Transfer of Task Representation in Reinforcement Learning using Policy-based Proto-value Functions [*]

(Short Paper)

Eliseo Ferrante
Dept. of Electronics and Information,
Politecnico di Milano
piazza Leonardo Da Vinci, 32, 20133 Milan, Italy
eliseo.ferrante@mail.polimi.it

Alessandro Lazaric
Dept. of Electronics and Information,
Politecnico di Milano
piazza Leonardo Da Vinci, 32, 20133 Milan, Italy
lazaric@elet.polimi.it

Marcello Restelli
Dept. of Electronics and Information,
Politecnico di Milano
piazza Leonardo Da Vinci, 32, 20133 Milan, Italy
restelli@elet.polimi.it

ABSTRACT

Reinforcement Learning research is traditionally devoted to solve single-task problems. Therefore, anytime a new task is faced, learning must be restarted from scratch. Recently, several studies have addressed the issue of reusing the knowledge acquired in solving previous related tasks by transferring information about policies and value functions. In this paper, we analyze the use of proto-value functions under the transfer learning perspective. Proto-value functions are effective basis functions for the approximation of value functions defined over the graph obtained by a random walk on the environment. The definition of this graph is a key aspect in transfer transfer problems in which both the reward function and the dynamics change. Therefore, we introduce *policy-based* proto-value functions, which can be obtained by considering the graph generated by a random walk guided by the optimal policy of one of the tasks at hand. We compare the effectiveness of *policy*-based and standard proto-value functions, on different transfer problems defined on a simple grid-world environment.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Learning

General Terms

Spectral graph theory

Keywords

Reinforcement Learning, Transfer Learning, Proto-value functions

1. INTRODUCTION

Reinforcement Learning (RL) [9] is a very general learning paradigm. Nonetheless, for each new task, the learning is restarted from scratch and this may lead to prohibitive complexity (*curse of dimensionality*). The goal of transfer learning is to design algorithms able to extract and reuse the knowledge learned in one or more tasks to efficiently develop an effective solution for a new task. Many works of transfer in RL relied on the option framework [8, 4], by learning options that can be profitably reused in a wide range of tasks. Another category of transfer approaches involves transfer of value functions and policies across tasks defined over different domains (i.e., with different state and action spaces). The approach proposed in [10] uses a transfer functional to map the value function of the source task to a corresponding value function for the target task. Finally, in [11], the transfer of policies via inter-task mappings across tasks defined on arbitrary domains is considered.

In this paper, we focus on the problem of learning and transferring the common representation underlying the optimal value functions of a set of related tasks. In particular, we build on the proto-value functions (PVF) framework [6], that provides a technique for the automatic extraction of a set of basis functions based on spectral graph theory. Unlike other works on transfer with PVFs [3], in which matrix perturbation theory and Nyström methods are adopted for domain transfer problems, we focus on the problem in which both state and action spaces are shared across all the tasks but both the dynamics and the reward function may vary. The PVF method is defined under the assumption that the function to be approximated can be effectively represented on a graph. Therefore, in the context of transfer, it is important to build a graph that captures both the dynamics and the reward function. For this reason, we introduce policy-based PVFs obtained from a graph built by considering the optimal policy of one of the *source* tasks at hand.

The rest of the paper is organized as follows. In Section 2, we review the proto-value functions framework. In Section 3, we give the definition of the transfer problem and we introduce the policy-based proto-value functions. In Section 4, we compare original PVFs to policy-based PVFs in a grid world transfer problem. Finally, in Section 5 we conclude and we discuss some possible future directions.

