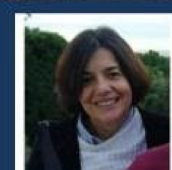
REINFORCEMENT LEARNING

Instructors

Josep Vidal



Margarita Cabrera

Xavier Giró-i-Nieto

Organizers

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

telecom BCN

Supporters

Google Cloud

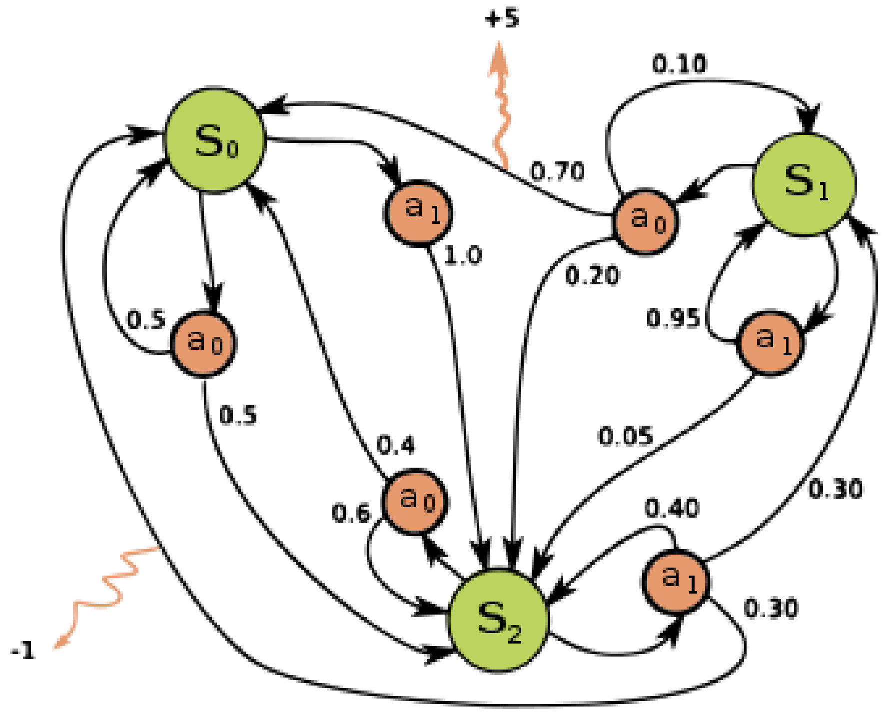
GitHub Education

+ info: http://bit.ly/upcrl-2020

telecos BCN

UPC

2. Markov Decision Processes



Markov Decision Process

- MDP formally describe an environment for RL
- We start by defining:
    - Markov Process
    - Markov Reward Process
    - Markov Decision Process

Markov Process

- A Markov process is a stochastic model describing a sequence of observed states in which the probability of each state depends only on the previous state.

- It is defined by the tuple $\langle \mathcal{S}, \mathcal{P} \rangle$

- In a Markov process, given the present, the future is independent of the past

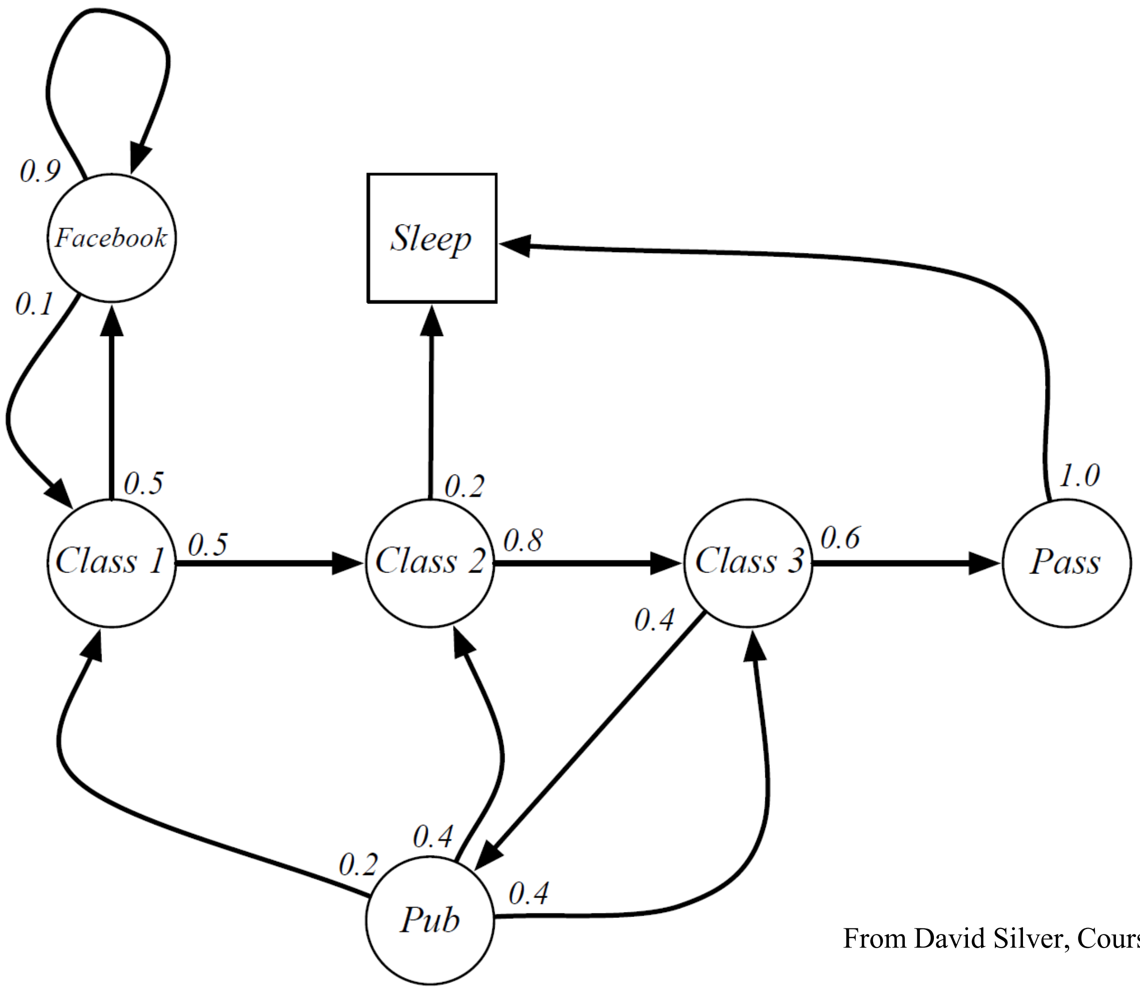
$$\Pr\left\{S_{t+1} \middle| S_1, \ldots, S_t\right\} = \Pr\left\{S_{t+1} \middle| S_t\right\}$$

