Exploiting Different Levels of Abstractions for Sample Efficient Reinforcement Learning

Roberto Cipollone, Giuseppe De Giacomo, Marco Favorito, Luca Iocchi and Fabio Patrizi

Sapienza University of Rome, Italy

Abstract

One of the main issues that limit the application of Reinforcement Learning to real robots is the large number of samples that common algorithms require. In this work, we specifically address the issue of sample efficiency. In the learning framework that we propose, a complex task on a target environment is simplified via multiple levels of abstraction, through the simulation of simplified models. These levels are intentionally coupled in a loose way, as they only require a mapping function from the states of one model to those of its abstraction. This hierarchy allows us to exploit the value function learnt from the coarser levels and guide the exploration of the learner, by means of reward shaping techniques. We assess the correctness of the approach and we experimentally show its effectiveness at improving convergence speed in case of sparse rewards.

1. Introduction

The large number of samples that most Reinforcement Learning (RL) algorithms require strongly limits its applicability to robotics, where the interaction with the environment is costly and comes with associated risks. This work is motivated by the need of improving sample efficiency in RL. We aim at developing a technique which is particularly effective with sparse rewards.

Following a long recognized approach [1, 2], we consider state abstractions and hierarchical decompositions, where the original problem is simplified into its essential components. In particular, we allow the original task to be simplified into one or more levels of abstraction. The novelty of this approach lies in how such abstractions should be related and exploited for learning. In particular, we adopt reward shaping techniques to reuse the information collected from abstract environments, and we ensure that the desirable optimality guarantees are preserved, even in the episodic setting.

As required, this abstraction hierarchy allows to transfer part of the training effort from the costly environments to the simulated abstractions. Specifically, our method allows to learn each of those models with a distinct algorithm. In fact, it can be coupled with any off-policy RL algorithm, appropriate for a specific state-action space, even those designed for Neural Networks.

AIRO 2021, workshop of the 20th International Conference of the Italian Association for Artificial Intelligence

2. The framework

Consider an environment in which experience is costly to obtain. This might be a realistic simulation of the agent's interaction, or an actual environment in which a physical robot is acting. This is our *ground* MDP $\mathcal{M}_0$, that we aim to solve while minimizing the number of interactions with the environment. Instead of learning on this MDP directly, we choose to solve a simplified, related problem, that we call the *abstract* MDP[1]. Once solved, its solution will be used as a heuristic to speed up learning in the target MDP. This idea is not limited to a single abstraction. Indeed, we consider a hierarchy of related MDPs $\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_n$, of decreasing difficulty, where the experience acquired by an expert acting in $\mathcal{M}_i$ can be exploited to speed up learning in the previous one, $\mathcal{M}_{i-1}$.

Associated to each MDP abstraction $\mathcal{M}_i$, we also define a *mapping* function f_i as a surjection $f_i : \mathcal{S}_i \to \mathcal{S}_{i+1}$, which maps states of each model $\mathcal{M}_i$ to states of its direct abstraction $\mathcal{M}_{i+1}$. Since each MDP in the sequence should be easier to solve than it's predecessor, we know that $|\mathcal{S}_i| \geq |\mathcal{S}_{i+1}|$. Therefore, the mapping induces a partition over $\mathcal{S}_i$, where multiple states are mapped through f_i to a single state in $\mathcal{M}_{i+1}$.

Starting from an MDP to simplify, abstract models and mapping functions are obtainable by devising a suitable relaxation of the environment dynamics. For example, in a navigation scenario, an abstraction could contain actions that allow to just "leave the room", instead of navigating with lower-level controls. Importantly, the relation between such different action spaces doesn't need to be modelled. Note that the connection between a model and its abstraction is intentionally loose, so that little constraints are posed in the definition of each mapping.

