) = \frac{1}{Z^{\mathcal{R}}}\sum_{j=1}^{m}\lambda^{\mathcal{R}}_{ij},$$

where $Z^{\mathcal{P}}$ and $Z^{\mathcal{R}}$ are normalization terms. Finally, we define the *compliance* of τ_i to S approximated using samples in $\widehat{S}$ as

$$\lambda_i = P(\tau_i|\widehat{S}) = \frac{1}{Z^{\mathcal{P}}Z^{\mathcal{R}}}\left(\sum_{j=1}^{m}\lambda^{\mathcal{P}}_{ij}\right)\left(\sum_{j=1}^{m}\lambda^{\mathcal{R}}_{ij}\right).$$

Recalling Equation 1, given the compliance of samples in $\langle s,a\rangle$, the probability of the model in $\langle s,a\rangle$ becomes

$$P\left(S|\widehat{T}_{\langle s,a\rangle}\right) \propto \prod_{\tau_i\in\widehat{T}_{\langle s,a\rangle}}\lambda_i P(S). \tag{2}$$

Starting from the probability in each state-action pair, we compute a global measure of the probability for the task to contain samples similar to target samples. We define the compliance of a task S as the average likelihood computed over each state-action pair experienced in the target task.

Definition 3 *Given the target samples $\widehat{T}$ and the source samples $\widehat{S}$, the* task compliance *of S is*

$$\Lambda = \frac{1}{|\widehat{U}|}\sum_{\langle s,a\rangle\in\widehat{U}} P\left(S|\widehat{T}_{\langle s,a\rangle}\right), \tag{3}$$

where $\widehat{U}$ contains all the distinct state-action pairs in the samples of $\widehat{T}$.

Since the probability to have two samples in the very same state-action pair is negligible, it follows that $|\widehat{U}| = |\widehat{T}| = t$ and the previous definition reduces to

$$\Lambda = \frac{1}{t}\sum_{i=1}^{t}\lambda_i P(S), \tag{4}$$

where $P(S)$ is a prior on the source task. When n source tasks with m samples each are available, and t samples are collected from T, the computation of the task compliance has a time complexity of $\Theta(nmt)$.

3.2. Sample Relevance

Although the measure of compliance is effective in identifying which sources, in average, are more convenient to transfer samples from, it does not provide any suggestion about which samples in $\widehat{S}$ are actually better to transfer. In the following, we introduce the concept of *relevance* of each sample $\sigma_j \in \widehat{S}$. The idea is to use the compliance of σ_j with the target task. Unfortunately, in this case, the measure of compliance is often unreliable because of a poor approximation of the target task. In fact, while each source task contains m samples, only $t \ll m$ samples are available for the target task. As a result, it may happen that the compliance of σ_j is computed according to samples τ_i that are significantly far in the state-action space. Therefore, we need a formulation of relevance strictly related to the compliance whenever the number of samples in $\widehat{T}$ close to σ_j is sufficient, while tending to a default value when the compliance is not reliable. Given the definition of compliance $\lambda^{\mathcal{P}}_{ji}$ and $\lambda^{\mathcal{R}}_{ji}$ of σ_j with a sample τ_i, the compliance of σ_j with the approximated model of the target task $\widehat{T}$ is

$$\lambda_j = P(\sigma_j|\widehat{T}) = \frac{1}{Z^{\mathcal{P}}Z^{\mathcal{R}}}\left(\sum_{i=1}^{t}\lambda^{\mathcal{P}}_{ji}\right)\left(\sum_{i=1}^{t}\lambda^{\mathcal{R}}_{ji}\right). \tag{5}$$

Let the samples τ_i be sorted in ascending order according to w_{ji}. We compute the average distance between σ_j and the samples $\tau_i \in \widehat{T}$ as

$$d_j = \frac{1}{h_j}\sum_{i=1}^{h_j} d(\langle s_j,a_j\rangle, \langle s_i,a_i\rangle), \tag{6}$$

where h_j is such that $\sum_{i=1}^{h_j} w_{ji} < \mu$, where $\mu \in (0;1]$ determines the fraction of the total number of samples considered in the computation of the average distance.

