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 to 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].

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 \( \langle S, A, T, R, \gamma \rangle \) where \( S \) is the state-space; \( A \) is the action-space; \( T : S \times A \times S \rightarrow [0, 1] \) defines the transition probability; \( R : S \times A \rightarrow \mathbb{R} \) is the reward function; and \( \gamma \in [0, 1] \) is the discount factor. \( T \) and \( R \) are initially unknown to the agent and the agent seeks to learn a policy \( \pi : S \mapsto A \) 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 \( S \) and \( A \) be a state-space and an action-space, respectively. A similarity function \( \sigma : S \times A \times S \times A \rightarrow [0, 1] \) maps every two state-action pairs in \( S \times A \) to the degree to which we expect the two state-action pairs to have a similar expected return. \( \sigma \) 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 \( \sigma \), we (potentially) update more than a single \( \langle s, a \rangle \) entry in the Q table. Multiple updates, one for each entry \( \langle s, a \rangle \) for which...
used a simple distance-based approach, which represented each
state according to the learning agent’s distance to its opponent and
goal.

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

\[
Q(\tilde{s}, \tilde{a}) = Q(\tilde{s}, \tilde{a}) + \alpha \sigma(s, a, \tilde{s}, \tilde{a}) \delta
\]

where \( \delta \) 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 \((s, a, r, s')\), an update is made to other pairs \((\tilde{s}, \tilde{a})\)
as if the real experience actually was \((\tilde{s}, \tilde{a}, r, s')\) (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. Al-
though 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 redundan-
cies. 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 bi-
ased) policy, they can only assume such symmetry exists.

3) **Transition similarity** can be defined based on the idea of rel-
ative 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 (de-
noted QA) and the Dyna algorithm (denoted Dyna) in three RL
tasks of varying complexity: Simple Soccer [21] (which we imple-
mented 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.

**Pursuit:**
QS used two major similarity notions: First, **representational simi-
larities** – 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. Sec-
ond, **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.

**Mario AI:**
QS 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 \((\Delta x_1, \Delta y_1, \Delta x_2, \Delta y_2)\) where \(\Delta x_1\)
and \(\Delta y_1\) is the difference between predator’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.

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).

**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-learning framework, QS-learning, are shown to sig-
nificantly speed up an agent’s learning process in well-studied do-
mains of varying complexity while accommodating different simi-
larities notions and retaining desired theoretical properties. In fu-
ture work we will fully describe an empirical investigation of our
approach in a study of human subjects, showing our approach’s ef-
ectiveness and efficiency among designers of different skills and
prior knowledge.
REFERENCES


