Nel Reinforcement Learning gli MDP sono una formalizzazione della **sequential decision making**, dove le azioni influenzano non solo la *reward* immediata ma anche gli stati e quindi, attraverso questi le reward future.
Negli *MDP* viene stimato il valore $q*(a,s)$ di ogni azione a in ogni stato s.

Gli MDP sono pensati per essere un semplice inquadramento del problema di imparare nel RL: essi descrivono un ambiente per il RL dove lo stato corrente caratterizza il processo.

"Dato il presente il futuro è indipendente dal passato."

> Uno stato S_t è di Markov **SSE** $\mathbf{P}[S_{t+1}|S_t] = \mathbf{P}[S_{t+1}|S_1, \dots, S_t]$

State Transition Matrix

Per uno stato di Markov s e uno stato successore s' la probabilità di *transizione di stato* è definita da:

$$P_{ss'} = \mathbf{P}[S_{t+1} = s'|S_t = s]$$

La matrice di transizione $\mathbf{P}$ definisce la probabilità di transizione da tutti gli stati s ad ogni successore s'.

$$\mathbf{P} = from \begin{bmatrix} \mathbf{P}_{11} & \dots & \mathbf{P}_{1n} \\ \vdots & & \\ \mathbf{P}_{n1} & \dots & \mathbf{P}_{nn} \end{bmatrix}$$

Ogni riga della matrice ha somma 1.

Markov Process

Un processo di Markov è un processo randomico senza memoria ovvero una sequenza di stati random $s1, s2, s3$ aventi la proprietà di Markov.

> Una *Processo di Markov* o **Markov Chain** è una tupla <S,P>, dove:
>
> - S è un insieme finito di stati
> - P è una matrice della probabilità di transizione di stato $P_{ss'} = \mathbf{P}[S_{t+1} = s'|S_t = s]$

Markov Reward Process

Si tratta di una *Markov Chain* con i valori, nello specifico:

> Un **Markov Reward Process** è una tupla <S,P,R,γ> dove:
>
> - S è un insieme di stati
> - P è una State Transition Matrix
> - R è una funzione di *Reward*, $R_s = \mathbf{E}[R_{t+1}|S_t = s]$
> - γ è un *discount factor*

Return value

Il valore di ritorno G_t è la *total discounted reward* dal time-step t.
La reward viene applicata quando si esce dallo stato, indipendentemente dalla decisione che si prende (guarda esempio sotto)

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

- Il *discount factor* $\gamma \in [0, 1]$ è il valore attuale delle *rewards* future.
- Il valore del *reward che viene ricevuta*, ovvero R, dopo k+1 step è $\gamma^k R$
- Abbiamo due reward: *immediate reward* e *delayed reward*
  - γ vicina a 0 ci da una valutazione "miope" o "*short-sighted*"
  - γ vicina a 1 invece ci da una valutazione "longimirante" o "*far-sighted*"

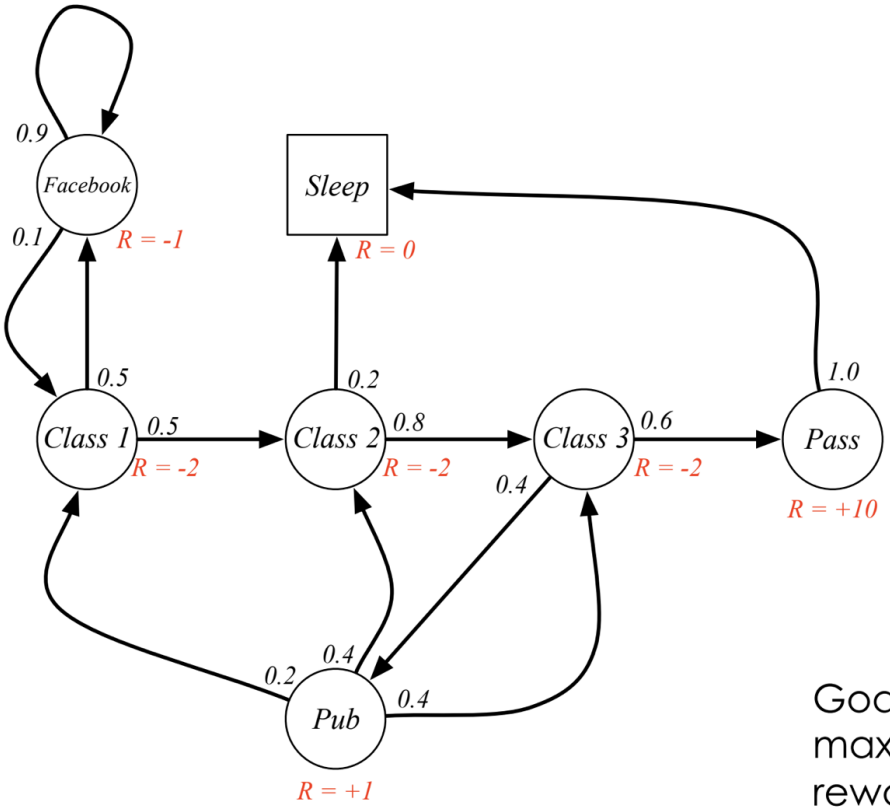
Why the discount?