Definition 4 *Given the compliance λ_j and the average distance d_j, the* relevance *of σ_j is defined as*

$$\rho_j = \rho(\overline{\lambda}_j, d_j) = e^{-\left(\frac{\overline{\lambda}_j - 1}{d_j}\right)^2}, \tag{7}$$

where $\overline{\lambda}_j$ is the compliance normalized over all the samples in $\widehat{S}$.

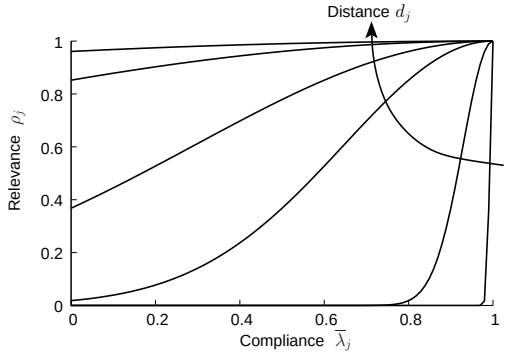

Figure 1. Relevance function for different values of d_j.

Algorithm 1 The sample transfer algorithm

Input: source tasks $\{S_k\}_{k\in\mathbb{N}_n}$, target task T
Parameters: δ_{sa}, $\delta_{s'}$, δ_r, t, m
Output: transferred sample set $\widetilde{T}$
for $k = 1$ to n **do**
  $\widehat{S}_k \leftarrow$ sampling(S_k, m)
end for
$\widehat{T} \leftarrow$ sampling(T, t)
for $k = 1$ to n **do**
  $\Lambda_k \leftarrow$ compliance$(\widehat{S}_k, \widehat{T})$
  for $\sigma_j \in \widehat{S}_k$ **do**
    $\rho_j \leftarrow$ relevance$(\sigma_j, \widehat{T})$
  end for
  Draw $(m-t)\overline{\Lambda}_k$ samples from $\widehat{S}_k$ proportionally to ρ_j
end for
Put the additional samples in $\widehat{T}$ and form the sample set $\widetilde{T}$

The relevance function is shown in Figure 1 for different values of distance d_j. As it can be noticed, sample σ_j may have high relevance in two distinct cases: *(i)* where there is a number of close samples τ_i which it is compliant with, *(ii)* where there are no close samples and, independently from the compliance, we assume a high relevance value. The assumption underlying this definition is that, whenever there is no evidence against the transfer of a sample, it is convenient to transfer it. In fact, in transfer problems the learner often needs to infer knowledge about unexplored regions of the target task. In these regions, the algorithm selects samples from the most compliant source tasks. The assumption is that samples far from target samples, but drawn from highly compliant tasks, are worth transferring, since they provide information in regions that have not been actually experienced.

3.3. Transfer of Samples

The actual transfer process is based on the compliance of the source tasks with the target samples and on the relevance of samples within each source task. For sake of simplicity, we bound the number of samples used by the learning algorithm to m. Since $|\widehat{T}| = t$ samples are already available, $m - t$ samples need to be extracted and transferred from the source tasks. For each source task S_k, the number of samples transferred to the sample set $\widetilde{T}$ of the new target task is proportional to its normalized compliance $\overline{\Lambda}_k = \frac{\Lambda_k}{\sum_{l=1}^{n} \Lambda_l}$. Then, for each source task, samples are drawn according to their relevance, thus avoiding to transfer samples that are quite dissimilar from those in the target task. The whole sample-transfer process is summarized in Algorithm 1.