2. PROTO-VALUE FUNCTIONS

RL problems are formally defined as a Markov Decision Process (MDP), described as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, where $\mathcal{S}$ is

[*]
Cite as: Transfer of Task Representation in Reinforcement Learning using Policy-based Proto-value Functions (Short Paper), Eliseo Ferrante, Alessandro Lazaric, Marcello Restelli, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. 1329-1332.

the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{T}^a_{ss'}$ is the transition model that specifies the transition probability from state s to state s' when action a is taken, and $\mathcal{R}^a_s$ is the reward function. The policy of the agent is defined as a function $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0; 1]$ that prescribes the probability to take an action in each state. The goal of the agent is to learn the optimal policy π^* that maximizes the reward received in the long run. Furthermore, it is possible to compute the optimal action-value function $Q^*(s, a)$, that is, the expected sum of discounted rewards in each state obtained by following π^*. In many practical applications it is unfeasible to store the value function with a distinct value for each state-action (*curse of dimensionality*). The most common approach to face this problem is to use linear function approximators for the action value function:

$$\widehat{Q}(s, a) = \sum_{i=1}^{k} \phi_i(s, a)\theta_i,$$

where $[\theta_1, \ldots, \theta_k]$ is the weights vector to be learned, $[\phi_1 \ldots \phi_k]$ are the basis functions, where $\phi_i : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}, i \in 1, \ldots, k$ is a basis function defined on the state-action space. Most of the RL algorithms consider a set of hand-coded basis functions (e.g., RBFs), while the learning process learns the weights vector that minimizes the approximation error. On the other hand, the PVF framework [6] provides an algorithm for the automatic extraction of a set of basis functions based on spectral graph theory [2]. The basic intuition is that the value functions can be approximated by a set of orthonormal basis computed from the Laplacian of the graph obtained by a random walk on the environment at hand. The Representation Policy Iteration (RPI) algorithm consists in two phases: the representation learning phase and the control learning phase. In the representation learning phase, PVFs are extracted. In particular, an undirected or directed weighted graph $G = \langle N, E, W\rangle$ that reflects the topology of the task is built, where N is the set of nodes (i.e. either states or state-action pairs), E the set of edges and W the matrix containing the weights w_{uv} between each pair of nodes $u, v \in N$. Subsequently, spectral analysis of the graph is performed, extracting the eigenvectors of some graph operator, that is the PVFs. The most used graph operator is the graph Laplacian, that in turns can be defined in many ways, with the most common being the normalized Laplacian definition:

$$\mathcal{L} = I - D^{-1/2} W D^{-1/2},$$

where I is the identity matrix and D is a diagonal matrix called the *valency matrix*, whose entries d_{uu} contain the degree of a node $d_{uu} = \sum_{v \in N} w_{uv}$. Finally, in the control learning phase, LSPI [5] is used to learn the weights vector.

One of the most critical part in the previous algorithm is the construction of the graph used to extract the PVFs. The agent explores the environment using a sampling policy π^σ and a set of sample transitions $\langle s, a, s', r\rangle$ is collected. In general, the sampling policy is the random policy $\pi^\sigma = \pi^R$. Subsequently, a graph can be constructed alternatively by taking into account the estimated transition model of the problem, i.e. by considering the random walk P (containing the probability of going from state s to s') computed as:

$$W \equiv P_s^{s'} = \sum_{a \in \mathcal{A}} \pi^\sigma(s, a)\mathcal{T}^a_{ss'},$$

or by defining a suitable distance function $d(s_i, s_j), s_i, s_j \in \mathcal{S}$ and using an exponential weighting $e^{-d(s_u, s_v)^2}$ to assign weights to each edge on the graph $(u, v) \in E$. In this paper, we focus on graphs defined over the state-action space [7].

3. POLICY-BASED PVFS FOR TRANSFER

We consider the following transfer learning problem. Let $\langle \mathcal{S}, \mathcal{A}, \mathbb{T}, \mathbb{R}\rangle$ with $\mathbb{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ and $\mathbb{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$ be a family of MDPs sharing the same state space $\mathcal{S}$ and action space $\mathcal{A}$ but with different transition models and reward functions. We define two probability distributions: $\mathcal{Q}_{\mathcal{T}} : \mathbb{T} \rightarrow [0, 1]$ and $\mathcal{Q}_{\mathcal{R}} : \mathbb{R} \rightarrow [0, 1]$, used to select the transition model and reward function respectively. In particular, we consider the scenario with one source task, from which the representation knowledge is extracted, and one or more target tasks, where learning exploiting transfer occurs. Both source and target tasks are drawn according to $\mathcal{Q}_{\mathcal{T}}$ and $Q_{\mathcal{R}}$. In the following, we distinguish between *goal transfer* and *dynamics transfer*. In goal transfer we assume that the goal changes between the source and the target task, whereas the transition model remains unchanged. In dynamics transfer the two tasks share the same goal, whereas the dynamics is different.

