Artificial Intelligence

Tutorial week 3 – Reinforcement learning

COMP9414 23T2

1 Theoretical Background

Reinforcement learning (RL) [1] is a machine learning method based on behavioural psychology. It allows an agent to learn how to perform new tasks by exploring the environment and observing state modifications and possible rewards. The method is oriented toward goals in which an agent, either human or robotic, tries to maximise the long-term cumulative reward by iterative interactions with the environment. Figure 1 shows the classic interaction loop between an RL agent and the environment.

An RL problem is comprised of:

- A policy: that defines how an agent selects an action aiming to maximize the reward signal obtained.
- A reward signal: that establishes a definition of positive and/or negative events, i.e., the aims to be achieved by the RL agent.
- A value function: that specifies how good the reward signal might be over time from one state or a state-action pair.
- Optionally, a model of the environment: that allows the agent to infer what will be the next state and reward, given any state and action.

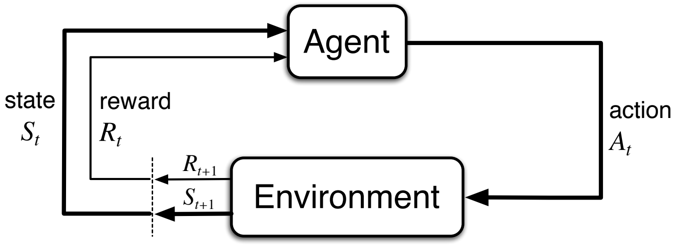

Figure 1: Reinforcement learning loop between the agent and the environment [1]. At each time-step the RL agent from s_t selects an action a_t to perform and receives from the environment the state s_{t+1} and a reward signal r_{t+1}.

Based on this model, an RL agent is able to learn the optimal policy that allows selecting the action leading to the highest cumulative reward given the value function.

Markov decision processes (MDPs) are the base of RL tasks. In an MDP, transitions and rewards depend only on the current state and the selected action by the agent [2]. In other words, a Markov state contains all the information related to the dynamics of a task, i.e., once the current state is known, the history of transitions that led the agent to that position is irrelevant in terms of the decision-making problem.

An MDP is characterized by the 4-tuple $< S, A, \delta, r >$ where:

- S is a finite set of states,
- A is a set of actions,
- δ is the transition function $\delta : S \times A \to S$, and,
- r is the reward function $r : S \times A \to \mathbb{R}$.

At each time t, the agent perceives the current state $s_t \in S$ and selects the action $a_t \in A$ to perform it. The environment returns the reward $r_t = r(s_t, a_t)$ and the agent transits to the state $s_{t+1} = \delta(s_t, a_t)$. The functions r and δ depend only on the current state and action, i.e., it is a process with no memory.

Actions are selected according to a policy π, which in psychology is called a set of stimulus-response rules or associations [3]. Thus, the value of taking an action a in a state s under a policy π is denoted $q^\pi(s, a)$ which is also called the action-value function for a policy π.

To find a policy, diverse learning methods exist, e.g., SARSA and Q-learning. The on-policy method SARSA [4] considers transitions from state-action pair to state-action pair as shown in Eq. (1). In the Q-learning method, state-action values are updated according to the Eq. (2) [5, 6].

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{1}$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a \in A(s_{t+1})} Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{2}$$

2 Setup

(a) Using Python, create a gridworld of variable dimension $n \times m$ allowing positive and negative rewards for some positions.

(b) Create methods for returning the current state, the reward, the number of states, and the number of actions.

(c) Create a method to perform an action into the gridworld, i.e., a method that moves the agent from one state to another given one of the following actions: right, left, up, down.

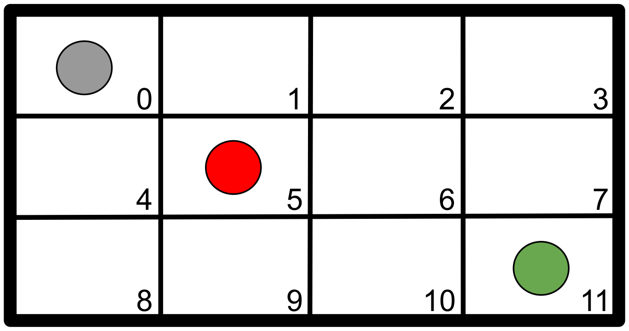

Figure 2: 3 × 4 gridworld with one goal state and one fear state..

(d) Consider only valid movements, i.e., actions attempting to move the agent out of the gridworld have no effect.

(e) Create a reinforcement learning agent with the following characteristics:

- Use a Q-table with Q-values randomly initialized. Use a uniform distribution between 0 and 0.01.
- Learning parameters: learning rate $\alpha = 0.7$, discount factor $\gamma = 0.4$, and ϵ-greedy action selection method with $\epsilon = 0.25$.
- Implement the temporal-different on-policy method SARSA for training.

3 Experiments

(a) Create a 3×4 gridworld. See Fig. 2.

(b) Use the grid $(2, 3)$ as the goal position with reward 1.0 and the grid $(1, 1)$ as a fear region with reward -1.0.

(c) Create a learner agent to navigate the gridworld previously created.

(d) Train the reinforcement learning agent using SARSA for 1000 episodes.

(e) Plot the Q-values. Observe the state-action pair moving the agent to both the goal and fear positions.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[2] M. L. Puterman, *Markov decision processes: Discrete stochastic dynamic programming.* John Wiley & Sons, 2014.

[3] S. Kornblum, T. Hasbroucq, and A. Osman, “Dimensional overlap: cognitive basis for stimulus-response compatibility–a model and taxonomy.,” *Psychological review*, vol. 97, no. 2, p. 253, 1990.

[4] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*, vol. 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[5] C. J. Watkins, *Learning from Delayed Rewards.* Doctoral dissertation, University of Cambridge, 1989.

[6] P. Dayan and C. Watkins, “Q-learning,” *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.