4. Experiments

In order to evaluate the performance of the sample-transfer algorithm we consider a variant of the boat problem proposed in (Lazaric et al., 2007). The problem is to learn a controller to drive a boat from the left bank to the right-bank quay of a river, in presence of a non-linear current. The boat's bow coordinates, x and y, are defined in the range $[0, 200]$ and the controller sets the desired direction $a \in [-90°, -45°, 0°, 45°, 90°]$. The chosen action is perturbed by a uniform noise in the range $[-5°; 5°]$. The control frequency is set to 1Hz. For the lack of space, we refer the reader to (Lazaric et al., 2007) for the equations of the dynamics. In addition, we introduce sandbanks, i.e., regions of the river in which the speed is reduced by 20%. The reward function is defined as:

$$\mathcal{R}(x,y) = \begin{cases} +10 & x = 200 \text{ and } y \in \mathcal{Z}_s \\ \text{D(x,y)} & x = 200 \text{ and } y \in \mathcal{Z}_v \\ \text{-10} & x = 200 \text{ and } y \in \mathcal{Z}_f \\ \text{-2} & (x,y) \in \text{sandbank} \\ \text{-2} & y \leq 0 \text{ or } y \geq 200 \\ 0 & \text{elsewhere} \end{cases} \quad (8)$$

where D is a function that gives a reward decreasing linearly from 10 to -10 relative to the distance from the quay, $\mathcal{Z}_s$ is the quay zone, $\mathcal{Z}_v$ is the viability zone around the quay, and $\mathcal{Z}_f$ is the failure zone in all the other bank points. The dynamics and learning parameters are summarized in Tab. 1. In the following experiments, we use Gaussian kernels and Mahalanobis distance (see Section 3.1). The results are obtained by averaging 100 runs. In FQI, we use extra-randomized trees (Ernst et al., 2005) with 50 trees, 2 random splits, and 2 minimum sample size for each node, trained on 25 iterations. Samples are obtained through random sampling run on independent episodes of maximum 50 steps each. Each episode restarts the boat at the left bank in a random position. Testing is performed on 1,000 episodes with the initial position drawn at random from 20 evenly spaced positions at the left bank.

The first experiment is meant to illustrate the effectiveness of the relevance in identifying which samples are worth transferring. We consider a transfer prob-

Parameter	Value
I/p	0.1 / 0.9
s_{MAX}/s_D	2.5 / 1.75
Z_s / Z_v width	10.0 / 10.0

Parameter	Value
m	2000
μ	0.8
δ_{sa}	0.1
δ_r	0.5
$\delta_{s'}$	0.1

Table 1. The dynamics and transfer parameters.

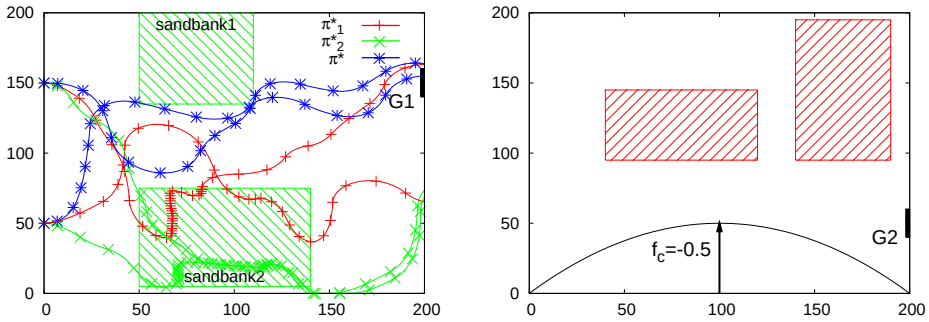

Figure 2. (*left*) Sandbanks and goal of the target task and trajectories of the optimal policies of S_1, S_2, and T tested in T. (*right*) Sandbanks and goal of S_2.

