Reinforcement Learning

The world

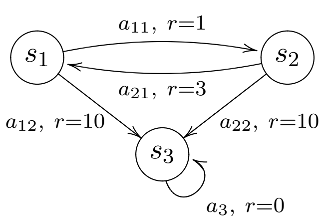
The world is defined in terms of a finite number of **states**. In each state there is a finite number of **actions**. Taking the action a while in the state s results in a reward $r(s, a)$ and a transition to another state $s' = \delta(s, a)$.

Example: Here we have three states s_1, s_2, s_3. The functions r and δ are:

$$\begin{aligned} r(s_1, a_{11}) = 1, &\quad r(s_1, a_{12}) = 10 \\ r(s_2, a_{21}) = 3, &\quad r(s_2, a_{22}) = 10 \\ r(s_3, a_3) = 0 & \end{aligned}$$

$$\begin{aligned} \delta(s_1, a_{11}) = s_2, &\quad \delta(s_1, a_{12}) = s_3 \\ \delta(s_2, a_{21}) = s_1, &\quad \delta(s_2, a_{22}) = s_3 \\ \delta(s_3, a_3) = s_3 & \end{aligned}$$

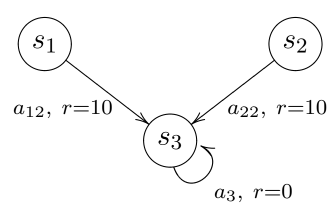
The following diagram illustrates this world:



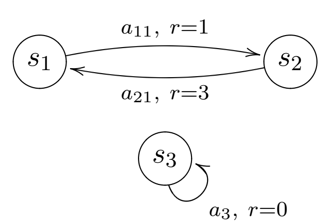
Policies

The goal of learning is to come up with a policy to determine what action is to be taken at each state. A policy is denote by π. Here are several examples.

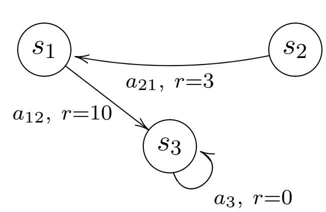
$\pi_1(s_1) = a_{12}, \quad \pi_1(s_2) = a_{22}, \quad \pi_1(s_3) = a_3,$



$\pi_2(s_1) = a_{11}, \quad \pi_2(s_2) = a_{21}, \quad \pi_2(s_3) = a_3,$



$\pi_3(s_1) = a_{12}, \quad \pi_3(s_2) = a_{21}, \quad \pi_3(s_3) = a_3,$



Evaluating policies

Given a policy and a starting state s_t, we can compute all future states, actions, and rewards as follows:

$$\begin{array}{lll} a_t = \pi(s_t), & r_t = r(s_t, a_t), & s_{t+1} = \delta(s_t, a_t), \\ a_{t+1} = \pi(s_{t+1}), & r_{t+1} = r(s_{t+1}, a_{t+1}), & s_{t+2} = \delta(s_{t+1}, a_{t+1}), \\ \ldots & \ldots & \ldots \end{array}$$

To evaluate a policy for a starting point s_t we add the current reward and discount future rewards by a factor of γ which can take values in the range $0 \leq \gamma < 1$:

$$V^\pi(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = \sum_{i=0}^{\infty} \gamma^i r_{t+i} = r_t + \gamma V^\pi(s_{t+1})$$

Observe that $V^\pi(s_t)$ is always finite since:

$$V^\pi(s_t) \leq r_{\max}(1 + \gamma + \gamma^2 + \ldots) = \frac{r_{\max}}{1 - \gamma}$$

Our goal is to find a policy that maximizes $V^\pi(s)$ for all s. For a given s we can compute a locally optimal policy by:

$$\pi(s) = \arg\max_\pi V^\pi(s)$$

Now *assume* that there is a single optimal policy π^* such that:

$$\text{for all } s\text{:} \quad \pi^*(s) = \arg\max_\pi V^\pi(s)$$

Q Learning

To simplify notation we write: $V^*(s) = V^{\pi^*}(s)$. An optimal policy π^* must satisfy:

$$\pi^*(s) = \arg\max_a V^*(s) = \arg\max_a (r(s,a) + \gamma V^*(\delta(s,a))) = \arg\max_a Q(s,a)$$

where:

$$Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a))$$

This gives the following dependency relation between π^*, V^*, Q:

$$\begin{tikzcd}
\pi^* \arrow[r, "r{,}\delta"'] & V^* \arrow[r, "r{,}\delta"'] & Q \arrow[ll, bend left, "r{,}\delta"]
\end{tikzcd}$$

This suggests the following algorithm:

Input: $\hat{Q}$, an initial approximation to Q.

Output: π^*

Iterate steps 1,2,3

1. Compute $\hat{\pi}$ from $\hat{Q}$ using:

$$\hat{\pi}(s) = \arg\max_a \hat{Q}(s,a)$$

2. Compute $\hat{V}$ from $\hat{\pi}$ using:

$$\hat{V}(s_t) = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

3. Compute $\hat{Q}$ from $\hat{V}$ using:

$$\hat{Q}(s,a) = r(s,a) + \gamma \hat{V}(\delta(s,a))$$

The algorithm can also be started with an initial approximation to $\hat{\pi}$ or $\hat{V}$.

Online version

In the online version we maintain $\hat{Q}$ and update it based on current "experience" in the form: s, a, r, s'. It is based on the observation that $V^*(s) = \max_{a'} Q(s,a')$ which gives the following recursive relation that Q must satisfy:

$$Q(s,a) = r(s,a) + \gamma \max_{a'} Q(\delta(s,a), a')$$

Input: $\hat{Q}$, an initial approximation to Q and a recent experience in the form of s, a, r, s'.

Output: an improvement of $\hat{Q}$.

Update:

$$\hat{Q}(s,a) = r + \gamma \max_{a'} \hat{Q}(s', a')$$