Practical 3: Approximate Q-Learning

Guillaume Charpiat Corentin Tallec

January 22, 2017

Preamble
This exercise session is adapted from a project of the EPFL *Unsupervised and Reinforcement Learning in neural networks* course.

Exercise
In this project, you are asked to solve the mountain car problem, using the approximate Q-Learning algorithm and its TD(λ) generalization. The mountain car problem is that of a car stuck amidst two mountains. A left or right constant thrust can be applied to the car. The goal is to make the car climb the mountain to its right. The thrust that can be applied is not sufficient for the car to directly go uphill. Momentum has to be gathered to climb high enough.

You are provided with an implementation of the mountain car problem. The state space of the agent consists of the x coordinate of the car, and its horizontal velocity v_x. x lies in $[-150, 0]$ and v_x lies in $[-20, 20]$. The class of approximating functions you are strongly advised to use proceeds as follow. It maps points (x, v_x) into a high dimensional representation and consider the class of linear functions in this high dimentional space. When the high dimentional space considered is well chosen, this considerably increases expressivity when compared to directly considering linear functions on the state space. Consider $s^{0,0}, \dots, s^{p,k}$ nodes of a discretization of the state space. Typically, dividing both axes evenly into respectively p and k intervals gives $(p+1) \times (k+1)$ points $s^{i,j}$ with coordinates

$$s^{i,j} = (-150 + i\frac{150}{p}, -20 + j\frac{40}{k}). \tag{1}$$

Each coordinate of the $(p+1) \times (k+1)$ dimensional representation vector ϕ only depends on the closeness of (x, v_x) and $s^{i,j}$. Typically, $\phi_{i,j}$ is 1 if

$(x, v_x) = s^{i,j}$ and tends to zero the farther (x, v_x) is to $s^{i,j}$. Formally

$$\phi_{i,j} = e^{-(x - s_1^{i,j})^2} e^{-(v_x - s_2^{i,j})^2}. \tag{2}$$

The Q-function approximation then takes the form

$$Q((x, v_x), a) = \sum_{i,j} W_{i,j}^a \phi_{i,j} \tag{3}$$

where the $W_{i,j}^a$ are real numbers. You are asked to design a Q-Learning and a TD(λ) algorithms that learn parameters $W_{i,j}^a$ to solve the mountain car task. p, k are to be tunable hyperparameters of your algorithm, along with any other hyperparameter of interest. The provided code gives you access to facilities to visualize your learning progresses.