3.1 Policy-based Proto-value Functions

The basic assumption underlying the original PVF framework (whose PVFs will be denoted as *dynamics-based* PVFs) is that the optimal value function V^* can be represented on the graph G obtained through a random walk following a fully random policy π^R on the task. In goal transfer, we assume that all the optimal value functions can be well approximated by the basis of the dynamics-based graph, i.e., the one that captures the dynamics of the environment but completely ignores the reward functions. However, in the more general case, both the transition model and the reward function may vary. Hence, PVFs should be extracted taking into account both of them. Unfortunately, it is not possible to use the reward function in the construction of graphs directly. Nonetheless, it is possible to bias the exploration of the environment towards the optimal policy of the source task, thus indirectly taking its reward function into account. This yields to a new sampling policy which is different from the random policy $\pi^\sigma \neq \pi^R$. As a result, we can compute a new kind of graph based on the new sampling policy. Such graph will be denoted as *policy-based graph*.

We consider a task with a completely connected but stochastic dynamics consisting of 5 states and with the goal in the center, denoted with `X`. A dynamics-based graph would be the one in Figure 1-*(top)*. On the other hand, a policy-based graph should take the goal into account. To strengthen the policy contribution, we compute its t-th power to get the distribution after t steps and we obtain the graph in Figure 1-*(center)*. This graph captures information about the goal, since only the weights on the edges directed towards the goal are very high. However, dynamics information is completely lost. Hence, we introduce averaged graphs (Figure 1-*(bottom)*). This new graph keeps information about both the goal and the dynamics.

Following these observations, we propose the state-action graph construction method. The method takes *policy bias factor* δ as input parameter used to adjust the bias towards the optimal policy π^* of the source task in the construction of the graph. In particular, we set the sampling policy π^σ

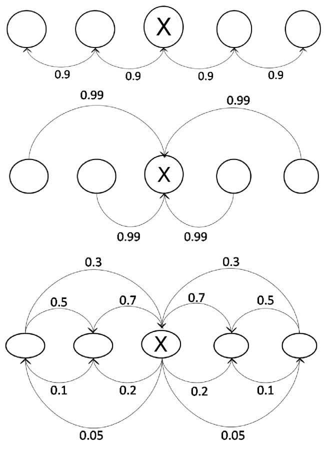


Figure 1: An example of dynamics-based graph (top), policy-based graph (center) and averaged-graph (bottom)

to π^R with probability $1-\delta$ and to π^* with probability δ. When $\delta = 0$, the optimal policy is not used and dynamics-based PVFs are extracted. On the other hand, when δ is close to 1, the policy contribution is very strong and the walk (and hence the graph) is strongly leaned towards the goal. The sampling policy π^σ is used to compute the initial weight matrix for the state-action graph [7] by setting each entry to

$$W((s_i,a_k),(s_j,a_l)) = \mathcal{T}_{s_i s_j}^{a_k} \pi^\sigma(a_l|s_j),$$

$\forall s_i, s_j \in \mathcal{S}, a_k, a_l \in \mathcal{A}$. Subsequently, the graph averaging is done. Here, we use the *time-averaged transition probability matrix* [1] with discounting, thus leading to the graph

$$W_{\widetilde{t}} = \sum_{i=1}^{t} \frac{W^i(1-\gamma)\gamma^{i-1}}{1-\gamma^t}.$$

Finally, the graph needs to be symmetrized, especially when using the Laplacian as defined on undirected graphs. With our method, the amount of biasing towards the policy is controlled by the δ parameter.

4. EXPERIMENTAL RESULTS

In order to compare the learning performance of dynamics- and policy-based PVFs in transfer problems, we consider the three-rooms grid-world domain used in [6]. This consists in a stochastic environment, where each action is successful with probability 0.9, whereas with probability 0.1 the agent stands still. In all experiments we use state-action graphs and 15,000 samples during both the representation and learning phase. LSPI parameters are: discount factor $\gamma = 0.9$, the maximum number of iterations is 16 and $\epsilon = 0,001$.