2.1. Exploiting the knowledge

Consider a hierarchy of abstractions $\mathcal{M}_0, \dots, \mathcal{M}_n$ together with their mapping functions $f_0, \dots, f_{n-1}$. The learning process proceeds incrementally, training in order from the easiest to the hardest one. At each level $\mathcal{M}_i$, we can exploit the knowledge acquired from its abstraction $\mathcal{M}_{i+1}$, in order to speed up learning. We do so by applying a RL technique called Reward Shaping.

Reward shaping (RS) [4] refers to the modification of the original rewards of an MDP, through the addition of a reward *shaping function*, i.e. $R^s(s, a, s') \coloneqq R(s, a, s') + F(s, a, s')$. *Potential-Based Reward Shaping* [5] (simply called "Reward Shaping" from now on), defines the shaping function in terms of a potential function $\Phi : \mathcal{S} \to \mathbb{R}$, as follows:

$$F(s, a, s') \coloneqq \gamma \Phi(s') - \Phi(s) \tag{1}$$

In the infinite horizon case, this choice guarantees that the optimal policy of the MDP with reward function R^s is the same as the one for the original MDP. Since the optimum is not modified, reward shaping can only provide an initial bias to the learning algorithm. In fact, it has been shown in [6] that learning under the shaping rewards of Equation (1) is equivalent to learning under a modified Q-function initialization: $Q_0'(s, a) \coloneqq Q_0(s, a) + \Phi(s)$.

Now, assume that model $\mathcal{M}_{i+1}$ has been solved already, and let $V^*_{i+1} : \mathcal{S}_{i+1} \to \mathbb{R}$ be its optimal value function. In order to speed up learning, we apply reward shaping to $\mathcal{M}_i$, with the

[1]We follow the nomenclature from [3].

following potential:

$$\Phi_i(s) := V^*_{i+1}(f_i(s)) \qquad \text{for } s \in \mathcal{S}_i \tag{2}$$

With this choice, the potential of a state is evaluated according to how desirable the corresponding state in the abstract model is. This is beneficial, as high potentials are associated to high state-action value initializations.

2.2. Policy invariance

Potential-Based Reward Shaping has been explicitly designed not to alter the optimal policies. In fact, in case of an infinite horizon, as mentioned above, but also when episodes always terminate in a zero-reward absorbing state, this property is guaranteed [5]. However, in RL, we often organize learning in episodes of finite length, so to diversify the agent experiences. Thus, in the episodic setting, these guarantees do not hold anymore, and the optimal policy may be altered [7].

The solution proposed by [7] is to assume, for every terminal state, the null potential $\Phi(s_n) = 0$, as this would preserve the original returns. However, this is not always a desirable solution.

First, let's consider a Goal MDP, specifically, one with a null reward function everywhere, except when transitioning to a distinct goal state. Regardless of the potential, we can show that all finite trajectories which do not contain the goal state are associated to the same return. This means that the return-invariant RS of [7] is not able to provide a persistent exploration bias to the agent, as it cannot guide it toward a goal that the agent is not able to reach initially. Our experiments also confirmed that the choice $\Phi(s_n) = 0$ did not help to converge on very sparse rewards (this is in line with the experiments from [8]).

Our solution Since we deliberately adopt a form of RS which is not return invariant, we devised a technique to recover the original optimality guarantees. This method can be integrated with any off-policy RL algorithm. We just consider Q-learning for the present discussion.

Let's consider $\mathcal{M}_i = \langle \mathcal{S}_i, \mathcal{A}_i, T_i, R_i, \gamma_i \rangle$, the MDP at the ith level of abstraction. We define $\mathcal{M}^s_i = \langle \mathcal{S}_i, \mathcal{A}_i, T_i, R^s_i, \gamma_i \rangle$ as the MDP associated to $\mathcal{M}_i$, with the following reward function:

$$R^s_i(s, a, s') = R_i(s, a, s') + \Phi_i(s') - \Phi_i(s) \tag{3}$$

where Φ_i is defined according to (2). As we don't need to require that $\mathcal{M}^s_i$ generates the same returns as $\mathcal{M}_i$, the missing γ_i term from the classic RS equation (1) is not an issue. Rather, it has been even considered experimentally helpful [9, chap. 2].

