Information Sciences 275 (2014) 101–114

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Oppositional extension of reinforcement learning techniques

M. Mahootchi [a], H.R. Tizhoosh [b,*], K. Ponnambalam [b]

[a] Industrial Engineering and Management Systems, Amirkabir University of Technology, Teheran, Iran
[b] Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada

ARTICLE INFO

Article history:
Received 2 August 2011
Received in revised form 14 October 2013
Accepted 9 February 2014
Available online 19 February 2014

Keywords:
Opposition-based learning
Reinforcement learning
Q-learning
Sarsa
Reservoir management
Grid world

ABSTRACT

In this paper, we present different opposition schemes for four reinforcement learning methods: Q-learning, Q(λ), Sarsa, and Sarsa(λ) under assumptions that are reasonable for many real-world problems where type-II opposites generally better reflect the nature of the problem at hand. It appears that the aggregation of opposition-based schemes with regular learning methods can significantly speed up the learning process, especially where the number of observations is small or the state space is large. We verify the performance of the proposed methods using two different applications: a grid-world problem and a single water reservoir management problem.

1. Introduction

Reinforcement learning (RL) includes various learning techniques in which single or multiple agents can be trained through interaction with stochastic or deterministic environments such that an optimal or near-optimal policy can be extracted. The most advantageous aspect of these techniques is their model-free basis, which makes them very attractive and useful for real world and online training applications. However, to converge to a steady state [20], all states and actions must be infinitely visited. This is usually not possible in large-scale applications. Opposition-based learning (OBL), first introduced by Tizhoosh [17,18], might be an effective way to speed up the learning process [16,12,13,8]. The idea underlying this methodology is to use the inherent oppositional relationships in the system to update the agent's knowledge more frequently. Where the state space is large, using function approximation (FA) or knowledge extraction techniques can be useful in speeding up the learning process. We will also demonstrate how FA techniques can be useful in finding the opposite action/state and how they can accelerate the learning process efficiently.

In this paper, two different applications are investigated to illustrate the benefit of the proposed methods. The first is a grid-world problem, which is quite a popular problem in RL research, especially for navigation purposes [4]. The second is a single water reservoir application, in which the target is to find a policy such that the predefined objective function is optimized in a long- or short-term perspective. Reservoir management can be very complex because multiple reservoirs are operated under highly stochastic conditions. Whereas in previous work (e.g., [8]) we introduced the notion of opposition (only type-I opposites) in the Q-learning algorithm, in this paper we extend our work to TD(λ) methods such as Q(λ) and Sarsa(λ) and use type-II opposites.

* Corresponding author. Tel.: +1 519 888 4567.
E-mail addresses: mmahootchi@aut.ac.ir (M. Mahootchi), tizhoosh@uwaterloo.ca (H.R. Tizhoosh), ponnu@uwaterloo.ca (K. Ponnambalam).

http://dx.doi.org/10.1016/j.ins.2014.02.024
0020-0255/

The remainder of this paper is organized as follows: Section 2 briefly overviews four different RL techniques. Section 3 reviews OBL versions of popular techniques in RL, specifically, Q-learning, Sarsa, Q(λ), and Sarsa(λ). Section 4 presents and discusses our experimental results, and Section 5 concludes this paper.

2. Reinforcement learning techniques

The core ideas in RL were derived from the formulation of SDP, in which the transition probabilities (known as a system model) were eliminated [3,5,20]. Every RL technique has four main components: action policy, reward signal, action-value function, and (optionally) a model. The action policy is used to take actions that change the state of the environment. It is a mapping from state to the decision (action), which is usually defined using a probabilistic feature (e.g., policy ϵ-greedy with ϵ as the probability of taking exploratory actions). The reward signal is the immediate or delayed response of the environment to the action taken by the agent. The action-value function, which is defined for every state-action pair, takes into account the accumulative reward from the starting point of learning. In other words, the action-value function, in contrast to the reward function, specifies the gain of the system for a given state-action pair after a long run. Indeed, the action-value functions can be calculated using the reward function accumulated by a discount factor. They can also be available either analytically or through simulation. The model component of RL determines the next state and the reward of the environment based on mathematical functions.

In the following subsections, the four RL techniques investigated in this paper, are explained in brief.

2.1. Q-learning

Q-learning, the most popular online RL technique, was first introduced by Watkins [20]. It is an effective way to find an optimal or near-optimal closed-loop operation policy in a stochastic environment. The agent updates its knowledge after each interaction with the environment (i.e., taking an action a in state $s^t = i$) using the immediate reward, $r^t(i,a)$, it receives and depending on the new situation, ($s^{t+1} = j$), and takes another admissible action based on the new information acquired (e.g., new action-value functions, $Q^t(i,a)$, and updated policy, π). The Q-learning that is used to update the action-value function pertinent to the current state-action pair, is generally formulated as follows, where the total reward has to be maximized:

$$Q^t(i,a) \;:=\; (1-\alpha)Q^t(i,a) + \alpha\left[r^t(i,a) + \gamma \max_{b\in A(j)} Q^{t+1}(j,b)\right]. \tag{1}$$

Here α is the learning parameter and $A(j)$ is the set of admissible actions for state j. This set of admissible actions must be specified before learning starts. In a stochastic environment, this set can be specified either optimistically or pessimistically [7,8].

In Q-Learning, the updating process is performed by looking one step ahead, a process called *single-step corrected truncated return* (CTR) [10]. Conversely, in the other RL techniques discussed below, more than one step ahead can be considered for updating the action-value functions under some conditions.

2.2. Sarsa

In contrast to Q-learning, Sarsa is an on-policy RL technique, which means that the agent follows the same policy π for the current state and all other states. In other words, instead of picking the greedy action for the next state in Eq. (1) to update $Q^t(i,a)$, the action is determined based on the same policy used for taking action in the current state [15]. The general formulation of Sarsa is as follows:

$$Q^t(i,a) \;:=\; (1-\alpha)Q^t(i,a) + \alpha\left[r^t(i,a) + \gamma Q^{t+1}(j,a')\right], \tag{2}$$

where a and a' are two actions chosen from policy π.

2.3. Sarsa(λ)

This RL method is an on-policy learning technique like Sarsa and uses an eligibility trace as a memory variable to update more action-value functions in every iteration [15]. The parameter λ and the discount factor γ play a decaying role in the eligibility trace to put more weight on those states that have been recently visited. The eligibility trace is a function of state s and action a is incremented for the visited state and the action taken in the current iteration and updated with $\gamma\lambda$ for all state-action pairs after editing the action-value functions. This process of learning is performed iteratively using these eligibility traces and continued until the convergence criteria are satisfied [15]. As we have seen in Sarsa, this technique uses a quintuple comprising the taking of an action a in current state $i = s$, the observing of reward r and next state $j = s'$, and the taking of another action a'. The initial values of action-value functions $Q^t(i,a)$, and eligibility traces $e(i,a)$ should be set to zero at the start of learning. One iteration of the Sarsa method is demonstrated in Algorithm 1 [15].