Reinforcement Learning – exercise 05

Hung Ngo & Vien Ngo
Machine Learning & Robotics lab, University of Stuttgart
Universitsstrae 38, 70569 Stuttgart, Germany

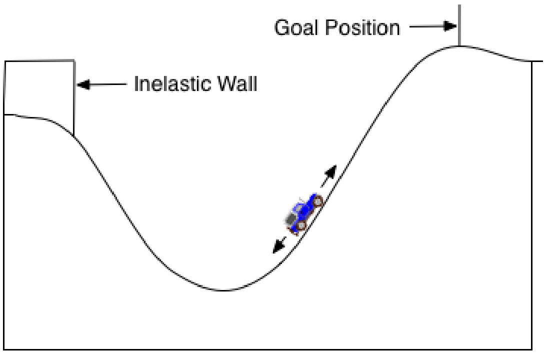
May 11, 2016

Mountain Car Problem with Tabular Q(λ)

The mountain car problem, depicted in the figure, is as follows: starting from the bottom of a valley, an underpowered car has to gain enough momentum to reach the top of a mountain. The objective is to minimize the number of time steps to reach the goal. There are three possible values of action a: full throttle (i.e., for acceleration) forward (+1), full throttle reverse (-1), and zero throttle (0). The continuous state space is defined by $\{p_t, \dot{p}_t\}$, where the state variables are *bounded* $p_t \in [-1.2; 0.5]$ and $\dot{p}_t \in [-0.07; 0.07]$ are respectively the position and velocity of the car. At the beginning of each episode, the car starts at the default initial state $\{p_0 = -0.52, \dot{p}_0 = 0.0\}$, i.e., in the bottom of the valley (verify for yourself!). The cost in this problem is -1.0 for all time steps until the car moves past its goal position (i.e., $p_t \geq 0.5$) at the top of the mountain, getting 0 reward, which ends the episode. The dynamics of the car can be described as

$$\dot{p}_{t+1} = \dot{p}_t + 0.001a - 0.0025\cos(3p_t); \quad p_{t+1} = p_t + \dot{p}_{t+1}$$

When p_{t+1} reaches the left bound, i.e., $p_{t+1} \leq -1.2$, the velocity is reset to zero.



We *discretize* the state space as follows: 20 intervals for p and 20 for $\dot{p}$ in order to obtain 400 discrete states. Note that you need to maintain the true values of state variables (position and velocity) following the dynamic equations, and implement a function mapping such continuous values to a table index (discrete state). Initialize the action-value table to all zeros. For eligibility traces: $\lambda = \{0, 0.3, 0.6, 0.9, 1\}$. Other parameters: $\gamma = 0.99, \alpha = 0.1, \epsilon = 0.0$. Remember to use random tie breaking for $\max$ operator.

For each value of λ,
– let the agent learn for 100 episodes using tabular Q(λ). Set maximum number of steps in each episode to e.g. 10^5 (episodic task). Record the number of steps for each episode.
– repeat the whole process 10 times, then plot i) the *averaged* cumulative number of successes (reaching goal state), and ii) the *averaged* number of steps per episode (y-axis) against number of episodes (x-axis).

Analyze the results:
– the learning curve
– what drives exploration? (as we set $\epsilon = 0$)
– the effect of eligibility parameter λ
– computation time when you increase #intervals to 200, 2000 (i.e., 10, 100 times)?