Our method proceeds as follows. When learning on the ith level of abstraction, we perform updates on two distinct Q-function estimates:

- The first, that we call the *active agent*, is in charge of estimating Q^{s*}, i.e. the optimal Q-value function of $\mathcal{M}^s_i$, the MDP with rewards from (3).
- The second, that we call the *passive agent*, is an estimate for Q^*, i.e. the optimal Q-value function of the original MDP $\mathcal{M}_i$.

The active agent is the only that executes actions in the environment. Since its advantage-based policy depends on Q^{s*}, with respect to this agent, the algorithm is exactly Q-learning. The second estimate, instead, is updated from the same state-action transitions as those observed by the active agent, with the only difference that rewards are generated from the original $R_i(s, a, s')$. Most importantly, both estimates converge to the desired quantities, as we state here:

Theorem 1. *Let $\hat{Q}_i^s$ and $\hat{Q}_i$ be, respectively, the Q-value function estimations of $\mathscr{M}_i^s$ and $\mathscr{M}_i$, updated as described above. Then, under the same assumptions as those for the convergence of the Q-learning algorithm, both $\hat{Q}^s$ and $\hat{Q}_i$ converge to Q^{s*} and Q^*, respectively.*

Specifically, this means that the passive agent converges to the optimal policy for the original task. Therefore, its value function $V_i^*(s) = \max_{a \in \mathscr{A}_i} Q_i^*(s, a)$ can be used, in turn, in the computation for the potential in the next level of abstraction to train. Finally, when training on the ground model $\mathscr{M}_0$, we consider Q_0^* the desired result of training.

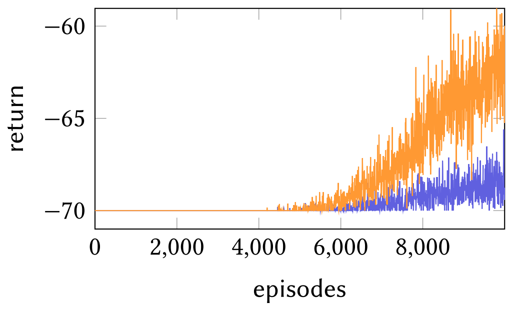
3. Experiments

In order to show the practical applicability of the proposed method in robotics domains, we consider the following behaviour in a office-like scenario: the agent should reach the entrance of each room in sequence; if a door is open, the agent should enter that room and interact with the person inside, if present. The agent is only positively rewarded only when the task is completed.

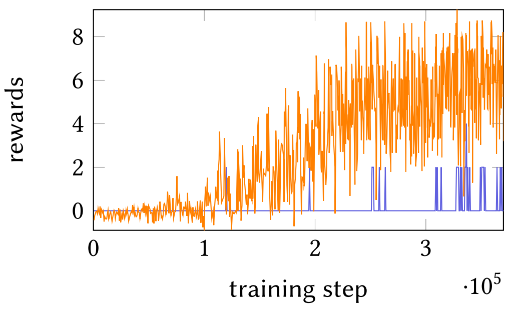
Temporally-extended goals Interesting behaviours always contain some constraints like the ones above, with the form "if this happened *then* ...". Naively encoding such tasks for RL would produce non-Markovian reward functions. Instead, the correct approach is to adopt the techniques from [10, 11, 12, 13], which provide an appropriate reduction to an MDP. The resulting model has a composite state space $\mathscr{S}_i \times \mathscr{Q}$, where the original states $\mathscr{S}_i$ are specific of each abstraction level, and $\mathscr{Q}$ tracks the relevant information with respect to the task. For our purposes, we may just consider them as MDPs. We've chosen such class of processes as they are a perfect example of sparse rewards with a slow convergence. Furthermore, they they are really relevant and applicable to robotics, where non-trivial behaviours are most interesting.