lem with three tasks in which S_1 and S_2 are the source tasks and T is the target task. In T the quay is G_1 and there are two sandbanks as illustrated in Figure 2-(*left*). In task S_1 there are two quays G_1 and G_2, and there is only one sandbank corresponding to the region labeled as *sandbank1* in Figure 2-(*left*). Task S_2 has the quay G_2 and the sandbanks illustrated in Figure 2-(*left*). While T and S_1 have the same current force ($f_c = 0.5$), the current in S_2 is in the opposite direction ($f_c = -0.5$). The source task S_2 has a completely different dynamics and reward function from those in T because of different sandbanks and current. Therefore, samples transferred from S_2 are likely to induce negative effects on the learning performance of T. Furthermore, as it can be noticed from the trajectories shown in Figure 2-(*left*), the optimal policy π^*_2 of S_2 obtains very poor performance when tested on T. On the other hand, S_1 has the same dynamics as T in large regions of the state-action space and shares one goal with T. Although its optimal policy π^*_1 is significantly different from π^*, it is possible to choose samples from $\widehat{S}_1$ to improve the performance in T.

Figure 3-(*left*) shows the performance obtained by FQI with four different configurations: *No Transfer*, *Random*, *Compliance*, and *Relevance Transfer*. In the first configuration FQI is run with samples directly collected from T. The other three configurations are run on the sample set $\widetilde{T}$ obtained by transferring samples chosen at random, according to the compliance, and according to the relevance respectively. Furthermore, we also report the performance obtained by transferring policies π^*_1 and π^*_2 as baselines. The augmentation of $\widehat{T}$ with samples drawn from S_1 and S_2 at random does not lead to any significant improvement of the performance with respect to learning directly on samples in $\widehat{T}$. In fact, the only advantage achieved with the transferred samples is that the agent avoids to go outside of the boundaries, but she learns neither to avoid sandbanks nor to achieve the goal. The main reason for this poor performance is that samples drawn from S_2 do not provide any information about the actual dynamics and rewards of T and, thus, may lead to learning very bad policies. On the other hand, the compliance-based transfer successfully excludes samples of S_2 from the transfer process (the normalized compliance of S_1 for $t = 200$ is $\overline{\Lambda}_1 = 0.93 \pm 0.09$), thus augmenting $\widehat{T}$ with samples mainly coming from S_1. Since S_1 shares with T the dynamics and the rewards in all the state space but at *sandbank2* and in the quay G_2, the transfer is positive and leads to a significant improvement in the performance of the learning process. Nonetheless, there are still many trajectories leading to the quay G_2 and crossing the sandbank because of the negative effect of transferring samples from regions with dynamics and reward different from T. In Figure 3-(*right*) we report the relevance of the samples in $\widehat{S}_1$ (averaged on all the actions). As it can be noticed, the relevance identifies the regions where samples are actually similar in source and target tasks, excluding samples coming from *sandbank2* and the lower quay G_2. As a result, the performance of the relevance-based transfer is further improved.

In order to evaluate the relative improvement of transfer, we compute the area ratio (Taylor et al., 2007) of the three transfer configurations, defined as the difference between the accumulated reward with and without transfer divided by the reward accumulated without transfer. Figure 3-(*center*) shows the area ratio for the three transfer configurations. As it can be noticed, random transfer does not lead to any significant improvement, while relevance-based transfer improves the performance by $75.3\% \pm 13.2$. All the differences are statistically significant ($p < 0.01$).

In the previous experiment, source and target tasks have been designed to show how the algorithm works. Now, we consider the general case in which tasks are

