

Reinforcement Learning – exercise 06

Hung Ngo & Vien Ngo

Machine Learning & Robotics lab, University of Stuttgart
Universitätsstrae 38, 70569 Stuttgart, Germany

May 24, 2016

Mountain Car Problem with Linear SARSA(λ) and RBFs

The mountain car problem, described in HW5, 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 use $n = n_p \times n_v$ fixed RBFs for feature extraction (i.e., the centroids are on a grid in the state space). Each RBF has the form $\phi_i(\mathbf{x}) = \exp(-\frac{d_i^2}{2})$ with $d_i^2(\mathbf{x}) = (\mathbf{x} - \mathbf{c}_i)^\top \Sigma_i^{-1} (\mathbf{x} - \mathbf{c}_i)$, where $\mathbf{x} = (p_t, \dot{p}_t)^\top$ the state variables, $\mathbf{c}_i = (p_i, \dot{p}_i)^\top$ the i -th centroid's coordinates, and $\Sigma_i = \text{diag}(\sigma_p^2, \sigma_v^2)$ the same diagonal matrix for all i . You can use $n_p = 4; n_v = 8; \sigma_p^2 = 0.04; \sigma_v^2 = 0.0004$.

For each action, use one distinct weight vector (controller) θ^a . Hence, action value function is approximated as $Q(\mathbf{x}, a) = \sum_{i=1}^n \theta_i^a \phi_i(\mathbf{x})$. Initialize all weight vectors to zeros. For eligibility traces: $\lambda = \{0, 0.3, 0.6, 0.9, 1\}$. Other parameters: $\gamma = 0.99, \alpha = 0.01, \epsilon = 0.0$. Remember to use random tie breaking for max operator.

For each value of λ ,

- let the agent learn for 100 episodes using linear SARSA(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 on the learning curve and the effect of eligibility parameter λ .