Abstraction levels We choose to model the environment dynamics in three levels of abstraction $\mathscr{M}_0, \mathscr{M}_1, \mathscr{M}_2$. In the most abstract MDP, $\mathscr{M}_2$, the states represent few interesting locations, such as doors and rooms. The actions $\mathscr{A}_2 \coloneqq \{Go_1, \dots, Go_n, Interact\}$ allow to reach each of those locations in one step or to start an interaction. In $\mathscr{M}_1$, instead, the agent can move in discrete steps along the four cardinal directions within a simplified map. Finally, at $\mathscr{M}_0$, we preserve the metric structure of the environment, and the robot state $(s_x, s_y, s_\theta) \in \mathscr{S}_0$ is constituted by the agent's mobile base position and orientation in space. Actions are now relative motions on the plane. The mapping functions f_1, f_0 are simple discretizations of the plane and associations to the interesting locations of $\mathscr{M}_0$, respectively.

Results The plot in Figure 1a compares the convergence speed of a passive agent, that is guided via reward shaping with our approach (orange line), with the baseline Q-learner (blue line). Both are trained with the same parameters, for the same task. Rewards from this plot are



(a) Training convergence on $\mathscr{M}_1$. Cumulative discounted return, averaged over 10 runs.



(b) Training convergence on $\mathscr{M}_0$. Episode total rewards, averaged over 5 runs.

Figure 1: Convergence speed for an agent without (blue) and with reward shaping applied (orange).

negative, as they give a reasonable improvement to the performances of the baseline [14]. In Figure 1b we show the same training comparison for the more complex MDP level $\mathscr{M}_0$, which was trained with Dueling DQN [15]. The advantage of the informed agent is much more evident in this last experiment, where the large state space contributes to the sparsity of rewards.

4. Related works

The field of Hierarchical RL specifically studies efficiency of algorithms in presence of abstractions. Some classic approaches are MAXQ [16], HAM [17] and the Options framework [18], which are algorithms specifically designed to work with all abstractions. In contrast, our method allow greater flexibility on how solutions for each level should be computed.

In other works, [19, 3, 20], the authors identify possible relations between MDPs and their abstractions. However, the abstractions that we consider here are mostly naive partitioning of contiguous regions, and they would hardly satisfy the assumptions hypothesized in them. Our abstractions resemble those of [21]. Simplified models involving both states and actions have been considered in [22], which defines relations that would be interesting to use when formalizing our hierarchy. Works as [23, 24] explore how abstractions for RS could be learnt automatically. Instead, we focus on the actual effectiveness of shaping. Recently, in [8], the authors increased the efficacy of RS by giving up desirable optimality guarantees.

5. Conclusion

The method that we presented allows to reuse the information learnt by an expert agent from the simulated environments to speed up learning in the target environment. Although some additional modelling effort is required, the abstraction hierarchy can be used to speedup learning in a variety of tasks. Furthermore, each level is allowed to have a distinct action space, most appropriate to each granularity level, and the relations between actions of different levels don't need to be modelled. Finally, as we've seen from our experiments, the method is particularly effective for MDPs with sparse rewards.

References

[1] P. Dayan, G. E. Hinton, Feudal reinforcement learning, in: S. J. Hanson, J. D. Cowan, C. L. Giles (Eds.), Advances in Neural Information Processing Systems 5, NIPS Conference, Morgan Kaufmann, 1992, pp. 271–278.

[2] M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. L. Dean, C. Boutilier, Hierarchical solution of markov decision processes using macro-actions, in: UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, 1998, pp. 220–229.

[3] L. Li, T. J. Walsh, M. L. Littman, Towards a unified theory of state abstraction for mdps, in: ISAIM, 2006, pp. 531–539.