- The state transition probabilities are defined as

$$p(s' \mid s) = \Pr\left\{S_{t+1} = s' \middle| S_t = s\right\} \qquad \mathcal{S} = \left\{s_1, \ldots, s_n\right\}$$

$$\sum_{s' \in S} p(s' \mid s) = 1$$

Example 2.1. Student Markov Chain



From David Silver, Course on RL, 2015

6 states plus one terminal state

Markov Reward Process

- A Markov reward process is a MP with return values.

- It is defined by the tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

Rewards may be random

- Rewards are defined as $R_s = E\left\{ R_{t+1} \middle| S_t = s \right\}$

- The return G_t is the total discounted reward from step *t*, it is a random variable

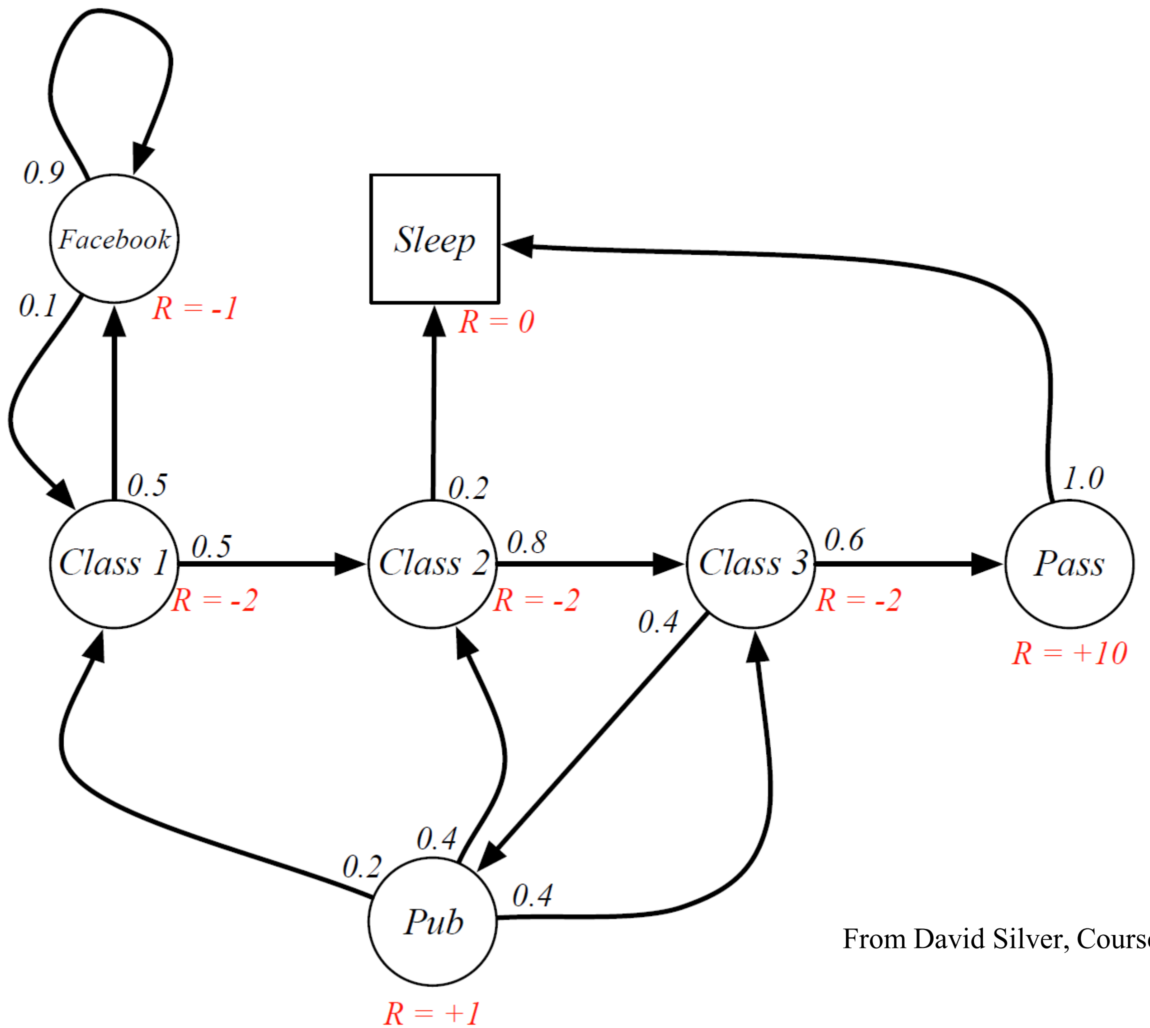
$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$$

γ avoids optimising over an infinite horizon:

$\gamma \simeq 0$ leads to "myopic" evaluation

$\gamma \simeq 1$ leads to "far-sighted" evaluation

Example 2.1. Student Markov Reward Process



From David Silver, Course on RL, 2015

The agent has no decision capabilities yet, but it receives a reward (in red) each time it visits a state.

