Speeding up Tabular Reinforcement Learning Using State-Action Similarities

(Extended Abstract)

Ariel Rosenfeld
Bar-Ilan University
arielros1@gmail.com

Matthew E. Taylor
Washington State University
taylorm@eecs.wsu.edu

Sarit Kraus
Bar-Ilan University
sarit@cs.biu.ac.il

ABSTRACT

One of the most prominent approaches for speeding up reinforcement learning is injecting human prior knowledge into the learning agent. This paper proposes a novel method to speed up temporal difference learning by using state-action similarities. These hand-coded similarities are tested in three well-studied domains of varying complexity, demonstrating our approach's benefits.

1. INTRODUCTION

Reinforcement Learning (RL) [1] has had many successes, solving complex, real-world problems. When tackling such problems, the designer must decide how much human knowledge into inject to the system. From the engineering or practical perspective, injecting human knowledge is desirable as it can help improve learning, but only if it is practical to gather or leverage, and only as long as it does not cause the agent to be limited to sub-optimal solutions after training. One may consider this approach as a human designer providing advice to an RL learner as opposed to the common framework in which agents advise people (e.g., [2, 3, 4, 5, 6, 7, 8]).

Many successful RL applications have used highly engineered state features. With the recent successes of DeepRL [9], convolutional neural networks have been shown to successfully learn features directly from pixel-level representations. However, such features are not necessarily optimal, significant amounts of human time is necessary to define the deep neural network's architecture, and a significant amount of data is required to learn the features. Therefore, this paper proposes a different, and potentially complementary, approach, in a 'shallow RL' setting.

Our novel approach, which we name *SASS*, standing for State Action Similarity Solutions, allows the generalization of knowledge across state-action values in the action-value function table by leveraging hand-coded heuristics. While there are many ways of leveraging human knowledge in an RL learner by leveraging demonstrations or direct human knowledge (e.g., inverse reinforcement learning [10, 11], learning from demonstration [12, 13], etc.), SASS focuses on allowing designers to specify *state-action similarities* in a given domain. In order to minimize confounding factors, we consider the simplest representation for temporal difference RL algorithms, a tabular representation of an action-value function, with variants of the well-studied Q-learning algorithm [14].

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

Our approach and its integration with the Q-learning framework provide several desired properties that compare favorably with existing RL methods: 1) SASS is able to significantly speed up the agents' learning process in terms of sample efficiency; 2) Unlike various other generalization techniques, SASS retains the convergence and near-optimal guarantees of the original Q-learning algorithm; 3) SASS is based on an arbitrary designer-defined similarity function that does not assume any specific functional form, as other generalization techniques often do.

We evaluate our methodology in three RL tasks of varying complexity: 1) The "toy" task of simple soccer, providing a basic setting for the evaluation of the proposed approach; 2) A large grid-world task named Pursuit, showing the scale-up of our approach; and 3) The popular game of Mario, exemplifying our approach's benefits in a task with billions of states.

2. THE SASS APPROACH

We define the RL task using a Markov Decision Process (MDP). An MDP is defined as $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ where $\mathcal{S}$ is the state-space; $\mathcal{A}$ is the action-space; $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ defines the transition probability; $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function; and $\gamma \in [0, 1]$ is the discount factor. $\mathcal{T}$ and $\mathcal{R}$ are initially unknown to the agent and the agent seeks to learn a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ that maximizes the expected total discounted reward (i.e., the expected return) while interacting with the environment.

SASS focuses on injecting human knowledge into an RL learner using the notion of similarity. We first formally define the similarity function:

Definition 1. *Let $\mathcal{S}$ and $\mathcal{A}$ be a state-space and an action-space, respectively. A similarity function $\sigma : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ maps every two state-action pairs in $\mathcal{S} \times \mathcal{A}$ to the degree to which we expect the two state-action pairs to have a similar expected return. σ is considered valid if $\forall$ pairs s, a, $\sigma(s, a, s, a) > 0$.*

In this study, we assume that the similarity function can be easily defined by a human designer. The investigation of this issue in a study of human subjects, showing our approach's effectiveness and efficiency with both expert and non-expert designers, will be fully described in future work.

In order to integrate the similarity function within the Q-learning framework, we adopt a previously introduced technique [15], where Q-learning is combined with a spreading function that "spreads" the estimates of the Q-function in a given state to neighboring states, exploiting an assumed spatial smoothness of the state-space. We extend the authors' approach as follows: for each experience $\langle s, a, r, s' \rangle$ that the agent encounters, depending on the similarity function σ, we (potentially) update more than a single $\langle s, a \rangle$ entry in the Q table. Multiple updates, one for each entry $\langle \tilde{s}, \tilde{a} \rangle$ for which

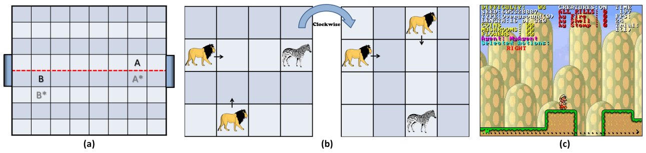


Figure 1: (a) Players in the simple soccer task are A and B; one cell down (A* and B*) are considered similar. (b) Two similar state-action pairs in the Pursuit domain. (c) A state in the Mario AI task where walking or running right are similar (i.e., falling into the gap).