- Per evitare cicli infiniti nelle *Markov chain*
- L'incertezza potrebbe non essere correttamente rappresentata
- Se la *reward* è finanziaria allora l'*immediate reward* potrebbe essere molto più rilevante della *delayed reward*
- Esseri umani e animali preferiscono la *immediate reward*
- È possibile utilizzare $\gamma = 1$ se tutte le sequenze terminano

The Value function

> La **state value function** V_s di un MRP è il *ritorno* atteso a partire da uno stato s
>
> $$v_s = \mathbf{E}[G_t|S_t = s]$$

Un esempio: (nota che la reward in rosso viene ottenuta prendendo qualsiasi arco uscente)



Goal:
maxir
rewar

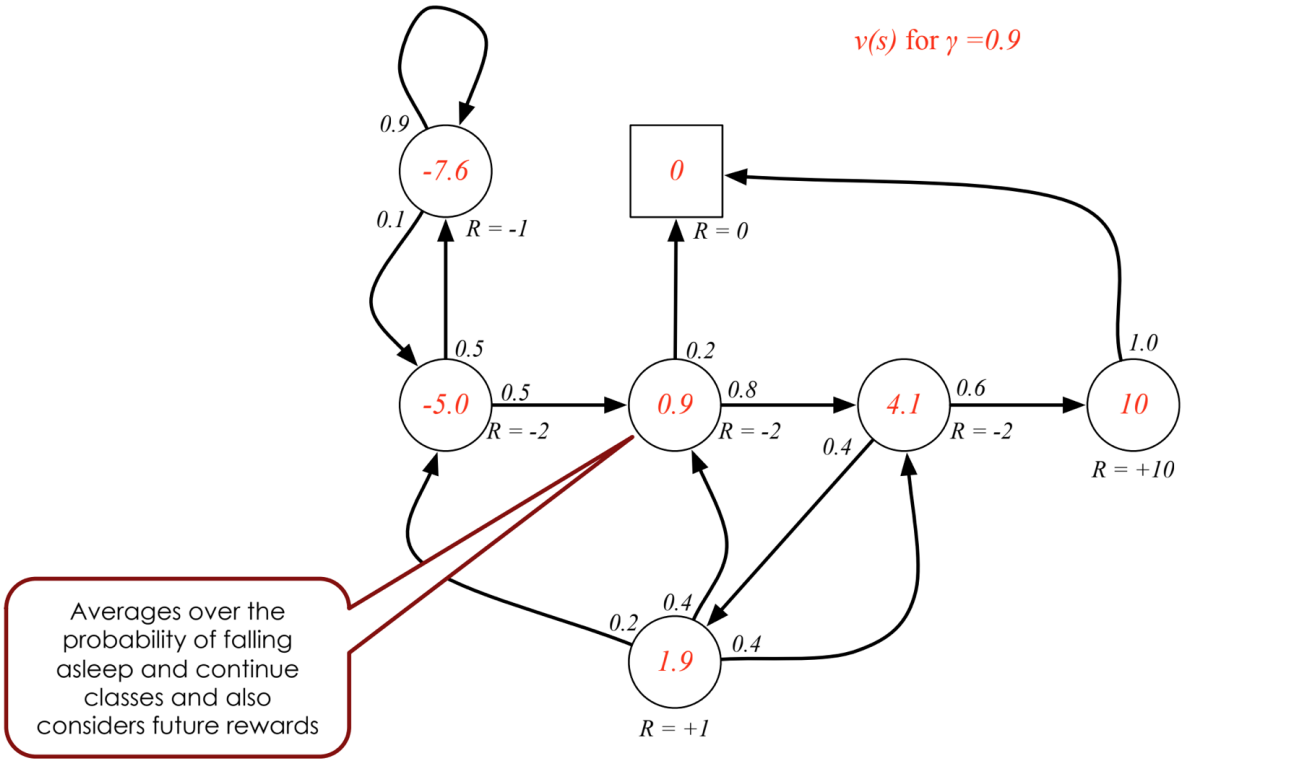
- Sample returns for Student MRP:

Starting from S_1 = C1 with $\gamma = \frac{1}{2}$

$$G_1 = R_2 + \gamma R_3 + \cdots + \gamma^{T-2} R_T$$

C1 C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8}$	=	−2.25
C1 FB FB C1 C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16}$	=	−3.125
C1 C2 C3 Pub C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \ldots$	=	−3.41
C1 FB FB C1 C2 C3 Pub C1 … FB FB FB C1 C2 C3 Pub C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \ldots$	=	−3.20

Altro esempio:



Nell'esempio sopra notare che i valori in rosso indicano il valore calcolato della *value function*.