Markov Reward Process

Why discounting?

- It is mathematically convenient.
- Uncertainty about the future may not be fully represented.
- Animal/human behaviour shows preference for immediate rewards rather than delayed rewards.
- It is possible to use undiscounted reward ($\gamma = 1$) if all sequences terminate.

Take away message: Rewards can be scaled and the MRP is unchanged. Mean substraction changes the MRP.

Bellman equation for MRP

- The state value function $v(s)$ of an MRP is the expected return starting from state s

$$v(s) = E\{G_t | S_t = s\}$$

- It can be decomposed into two parts:
  - immediate reward
  - discounted value of successor rate

$$\begin{aligned}
v(s) = \boxed{E}\{G_t | S_t = s\} &= E\{R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots | S_t = s\} \\
&= E\{R_t + \gamma(R_{t+1} + \gamma R_{t+2} + \ldots) | S_t = s\} \\
&= R_s + \gamma E\{G_{t+1} | S_t = s\} = R_s + \gamma \sum_{s' \in \mathcal{S}} p(s' | s) E\{G_{t+1} | S_{t+1} = s'\} \\
&= R_s + \gamma \sum_{s' \in \mathcal{S}} p(s' | s) v(s')
\end{aligned}$$

Expectation due to randomness of the environment

Bellman equation for MRP

- The state value function $v(s)$ can be computed from the matrix Bellman equation:

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} + \gamma \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

$$\mathbf{v} = \mathbf{R} + \gamma \mathbf{P} \mathbf{v} \longrightarrow \mathbf{v} = \left(\mathbf{I} - \gamma \mathbf{P} \right)^{-1} \mathbf{R}$$

- Matrix inversión entails $O(n^3)$ complexity for n states.
- Direct solution only possible for small MRP.

Markov Decision Process

- A Markov decision process is a Markov reward process with decisions. Under the Markov assumption, any action affects (1) the immediate reward and (2) the next state.

- It is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$
- $\mathcal{A}$ is a finite set of possible actions.
- Model components predict the reaction of the environment:

  - $\mathcal{P}$ is a state transition probability matrix characterizing the environment, and whose elements are:

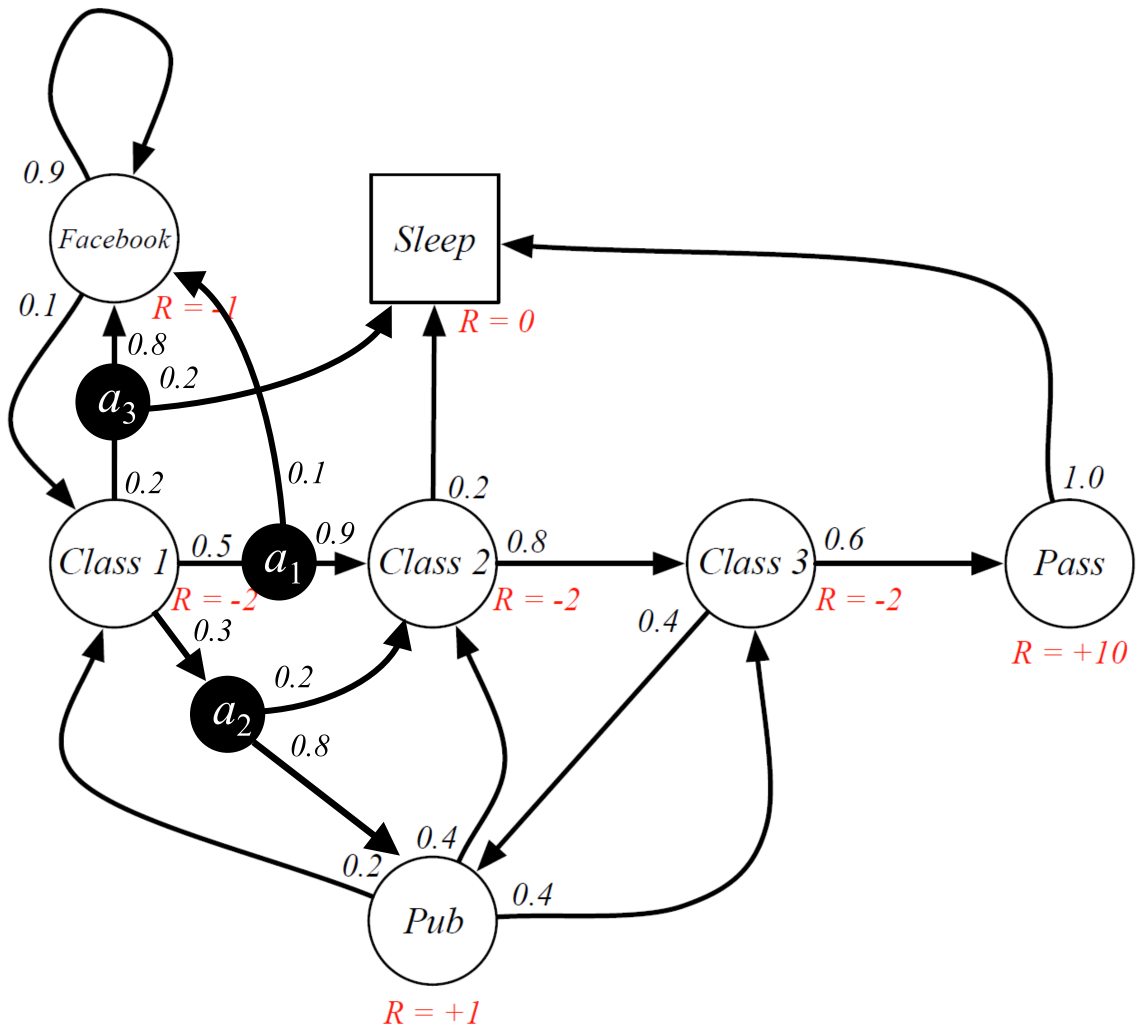
Next state may be deterministic or random