4.1 Goal Transfer Experiment

We first perform goal transfer experiments, in which all the tasks share the same dynamics but have different reward functions. We extract a total of 24 state-action dynamics-based PVFs. We would expect dynamics-based PVFs to perform well, since they effectively capture the dynamics of the task. In the first experiment we consider one source task with the goal in the upper right angle of the grid-world, and one target task with the goal in the upper left corner. The exact value function of the source task presents some nonlinearities in the regions that are close to the walls. In [6] it is shown that dynamics-based PVFs can effectively capture those nonlinearities. Learning performance are reported in Figure 2. In this case, the two tasks are completely unrelated in terms of their goal and their optimal policy, whereas their dynamics is the same. Results show that dynamics-based PVFs can effectively achieve goal transfer in this case.

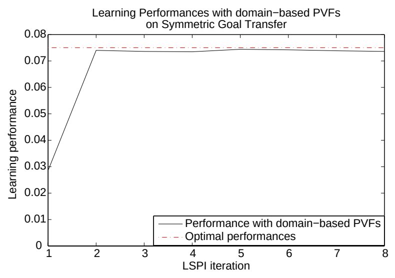


Figure 2: Learning performance in goal transfer obtained by moving the goal in a symmetric position

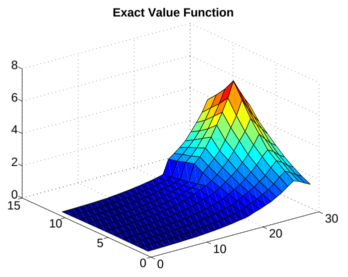
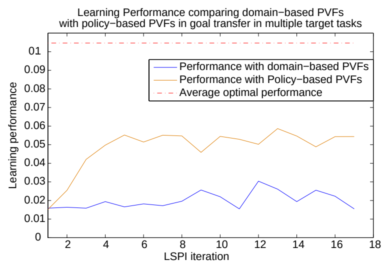
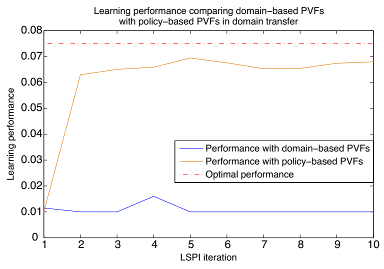


Figure 3: A value function with nonlinearities due to the reward function and not the transition model.

We now consider a goal transfer problem in which the reward functions are obtained by perturbation of the reward function of a source task. In the target task the goal is placed at $(8, 27)$, close to the upper-right corner and the corresponding optimal value function is reported in Figure 3. It is interesting to notice that, in this case, the value function has many nonlinearities that are not related to the dynamics of the environment but that are generated by the reward function. Thus, the dynamics-based PVFs are likely to fail to approximate the optimal value functions of target tasks that share these characteristics with the source task. On the other hand, the policy-based PVFs generated from the graph obtained by biasing the exploration towards the optimal policy of the source task, better capture the particular shape of the value functions to approximate.

We consider 9 target tasks where the goal is moved around the goal of the source task (including itself). We plot the average learning performances of dynamics-based PVFs and policy-based PVFs obtained with policy bias factor $\delta = 0.75$, and we compare them with the average optimal performance. As it can be noticed in Figure 4-*(left)*, policy-based PVFs performs better than dynamics-based PVFs. This is because policy-based PVFs help to capture and transfer the information about the nonlinearities close to the goal.