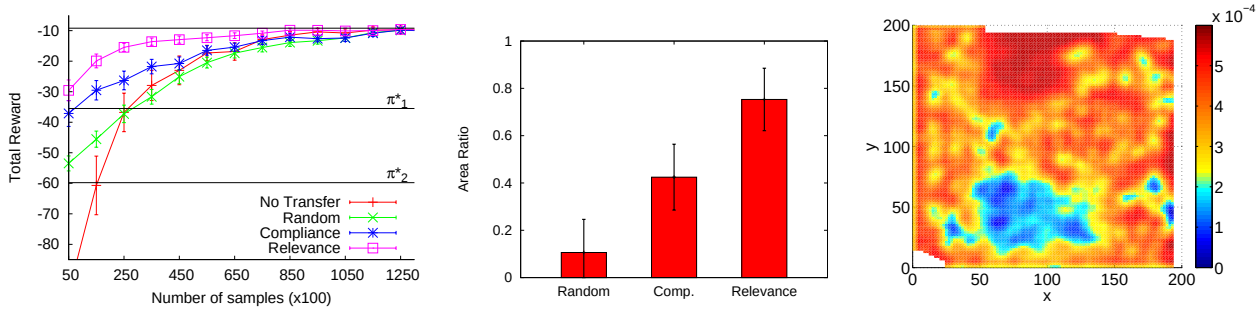

Figure 3. (*left-center*) Total reward and area ratio. (*right*) Relevance of samples in $\widehat{S}_1$ at convergence.

drawn from an infinite task space $\mathcal{T}$. For sake of simplicity, we consider the same target task of the previous experiment, while source tasks have current f_c=0.5 and one sandbank. Source tasks are drawn from a distribution Ω such that the coordinates of the center, height, and width of the sandbank are uniformly drawn from the space $[20.0; 180.0] \times [20.0; 180.0] \times [40.0; 100.0] \times [40.0; 100.0]$, while the quay position is drawn uniformly in $[20.0; 180.0]$. In Figure 4, we report the results of relevance-based transfer obtained by averaging the result with 10 different sets of five source tasks. Although the source tasks are different from the target in large regions, the transfer algorithm is able to identify which samples are worth transferring from the source tasks and it successfully improves the learning performance with an area ratio of $59.5\% \pm 15.4$.

Since the algorithms of transfer in RL proposed so far rely on temporal-difference or model-based learning algorithms, an empirical comparison with the performance of sample transfer would not be fair. Nonetheless, in the next section, we discuss its similarities and differences with other transfer approaches.

5. Related Works

In (Sunmola & Wyatt, 2006) a Bayesian approach is used for transfer of MDPs, where source task models are pre-posteriors for the distributions of the parameters of the target model and model-based RL is used to compute the solution. Although we similarly adopt a Bayesian argument in the compliance, we directly transfer samples and we use a model-free learning algorithm. Furthermore, instead of a parametric approximation of the model of the source tasks, we follow a non-parametric solution.

The task compliance can be interpreted as a sort of distance metric between tasks. In (Ferns et al., 2004), distance metrics for MDP similarity are introduced in the context of bisimulation to aggregate states with similar dynamics and reward. Under a transfer perspective, these metrics can be used to measure the difference between states in distinct tasks and to bound the performance loss of using the optimal policy of a source task in the target task. Unfortunately, this technique cannot be directly applied to our scenario for different reasons. The computation of the Kantorovich distance between different states is very expensive, because it requires the solution of a complex optimization problem. Furthermore, the proposed algorithm needs either the exact models of tasks or accurate approximations. On the other hand, we adopt a solution with low computational complexity, linearly depending on the number of samples of the source tasks. Finally, empirical analysis (Phillips, 2006) showed that the theoretical bounds on the performance loss are too loose and they do not provide useful directions about the actual performance of the transferred policy.

The transfer of samples is also related to works about transfer of solutions in the RL context (Taylor et al., 2007). Although the transfer of samples or solutions (e.g., policies) from only one source task obtains similar results, there are situations in which sample transfer can obtain better results than solution transfer. Even when the difference between source and target tasks is limited to few state-action pairs, the optimal policies of the two tasks can be significantly different and the transfer may achieve very poor performance. On the other hand, the transfer of samples can still be effective. In fact, since most of the samples in the two tasks are identical, the learning algorithm can benefit from samples coming from the source task independently from the actual difference of their optimal policies. Furthermore, the transfer of samples does not require to actually solve the source tasks, and it can be used even when the samples are not enough to solve source tasks. In (Mehta et al., 2005) a solution in which the model-based hierarchical task decomposition allows for transfer at multiple levels of the hierarchy is proposed. This approach relies on the assumption that rewards are a linear combination of *basis* re-