$$p(s' \mid s, a) = \Pr\left\{ S_{t+1} = s' \middle| S_t = s, A_t = a \right\}$$

  - $\mathcal{R}$ is a reward function taking values

$$R_s^a = E\left\{ R_{t+1} \middle| S_t = s, A_t = a \right\}$$

Example 2.1. Student Markov Reward Process



The agent has now some decision capabilities in state "*Class 1*" (represented by actions in black circles) and it receives a reward each time it visits a state. Actions from other states have not been represented for simplicity.

Markov Decision Process

A policy π is a distribution of possible actions given states and fully defines the behavior of an agent.

Policies depend on the current state (not on the history) and can be…

- ✓ Deterministic $a = \pi(s)$ `if s then a`
- ✓ Random $\pi(a|s) = \Pr\{A_t = a | S_t = s\}$

Markov Decision Process

Linking concepts…

Given an MDP and a policy π:

- The state sequence $S_1 S_2 \ldots$ is a Markov process
- The state and reward sequence $S_1 R_1 S_2 R_2 \ldots$ is an MRP, where

$$P_{ss'}^{\pi} = \sum_{a \in \mathcal{A}} \pi\left(a \middle| s\right) p(s' \mid s, a) \qquad R_s^{\pi} = \sum_{a \in \mathcal{A}} \pi\left(a \middle| s\right) R_s^a$$

We'll get rid of randomness on actions and/or environment by taking expectations…

Markov Decision Process

- Action-value function is the expected return starting from state *s*, taking action *a*, and then following policy π

$$q_\pi(s,a) = E_\pi\{G_t \mid S_t = s, A_t = a\} = R_s^a + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s,a) v_\pi(s')$$

- State-value function is the expected return starting from state *s*, and following policy π

$$v_\pi(s) = E_\pi\{G_t \mid S_t = s\}$$

Averaged over all actions, states and rewards starting from t+1 until T (the end of the episodes)

Markov Decision Process

- State-value function is the expected return starting from state s and following policy π, and can be decomposed into immediate reward plus discounted value of successor state:

$$
\begin{aligned}
v_\pi(s) &= E_\pi\left\{R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots \middle| S_t = s\right\} \\
&= E_\pi\left\{R_t + \gamma G_{t+1} \middle| S_t = s\right\} \\
&= \sum_a \pi(a|s) \sum_{r,s'} p(r,s'|s,a)\left[r + \gamma E_\pi\left\{G_{t+1} \middle| S_{t+1} = s'\right\}\right] \\
&= \sum_a \pi(a|s) \sum_{r,s'} p(r,s'|s,a)\left[r + \gamma v_\pi(s')\right]
\end{aligned}
$$

Stochastic environment

Stochastic policy

All future random variables are included in this $E_\pi\{.\}$: given S_{t+1}, the return G_{t+1} does not depend on the past

Proof. Rewards beyond t+1 are associated to future states, not to past, therefore average is done with respect to random variables in the future:

Includes all other future variables, not shown for simplicity

$$E_\pi\left\{G_{t+1}\middle|S_t=s\right\}=\sum_{g,s',s'',\mathbf{a}} g\cdot p\left(g,s',s'',\mathbf{a}\middle|s\right)$$

$$=\sum_{g,s',s'',\mathbf{a}} g\cdot p\left(g,s'',\mathbf{a}\middle|s',s\right)p\left(s'\middle|s\right)$$

Markov assumption: given s', s'' does not depend on s

$$=\sum_{g,s',s'',\mathbf{a}} g\cdot p\left(g,s'',\mathbf{a}\middle|s'\right)p\left(s'\middle|s\right)$$

$$=\sum_{s'} p\left(s'\middle|s\right)\sum_{g,s'',\mathbf{a}} g\cdot p\left(g,s'',\mathbf{a}\middle|s'\right)$$

$$=\sum_{s'} p\left(s'\middle|s\right)E_\pi\left\{G_{t+1}\middle|S_{t+1}=s'\right\}=\sum_{s'} p\left(s'\middle|s\right)v_\pi(s')$$