Come vengono calcolati i valori?

Equazione di Bellman

Data uno stato s_t la reward attesa si calcola utilizzando l'**Equazione di Bellman**:

$$v_s = \mathbf{E}[G_t|S_t = s] = R_{s_t} + \left[P^1_{s_t} \cdot (\gamma \cdot v^1_{s_{t+1}})\right] + \left[P^2_{s_t} \cdot (\gamma \cdot v^2_{s_{t+1}})\right] + \cdots + \left[P^n_{s_t} \cdot (\gamma \cdot v^n_{s_{t+1}})\right]$$

Dove $P^i_{s_t}$ è la probabilità di prendere l'azione $i-$esima nello stato s_t e $V^i_{s_{t+1}}$ è la *value function* calcolata allo stato raggiungibile prendendo la direzione $i-$esima.
I valori vengono calcolati a partire dall'ultimo stato (ovvero quello con reward 10) e poi andando a ritroso.
Infatti, sopra $-5 = 0,5 \cdot [-2 + 0,9 \cdot 0,9] - 0,5 \cdot [-2 + 0,9 \cdot -7,6]$

L'*Equazione di Bellman* ha anche la forma matriciale:

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

Risolvere l'equazione di Bellman

- The Bellman equation is a linear equation
- It can be solved directly:

$$
\begin{aligned}
v &= \mathcal{R} + \gamma \mathcal{P} v \\
(I - \gamma \mathcal{P})\, v &= \mathcal{R} \\
v &= (I - \gamma \mathcal{P})^{-1} \mathcal{R}
\end{aligned}
$$

- Computational complexity is $O(n^3)$ for n states
- Direct solution only possible for small MRPs
- There are many iterative methods for large MRPs, e.g.
  - Dynamic programming
  - Monte-Carlo evaluation
  - Temporal-Difference learning

Markov Decision Process

Un **MDP** è un Markov Reward Process con le *decisioni*.

Definition

A Markov Reward Process is a tuple $\langle S, A, P, R, \gamma \rangle$

- S is a (finite) set of states
- A is a finite set of actions
- P is a state transition probability matrix,
  $P^{a}_{ss'} = \mathbb{P}\,[S_{t+1}=s' \mid S_t=s, A_t=a]$
- R is a reward function,
  $R^{a}_{s} = \mathbb{E}\,[R_{t+1} \mid S_t = s, A_t=a]$
- γ is a discount factor, $\gamma \in [0, 1]$

One matrix for each action

Policy

> Una policy π è una distribuzione sulle azioni avendo noti gli stati.
>
> $$\pi(a|s) = \mathbf{P}[A_t = a|S_t = s]$$

- La policy definisce il comportamento di un agent.
- Le policy nell'MDP dipendono dallo stato corrente e non dalla azioni passate
- Le policy sono indipendenti dal tempo.

Le Value Function in MDP

- **Definizione**: La *state-value* function $v_\pi(s)$ di un MDP è il ritorno atteso a partire da uno stato s e seguendo la policy π

$$v_\pi(s) = \mathbf{E}_\pi[G_t|S_t = s] = \mathbf{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]$$

- **Definizione**: La *action-value* function $q_\pi(a|s)$ è il ritorno atteso a partire da uno stato s, *prendendo l'azione a e seguendo la policy* π.

$$q_\pi(s,a) = [G_t|S_t = s, A_t = a] = \mathbf{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a]$$