[4] J. Randløv, P. Alstrøm, Learning to drive a bicycle using reinforcement learning and shaping, in: J. W. Shavlik (Ed.), Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Morgan Kaufmann, 1998, pp. 463–471.

[5] A. Y. Ng, D. Harada, S. J. Russell, Policy invariance under reward transformations: Theory and application to reward shaping, in: ICML, San Francisco, CA, USA, 1999, p. 278–287.

[6] E. Wiewiora, Potential-based shaping and q-value initialization are equivalent, JAIR 19 (2003) 205–208.

[7] M. Grzes, Reward shaping in episodic reinforcement learning, in: AAMAS, 2017, p. 565–573.

[8] I. Schubert, O. S. Oguz, M. Toussaint, Plan-based relaxed reward shaping for goal-directed tasks, in: 9th international conference on learning representations, ICLR 2021, virtual event, austria, may 3-7, 2021, OpenReview.net, 2021.

[9] M. Grzes, Improving exploration in reinforcement learning through domain knowledge and parameter analysis, Ph.D. thesis, University of York, 2010.

[10] R. I. Brafman, G. De Giacomo, F. Patrizi, LTLf/LDLf non-Markovian rewards, in: AAAI, 2018, pp. 1771–1778.

[11] R. T. Icarte, T. Klassen, R. Valenzano, S. McIlraith, Using reward machines for high-level task specification and decomposition in reinforcement learning, in: ICML, 2018, pp. 2107–2116.

[12] G. De Giacomo, L. Iocchi, M. Favorito, F. Patrizi, Foundations for restraining bolts: Reinforcement learning with LTLf/LDLf restraining specifications, in: ICAPS, 2019, pp. 128–136.

[13] G. De Giacomo, R. De Masellis, F. M. Maggi, M. Montali, Monitoring constraints and metaconstraints with temporal logics on finite traces, CoRR abs/2004.01859 (2020).

[14] S. Koenig, R. G. Simmons, The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms, Machine Learning 22 (1996) 227–250.

[15] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, N. de Freitas, Dueling network architectures for deep reinforcement learning, in: ICML, volume 48, JMLR.org, 2016, pp. 1995–2003.

[16] T. G. Dietterich, Hierarchical reinforcement learning with the maxq value function decomposition, JAIR 13 (2000) 227–303.

[17] R. Parr, S. Russell, Reinforcement learning with hierarchies of machines, Advances in neural information processing systems (1998) 1043–1049.

[18] R. S. Sutton, D. Precup, S. Singh, Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning, Artif. Intell. 112 (1999) 181–211.

[19] B. Ravindran, A. G. Barto, Model minimization in hierarchical reinforcement learning, in: Abstraction, Reformulation and Approximation, 5th International Symposium, SARA 2002, volume 2371 of *Lecture Notes in Computer Science*, Springer, 2002, pp. 196–211. doi:10.1007/3-540-45622-8_15.

[20] D. Abel, D. Hershkowitz, M. Littman, Near optimal behavior via approximate state abstraction, in: Proceedings of The 33rd International Conference on Machine Learning, volume 48 of *Proceedings of Machine Learning Research*, PMLR, 2016, pp. 2915–2923.

[21] M. Grześ, D. Kudenko, Online learning of shaping rewards in reinforcement learning, Neural Networks 23 (2010) 541–550.

[22] D. Abel, N. Umbanhowar, K. Khetarpal, D. Arumugam, D. Precup, M. Littman, Value preserving state-action abstractions, in: Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, volume 108 of *Proceedings of Machine Learning Research*, PMLR, 2020, pp. 1639–1650.

[23] B. Marthi, Automatic shaping and decomposition of reward functions, in: Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007, volume 227 of *ACM International Conference Proceeding Series*, ACM, 2007, pp. 601–608. doi:10.1145/1273496.1273572.

[24] M. Grześ, D. Kudenko, Multigrid reinforcement learning with reward shaping, in: ICANN, 2008, pp. 357–366.