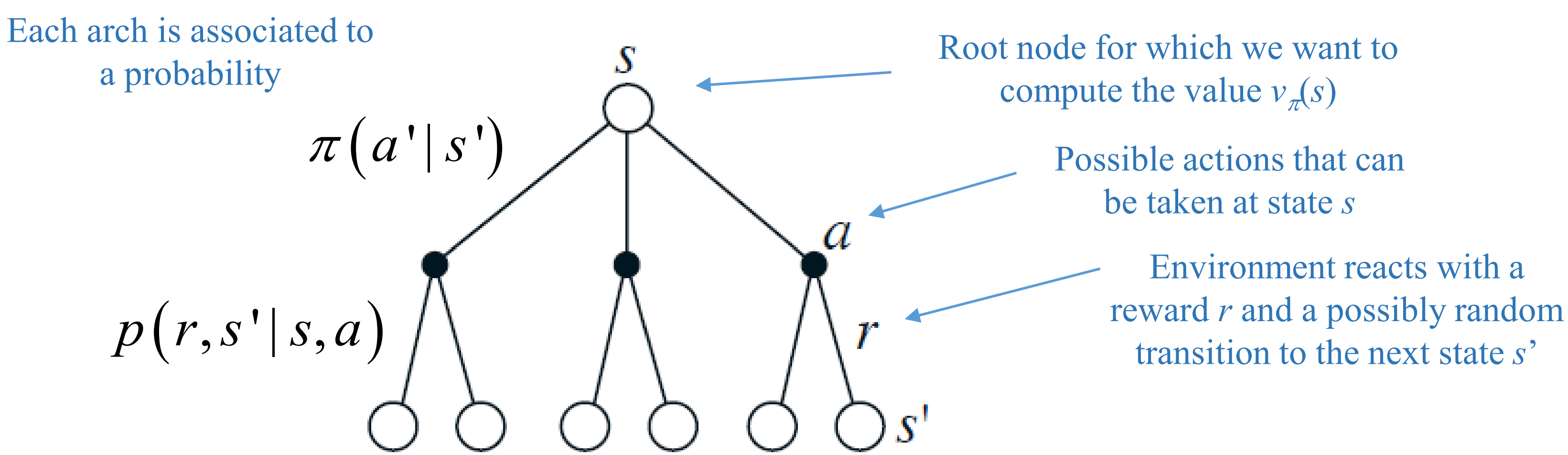
Bellman expectation equations for MDP

This recursive definition is very relevant in RL…

$$v_\pi(s) = E_\pi\left\{R_t + \gamma v_\pi(S_{t+1}) \middle| S_t = s\right\}$$

$$\mathbf{v}_\pi = \mathbf{R}^\pi + \gamma \mathbf{P}^\pi \mathbf{v}_\pi$$

Can be understood as propagating values over a backup diagram:



Information sums up from bottom to top.

Markov Decision Process

- Action-value function is the value of following policy π after committing action a in state s. It can be decomposed as:

$$
\begin{aligned}
q_\pi(s,a) &= E_\pi\left\{G_t \middle| S_t = s, A_t = a\right\} \\
&= E_\pi\left\{R_{t+1} + \gamma G_{t+1} \middle| S_t = s, A_t = a\right\} \\
&= \sum_{r,s'} p\left(r,s' \middle| s,a\right)\left[r + \gamma E_\pi\left\{G_{t+1} \middle| S_{t+1} = s'\right\}\right] \\
&= \sum_{r,s'} p\left(r,s' \middle| s,a\right)\left[r + \gamma v_\pi\left(s'\right)\right]
\end{aligned}
$$

Note we are not averaging over π

It is possible to write $v(s)$ in terms of $q(s, a)$? Yes…

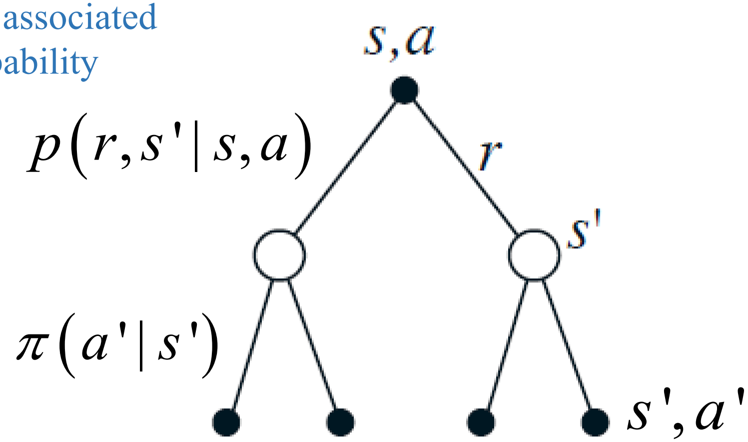
$$
v_\pi(s) = \sum_a \pi(a \mid s) \sum_{r,s'} p\left(r,s' \middle| s,a\right)\left[r + \gamma v_\pi(s')\right] = \sum_a \pi(a \mid s)\, q_\pi(s,a)
$$

Bellman expectation equation for $q(s,a)$

Putting both equations together…

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

Each arch is associated to a probability



Information sums up from bottom to top.

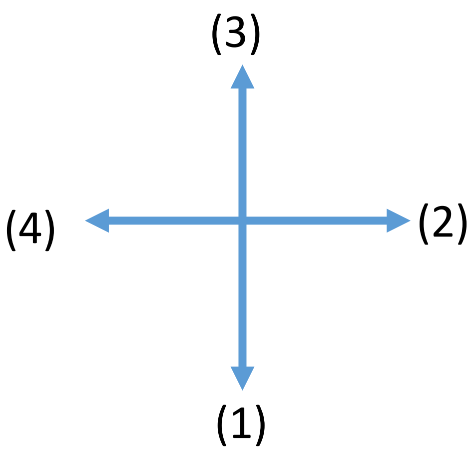
Direct solution only possible for small MDP.