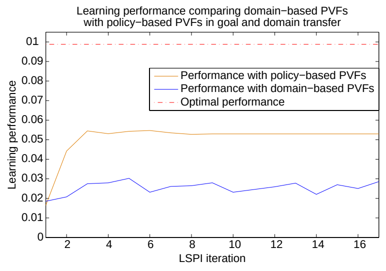


Figure 4: Learning performance in goal (*left*), dynamics (*center*), and goal-dynamics (*right*) transfer

4.2 Dynamics Transfer Experiment

We consider a dynamics transfer experiment in which tasks are strictly related in terms of their optimal policy but with different dynamics. Furthermore, we assume that the shared representation can be compactly extracted by using a low number of PVFs (6 in the experiment). The goal is placed in the upper-right corner in both tasks. The source task has a dynamics which is "tilted" in the opposite direction of the goal. This means that the probability of success of actions that aims in the opposite direction of the goal are higher than the one aiming towards the goal. In the target task, the dynamics is unchanged, with actions having the same probability of success. Figure 4-*(center)* compares the learning performance of dynamics-based PVFs with those of policy-based PVFs obtained with $\delta = 0.75$. As it can be noticed, policy-based PVFs outperform dynamics-based PVFs in this transfer experiment. This is because the two tasks share the representation about their optimal policy, and this can be better captured using policy-based PVFs.

4.3 Goal-Dynamics Transfer Experiment

In the final experiment we consider a source task whose dynamics is tilted towards the bottom-left direction and a goal at $(8, 27)$. We consider three target tasks: *(i)* standard untilted dynamics and the goal at $(8, 28)$, *(ii)* dynamics tilted towards north and the goal at $(9, 27)$, and *(iii)* dynamics tilted towards south and the goal at $(9, 26)$. We consider 6 PVFs and $\delta = 0.5$. Figure 4-*(right)* compares the learning performance of dynamics-based PVFs with those of policy-based PVFs. Optimal performances are reported as well. Also in this case, policy-based PVFs outperform dynamics-based PVFs. By transferring information about the optimal policy, we are able both to counter-balance the effect of the nonlinearity close to the goal and the change in the dynamics.

5. CONCLUSIONS

In this paper, we focused on the problem of transfer in terms of the extraction of a set of basis functions that can be profitably used for the approximation of the optimal value functions of a set of related tasks. In particular, building on the PVF framework, we showed that, in the transfer problem, the graph construction method should take into consideration both the transition model and the reward function. Hence, we proposed policy-based PVFs, extracted using an averaged discounted graph obtained through an exploration biased towards the optimal policy of the source task. In case of goal transfer, dynamics-based PVFs achieve effective transfer whenever the optimal value functions has nonlinearities directly related with the intrinsic structure of the environment. On the other hand, when optimal value functions present some nonlinearities *caused* by the reward function, the use of the optimal policy of the source target leads to policy-based PVFs that can better approximate the target functions. Furthermore, the dynamics transfer experiments showed that the policy-based PVFs can improve transfer capabilities in cases where the source and the target task share a similar optimal policy, but the dynamics are different.

A direction for future work is to define a method to incrementally adapt the initial set of PVFs according to the target tasks at hand, thus improving their approximation capabilities on the tasks that must be actually solved.

6. REFERENCES

[1] A. T. Bharucha-Reid. *Elements of the Theory of Markov Processes and Their Applications.* Dover Publications, 1997.

[2] F. R. Chung. *Spectral Graph Theory.* Amer Mathematical Society, 1997.

[3] K. Ferguson and S. Mahadevan. Proto-transfer learning in markov decision processes using spectral methods. In *ICML Workshop on Transfer Learning*, 2006.

[4] G. Konidaris and A. G. Barto. Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, pages 895–900, 2007.

[5] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *JMLR*, 4:1107–1149, 2003.

[6] S. Mahadevan and M. Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *JMLR*, 8:2169–2231, 2007.

[7] S. Osentoski and S. Mahadevan. Learning state-action basis functions for hierarchical mdps. In *ICML '07*, pages 705–712, 2007.

[8] T. J. Perkins and D. Precup. Using options for knowledge transfer in reinforcement learning. Technical report, University of Massachusetts, Amherst, MA, USA, 1999.

[9] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998.

[10] M. E. Taylor, P. Stone, and Y. Liu. Value functions for RL-based behavior transfer: A comparative study. In *AAAI*, pages 880–885, July 2005.

[11] M. E. Taylor, P. Stone, and Y. Liu. Transfer learning via inter-task mappings for temporal difference learning. *JMLR*, 8:2125–2167, 2007.