$\sigma(s, a, \tilde{s}, \tilde{a}) > 0$, are performed using the following update:

$$Q(\tilde{s}, \tilde{a}) = Q(\tilde{s}, \tilde{a}) + \alpha\sigma(s, a, \tilde{s}, \tilde{a})\delta \qquad (1)$$

where δ is the temporal difference error term $(r+\gamma max_{a'}Q(s', a')-Q(s, a))$. The above update rule does not compromise the theoretical guarantees of Q-learning (see [16, 17]).

In other words, the update rule states that as a consequence of experiencing $\langle s, a, r, s'\rangle$, an update is made to other pairs $\langle \tilde{s}, \tilde{a}\rangle$ as if the real experience actually was $\langle \tilde{s}, \tilde{a}, r, s'\rangle$ (discounted by the similarity function). We will use the term QS-learning for the above Q-learning-with-$SASS$ interpretation.

Similarity functions can be defined in multiple ways to capture various assumptions and insights about the state-action space. Although people can easily identify similarities in real-life, they are often incapable of articulating sophisticated rules for defining such similarities. Therefore, in the following, we identify and discuss three notable similarity notions:

1) Representational Similarity from the tasks' state-action space. Function approximation [18] is perhaps the most popular example of the use of this technique. The function approximator (e.g., tile coding, neural networks, abstraction, etc.) approximates the Q-value and therefore implicitly forces a generalization over the feature space. See Figure 1 (a) for an illustration.

2) Symmetry similarity seeks to consolidate state-action pairs that are identical or completely symmetrical in order to avoid redundancies. For example, in the Pursuit domain, one may consider the 90°, 180° and 270° transpositions of the state around its center (along with the direction of the action) as being similar (see Figure 1 (b)). However, as the predators do not know the prey's (potentially biased) policy, they can only assume such symmetry exists.

3) Transition similarity can be defined based on the idea of *relative effects* of actions in different states. A relative effect is the change in the state's features caused by the execution of an action. Exploiting relative effects to speed up learning was proposed [19, 20] in the context of model learning. For example, in the Mario domain, if Mario *walks right* or *runs right*, outcomes are assumed to be similar as both actions induce similar relative changes to the state (see Figure 1 (c)). In environments with complex or non-obvious transition models, it can be difficult to intuit this type of similarity.

3. EVALUATION

We evaluate our approach (denoted QS) against regular Q-learning (denoted Q), Q-learning combined with state-space abstraction (denoted QA) and the Dyna algorithm (denoted $Dyna$) in three RL tasks of varying complexity: Simple Soccer [21] (which we implemented in this study), Pursuit [22] (as implemented in [23]) and Mario AI [24] (as implemented in [11]).

Simple Soccer:

QA used a simple distance-based approach, which represented each state according to the learning agent's distance to its opponent and goal.

QS used two major similarity notions: First, *representational similarities* – the agent artificially moves *both players* together across the grid, keeping their original relative distance (see Figure 1). As the players are moved further and further away from their original positions, the similarity estimation gets exponentially lower. Second, *symmetry similarities* – experiences in the upper half of the field are mirrored in the bottom part by mirroring states and actions with respect to the Y-axis and vice-versa. *Transition similarities* were not defined by the expert for this task.

Pursuit:

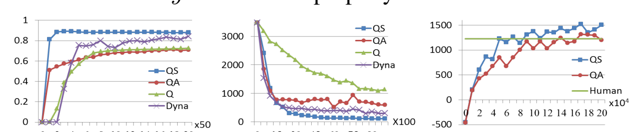
QA was already defined by Brys et al. [23] who implemented tile-code approximation.

QS was defined based on linear differences and angular rotations. Each state is represented as $\langle \Delta_{x_1}, \Delta_{y_1}, \Delta_{x_2}, \Delta_{y_2}\rangle$ where Δ_{x_i} (Δ_{y_i}) is the difference between predator i's x-index (y-index) and the prey's x-index (y-index). A similarity of 1 was set for all states in which the relative positioning of the prey and predators is the same. Symmetry similarities were defined using 90°, 180° and 270° transpositions of the state around its center (along with the direction of the action). Furthermore, experiences in the upper (left) half of the field are mirrored in the bottom (right) part by mirroring states and actions. Transition similarities were defined for all state-action pairs that are expected to result in the same state.

Mario AI:

QA was implicitly defined by the original authors as they had already abstracted the state space.