For large MDP, iterative methods exist: Dynamic programming, Monte-Carlo evaluation, Temporal-difference learning (next lectures).

Example 2.2. 5×5 Gridworld example (I)

- $N = 25$ states; $\mathcal{S} = \{1,2,..,25\}$
- Actions $\mathcal{A}=\{1,2,3,4\}$ south (1), east (2), north (3) and west (4)
- Actions that take the agent off the grid leave its location unchanged, $r = -1$
- Other actions $r = 0$
- Special state 6(16): all actions $r = +10(5)$ and takes the agent to state 10(18)

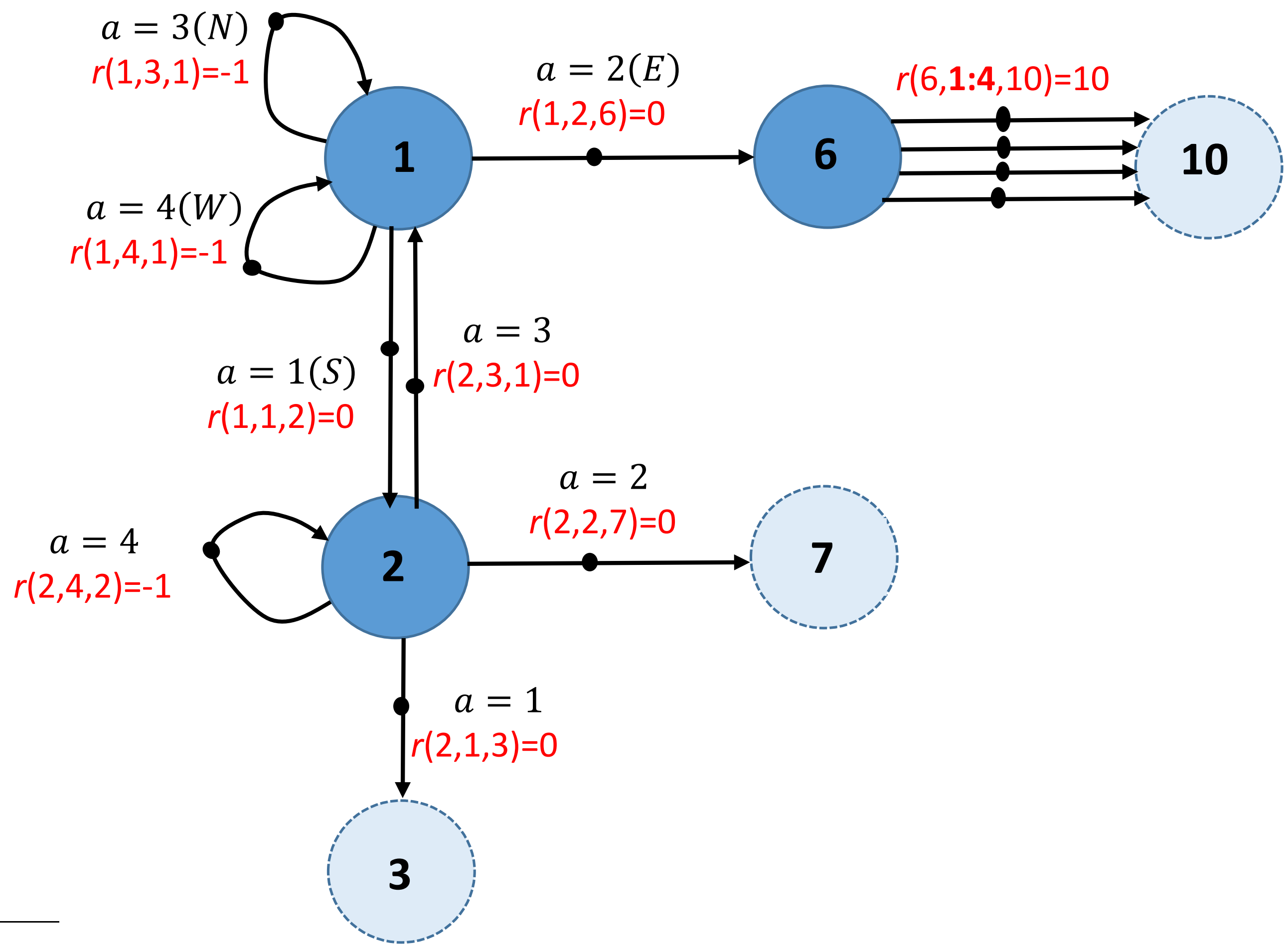
1	6	11	16	21
2	7	12	17	22
3	8	13	18	23
4	9	14	19	24
5	10	15	20	25



From Sutton & Barto, Reinforcement Learning: An Introduction, 1998

Example 2.2. 5×5 Gridworld example (II)

- Transition graph for states 1, 2 and 6; $r(s,a,s')$ is the reward of state s when action a is run and the immediate next state is s'.



