# Introduction to Reinforcement Learning II

ML on quantum and classical data

Gorka Muñoz-Gil (**ICFO**)

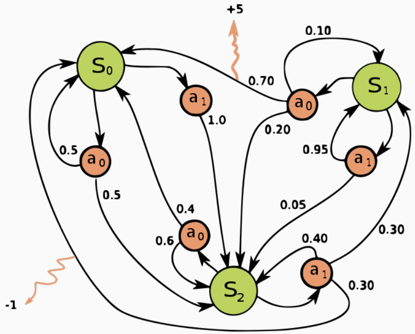
Master in Photonics (UPC-UAB-UB-ICFO), 2018-2019

## Markov decision process

MAB: the action was done over the same *invariant* system. What happens when our actions affect the system?

$\rightarrow$ We will need to learn how to choose different actions in different situations! Namely: $q_*(a) \rightarrow q_*(s, a)$

Before that, what is a **MDP**?



## Markov decision process

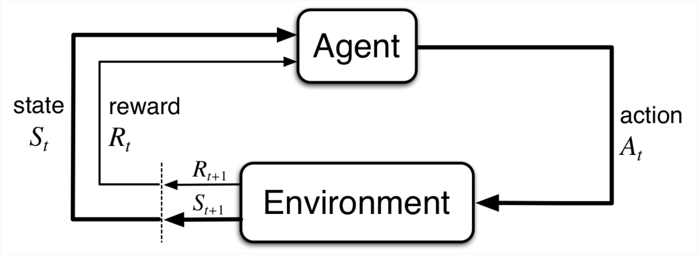
**Elements of MPD**

Formally, an MDP is an Markovian process ('no memory') define by a 5 elements:

- Finite set of states $S$.
- Finite set of actions $A$.
- Set of probabilities $P_a(s, s')$
- Set of rewards $R_a(s, s')$
- $\gamma$ discount factor, which represents the difference in importance between future and past.

## Markov decision process

In terms of RL, we treat with systems like



To keep track of learning, we will use the tuple $(S_t, A_t, S_{t+1}, R_{t+1})$.

We will consider *finite* MDP: finite set of states, actions and rewards.

## Markov decision process

The MDP is completely characterized by the probability

$$p(s', r|s, a) = \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$$

See that $\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1$. From here one can set interesting probabilistic relations such as the *expected reward for state-action pairs*,

$$r(s, a) = \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a] = \sum_{s' \in \mathcal{S}} r \sum_{r \in \mathcal{R}} p(s', r|s, a)$$

In general, MDPs allow for a simple and elegant representation of our real world problem. The only requirement is to draw the correct agent-environment system.

## Markov decision process

**Goals and Rewards**

In MDPs, the goal is to maximize the expected value of the cumulative sum of received rewards.

Rewards can be either positive, negative (usually referred as penalization) or zero.

**Example**

Positive reward: when making a robot learn to walk, reward is proportional to robot's forward motion.

Penalization: when making a robot learn to escape a maze as faster as possible, each step in maze give reward -1.

## Markov decision process

**Returns and Episodes**

As in the MAB, we want to maximize the expected return $G_t = R_{t+1} + R_{t+2} + ... + R_T$. H

We consider the agent-environment interact broken into *episodes* of lenght $T$. Normally they all start in same initial state.

Additionally, we may want to choose $A_t$ to maximize the expected *discounted return*:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = R_{t+1} + \gamma G_{t+1} = \sum_{k=0}^{T-1} \gamma^k R_{t+k+1}$$

## Markov decision process

**Policies and value functions**

Policy $\pi(a|s)$: mapping from states to probabilities of selecting each possible action. Is the 'brain' of the agent.

Value function $v_\pi(s)$: is the expected return when starting in state $s$ and follow policy $\pi$,

$$v_\pi(s) = \mathbb{E}[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{T-1} \gamma^k R_{t+k+1} \middle| S_t = s\right]$$

The value of taking action $a$ in state $s$, the action-value function, under policy $\pi$ is defined as

$$q_\pi(s,a) = \mathbb{E}[G_t|S_t = s, A_t = a]$$

## Markov decision process

### Optimal policies and value functions

Policy $\pi$ is better than policy $\pi'$ (also written as $\pi > \pi'$) if and only if $v_\pi(s) > v_{\pi'} \; \forall \; s$. We denote the optimal policy as $\pi_*$. We then have

$$\boxed{q_* = \max_\pi q_\pi(s, a)} \quad \boxed{v_* = \max_\pi v_\pi(s)}$$

Both functions follow the *Bellman optimality equation*:

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]$$

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')]$$

Once here, we have now all the tools needed to create our own policy!