QS was defined on top of the authors' abstraction. State is defined such that whether Mario can jump or shoot are 2 Boolean variables. Given a state-action pair in which Mario does not jump or shoot, all respective states (with the four variations of these two Boolean variables) were defined as similar to the original pair. Namely, if Mario walks right, then regardless of Mario's ability to shoot or jump, the state-action pair is considered similar to the original one. Symmetry similarities are defined using the mirroring of the state-actions across an imaginary horizontal line that divides the environment in half, with Mario in the middle. As illustrated in Figure 1, regardless of specific state, performing action a (e.g., move right) is assumed similar to using action a+"run" (e.g., run right). In the Mario AI task, due to the huge state-action space, Q-learning without the authors' abstraction will not be evaluated. Furthermore, due to extreme memory requirements in run-time, we were unable to evaluate the $Dyna$ condition properly.



(a) Simple Soccer. (b) Pursuit. (c) Mario AI.

Figure 2: The QS-learning agent outperforms QA-learning, Q-learning and $Dyna$ agents in all three domains.

4. CONCLUSIONS

In this paper, we proposed and extensively evaluated a novel approach for speeding up Q-learning agents using the notion of state-action similarities. Our approach, SASS, and its instantiation within the Q-leaning framework, QS-learning, are shown to significantly speed up an agent's learning process in well-studied domains of varying complexity while accommodating different similarity notions and retaining desired theoretical properties. In future work we will fully describe an empirical investigation of our approach in a study of human subjects, showing our approach's effectiveness and efficiency among designers of different skills and prior knowledge.

REFERENCES

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

[2] Ariel Rosenfeld and Sarit Kraus. Providing arguments in discussions on the basis of the prediction of human argumentative behavior. *TiiS*, 6(4):30:1–30:33, 2016.

[3] Ariel Rosenfeld, Joseph Keshet, Claudia V. Goldman, and Sarit Kraus. Online prediction of exponential decay time series with human-agent application. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands*, pages 595–603, 2016.

[4] Ariel Rosenfeld. Automated agents for advice provision. In *Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI, Buenos Aires, Argentina, July 25-31, 2015*, pages 4391–4392, 2015.

[5] Ariel Rosenfeld, Noa Agmon, Oleg Maksimov, Amos Azaria, and Sarit Kraus. Intelligent agent supporting human-multi-robot team collaboration. In *Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI, Buenos Aires, Argentina, July 25-31, 2015*, pages 1902–1908. AAAI Press, 2015.

[6] Ariel Rosenfeld, Amos Azaria, Sarit Kraus, Claudia V. Goldman, and Omer Tsimhoni. Adaptive advice in automobile climate control systems. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS, Istanbul, Turkey, May 4-8, 2015*, pages 543–551, 2015.

[7] Amos Azaria, Ariel Rosenfeld, Sarit Kraus, Claudia V Goldman, and Omer Tsimhoni. Advice provision for energy saving in automobile climate-control system. *AI Magazine*, 36(3):61–72, 2015.

[8] Priel Levy and David Sarne. Intelligent advice provisioning for repeated interaction. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Stig Ostrovski, Georg Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[10] Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670, 2000.

[11] Halit Bener Suay, Tim Brys, Matthew E Taylor, and Sonia Chernova. Learning from demonstration for shaping through inverse reinforcement learning. In *AAMAS*, pages 429–437, 2016.

[12] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[13] Matthew E Taylor, Halit Bener Suay, and Sonia Chernova. Integrating reinforcement learning with human demonstrations of varying ability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 617–624. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[14] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

[15] Carlos HC Ribeiro. Attentional mechanisms as a strategy for generalisation in the q-learning algorithm. In *Proceedings of ICANN*, volume 95, pages 455–460, 1995.

[16] Csaba Szepesvári and Michael L Littman. A unified analysis of value-function-based reinforcement-learning algorithms. *Neural computation*, 11(8):2017–2060, 1999.

[17] Carlos Ribeiro and Csaba Szepesvári. Q-learning combined with spreading: Convergence and results. In *Procs. of the ISRF-IEE International Conf. on Intelligent and Cognitive Systems (Neural Networks Symposium)*, pages 32–36, 1996.

[18] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC press, 2010.

[19] Nicholas K Jong and Peter Stone. Model-based function approximation in reinforcement learning. In *AAMAS*, page 95. ACM, 2007.

[20] Bethany R Leffler, Michael L Littman, and Timothy Edmunds. Efficient reinforcement learning with relocatable action models. In *AAAI*, volume 7, pages 572–577, 2007.

[21] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *ICML*, volume 157, pages 157–163, 1994.

[22] Miroslav Benda. On optimal cooperation of knowledge sources. *Technical Report BCS-G2010-28*, 1985.

[23] Tim Brys, Ann Nowé, Daniel Kudenko, and Matthew E Taylor. Combining multiple correlated reward and shaping signals by measuring confidence. In *AAAI*, pages 1687–1693, 2014.

[24] Sergey Karakovskiy and Julian Togelius. The Mario AI benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):55–67, 2012.