Example 2.2. 5×5 Gridworld example (III)

Code in Matlab or Python this algorithm…

a) Initiate the variable $p(s'|a,s)$ and the corresponding reward $r(s,a,s')$ by giving values for s, s'=1,...,25 and a =1,...,4.
   Hint: Directly generate matrix elements P($|\mathcal{S}|$ rows, $|\mathcal{A}|$ columns) as $P(s,a) = s'$;
   Hint: Directly generate matrix elements R($|\mathcal{S}|$ rows, $|\mathcal{A}|$ columns) as $R(s,a) = r(s,a,s')$;

b) If π is the equiprobable random policy, obtain the reward vector R^{π} and the probability matrix P^{π}. Draw in a square image P^{π}

c) Solve the Bellman equation with a discount factor $\gamma = 0.9$ and draw in a square figure the value function of each state.

Optimal Value Functions

Our ultimate goal is to determine which is the best policy.

- The **optimal state-value function** is the maximum value-function over all policies

$$v_*(s) = \max_\pi v_\pi(s)$$

- The **optimal action-value function** is the maximum action-value function over all policies

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

as the expected return obtained when taking action *a* in state *s*, and then, follow the optimal policy for the rest of the episode.

How to compare policies? $\pi \geq \pi' \quad \text{if} \quad v_\pi(s) \geq v_{\pi'}(s), \quad \forall s$

Optimal Value Functions

The optimal value function specifies the best possible performance in the MDP. An MDP is “solved” when we know the optimal value function.

Fundamental theorem. For any MDP…

There exists at least one **optimal policy** π^* that is better than or equal to all other policies.

- All optimal policies achieve the optimal value function

$$v_{\pi^*}(s) = v_*(s)$$

- All optimal policies achieve the optimal action-value function

$$q_{\pi^*}(s,a) = q_*(s,a)$$

Optimal Value Functions

- In addition,

$$\pi^*(a \mid s) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in \mathcal{A}} q_*(s,a) \\ 0 & \text{otherwise} \end{cases}$$

that is, in any MDP there is always a deterministic optimal policy.

- If we know $q_*(s,a)$ we immediately have the optimal policy.

- An optimum policy is deterministic[1]: it points out the best action *a* given a state *s*.

Let's assume π_d is a deterministic policy:

$$\pi_d(a \mid s) = \begin{cases} 1 & a = a_s^d \\ 0 & \text{otherwise} \end{cases}$$

In vector form

alternatively: $\pi_d(s) = a_s^d$ or $\boldsymbol{\pi}_d = [a_1^d, \ldots, a_n^d]^T$

then: $v_{\pi_d}(s) = \sum_a \pi_d(a \mid s) q_{\pi_d}(s,a) = q_{\pi_d}(s, a_s^d)$

and: $q_{\pi_d}(s,a) = \begin{cases} q_{\pi_d}(s, a_s^d) \ \text{ for } \ a = a_s^d \\ \text{no interest for } a \neq a_s^d \end{cases}$

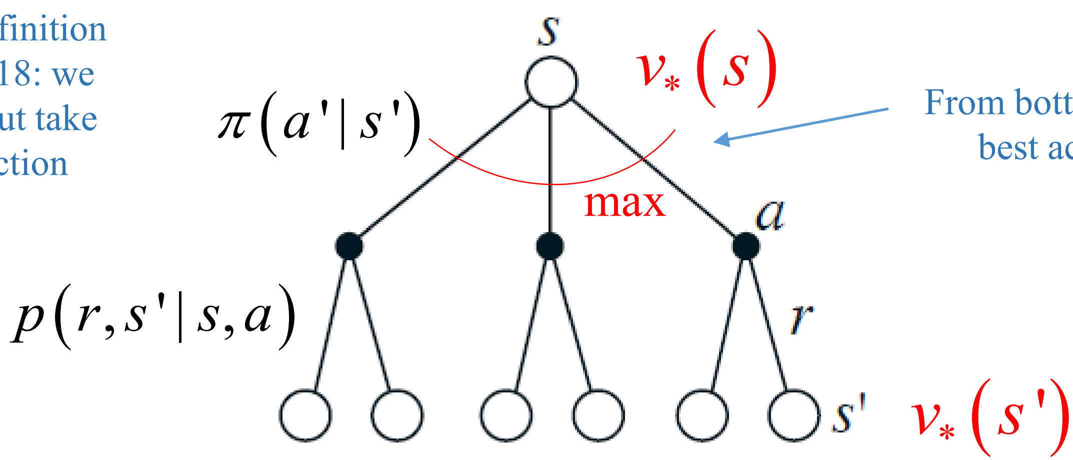
(1) In the gridworld example some states have more than one action with the same action-state value. Then we can choose different optimum policies including those having more than one action for a given state.

Bellman Optimality Equations

Bellman optimality equation is obtained for the best greedy action:

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

Compare with definition in slides 15 and 18: we do not average but take the optimum action



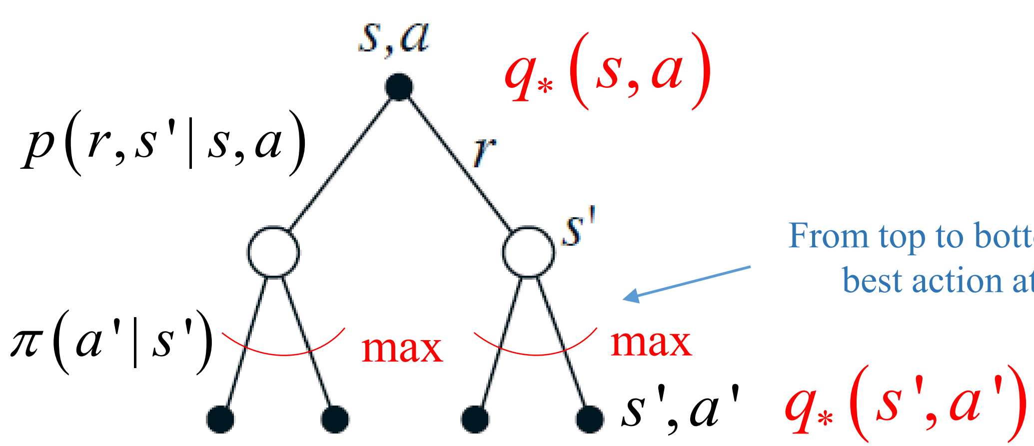
From bottom to top, select the best action at this level

- Non linear, no closed form solution (in general)
- Many iterative solution methods have been developed: Value Iteration, Policy Iteration, Q-learning, Sarsa,… (next lectures).