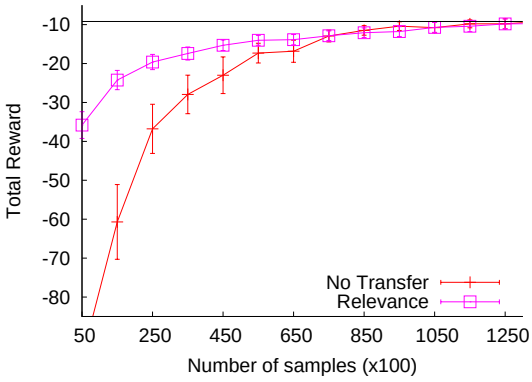

Figure 4. Performance with five random source tasks.

ward functions and it can be applied only to problems of goal transfer, with a fixed transition model. On the other hand, sample transfer can be applied to any transfer scenario. Finally, a method for mapping samples from a source to a target task with different state-action space is proposed in (Taylor et al., 2008).

6. Conclusions

In this paper, we introduced a mechanism for the transfer of samples with the aim of improving the learning performance. The main advantages of the proposed solution are: *(i)* it is independent from the similarity of the policies and action-value functions of the tasks at hand and, thus, can be applied to a wide range of problems, *(ii)* it is independent from the batch RL algorithm, *(iii)* it can be applied to any transfer problem in which either reward or transition or both models change. Experimental results show the effectiveness of the method in improving the learning performance and in avoiding negative transfer when the source tasks are significantly different from the target.

Some aspects of the algorithm can be improved in future works. In case of tasks that either share exactly the same transition or reward model, it is possible to transfer only the part of the samples common to all the tasks. For instance, if two tasks share the same transition model, but have different goals, it is possible to transfer the $\langle s, a, s' \rangle$ part of the samples and to "complete" the sample using an approximation of the reward function of the target task (e.g., using the first iteration of FQI). Furthermore, the sample-transfer algorithm could be integrated with the model proposed in (Taylor et al., 2008) in order to deal with problems with tasks defined on different state-action spaces.

References

Carroll, J. L., & Seppi, K. (2005). Task similarity measures for transfer in reinforcement learning task libraries. *Proceddings of IJCNN* (pp. 803–808).

Şimşek, O., Wolfe, A. P., & Barto, A. G. (2005). Identifying useful subgoals in reinforcement learning by local graph partitioning. *Proceedings of ICML* (pp. 816–823).

Ernst, D., Geurts, P., & Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, *6*, 503–556.

Ferns, N., Panangaden, P., & Precup, D. (2004). Metrics for finite markov decision processes. *Proceedings of UAI* (pp. 162–169).

Jong, N. K., & Stone, P. (2007). Model-based function approximation for reinforcement learning. *Proceedings of AAMAS* (pp. 1–8).

Konidaris, G., & Barto, A. G. (2007). Building portable options: Skill transfer in reinforcement learning. *Proceedings of IJCAI* (pp. 895–900).

Lazaric, A., Restelli, M., & Bonarini, A. (2007). Reinforcement learning in continuous action spaces through sequential monte carlo methods. *Advances in Neural Information Processing Systems*.

Mehta, N., Natarajan, S., Tadepalli, P., & Fern, A. (2005). Transfer in variable-reward hierarchical reinforcement learning. *NIPS Workshop on Inductive Transfer*.

Phillips, C. (2006). *Knowledge transfer in markov decision processes* (Technical Report). McGill School of Computer Science. (http://www.cs.mcgill.ca/~martin/usrs/phillips.pdf).

Rosenstein, M. T., Marx, Z., Kaelbling, L. P., & Dietterich, T. G. (2005). To transfer or not to transfer. *NIPS Workshop on Inductive Transfer*.

Sunmola, F. T., & Wyatt, J. L. (2006). Model transfer for markov decision tasks via parameter matching. *Workshop of the UK Planning and Scheduling Special Interest Group*.

Taylor, M. E., Jong, N. K., & Stone, P. (2008). Transferring instances for model-based reinforcement learning. *AAMAS 2008 Workshop on Adaptive Learning Agents and Multi-Agent Systems*.

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