Bellman Optimality Equations

… and for the action-value function:

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



From top to bottom, select the best action at this level

Quiz #1

Example 2.3: Recycling robot (I)

- A mobile robot has the job of collecting empty soda cans.
- The robot makes its decisions from $\mathcal{A} = \{S, W, R\}$, i.e.

***S*: Search**

***W*: Wait**

***R*: Recharge**

- Two levels or states, high (H) and low (L) as a function of the energy level of the battery, so that the state set is $\mathcal{S} = \{H, L\}$

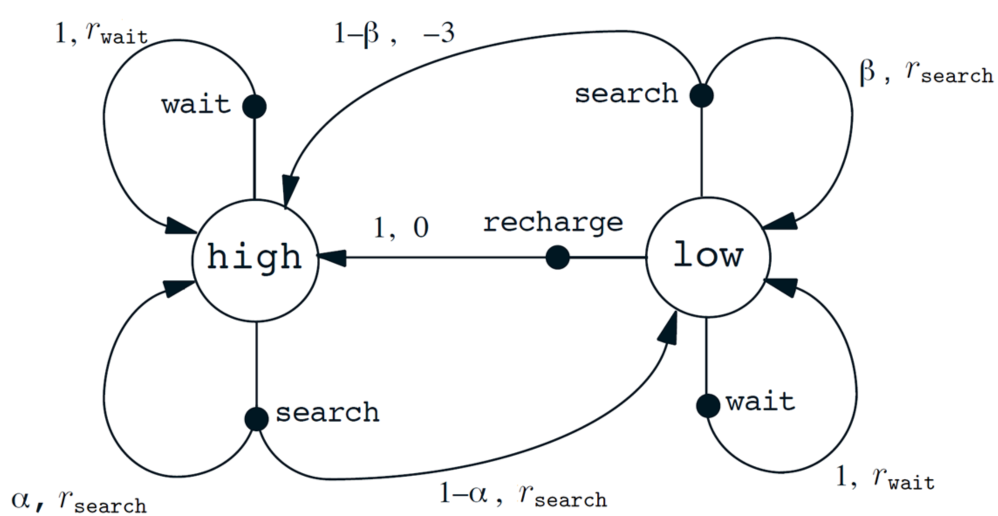
From Sutton & Barto, Reinforcement Learning: An Introduction, 1998

Example 2.3: Recycling robot (II)

Transition probabilities and expected rewards:

s	H	H	L	L	H	H	L	L	L	L
s'	H	L	H	L	H	L	H	L	H	L
a	S	S	S	S	W	W	W	W	R	R
$p(s'\|s,a)$	α	$1-\alpha$	$1-\beta$	β	1	0	0	1	1	0
$r(s,a,s')$	r_s	r_s	-3	r_s	r_w	r_w	r_w	r_w	0	0

Transition graph:

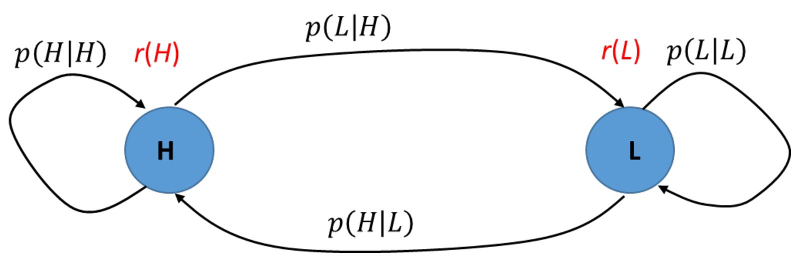


Example 2.3: Recycling robot (III)

Table shows a random policy π_A

$\pi(a/s)$	$\pi(S \mid s)$	$\pi(W \mid s)$	$\pi(R \mid s)$
$s = \mathrm{H}$	0,75	0,25	0
$s = \mathrm{L}$	0,25	0,25	0,5

a) For policy π_A, obtain the reward vector $\mathbf{R}$ and the transition probability matrix $\mathbf{P}$ as functions of α, β, r_s and r_w. Note that this identifies the MRP derived from this MDP by following policy π_A.



b) Define six deterministic policies π_i, $i = 1,\ldots,6$ compute the reward vector and the transition probability matrix for $i = 1,\ldots,6$.

c) Apply brute force to obtain the optimum policy π^* by evaluating in Matlab or Python the value function vector for $i = 1,\ldots,6$ for $\alpha = 0.9$, $\beta = 0.1$, $r_s = 6$ and $r_w = 1$, considering a discount factor $\gamma = 0.9$.

Example 2.3: Recycling robot (IV)

Optimum Policy in terms of r_s, r_w

- Example $\alpha = 0.9$, $\beta = 0.1$, $\gamma = 0.9$