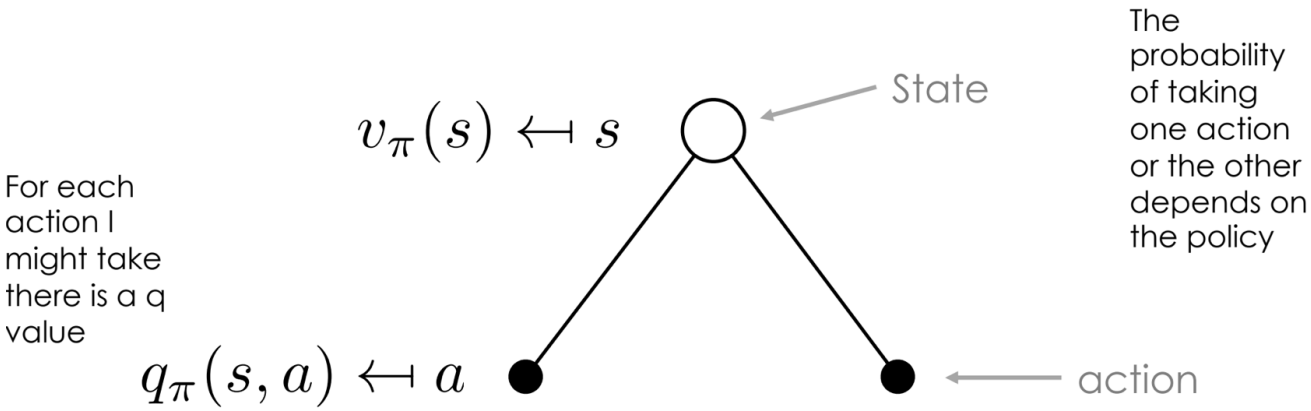
Bellman Expectation Equation for v_π

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s,a)$$

Bellman Expectation Equation for q_π

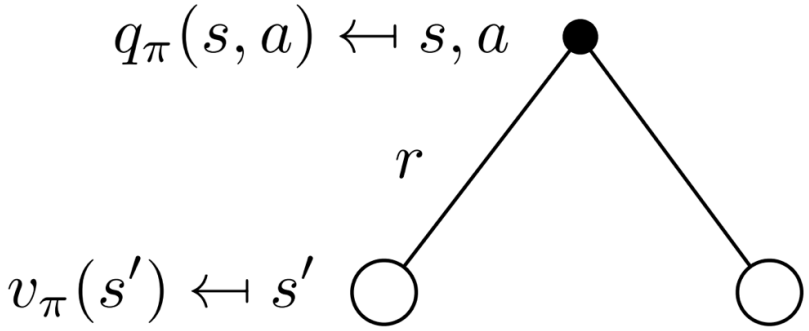
$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')$$

Bellman Expectation Equation for V^{π}



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

Bellman Expectation Equation for Q^{π}



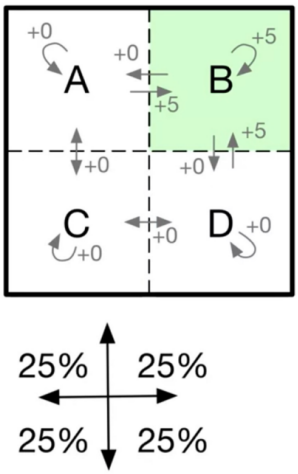
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

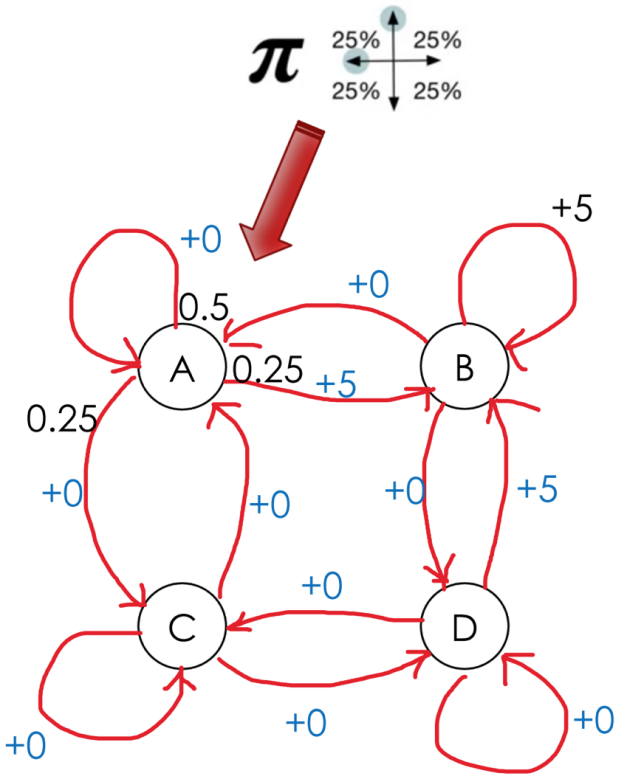
Example: Gridworld

Nell'esempio sotto le percentuali sulle frecce rappresentano la probabilità di andare in quella direzione. L'operazione da fare è rappresentare la griglia con un diagramma a stati.
Nota che la probabilità di rimanere nello stesso stato è data dalla somma delle probabilità presenti lungo le direzioni che non portano ad altre caselle, esempio: in A se vado in alto o a sx non vado da nessuna parte quindi resto in A se sommo le due

probabilità 25% + 25% ottengo 50%.

Example: Gridworld





How to find the optimal policy?

La policy

> La **optimal state-value function** $v_*(s)$ è la value function massimale fra tutte le policy.
>
> $$v_*(s) = \max_\pi v_\pi(s)$$

> La **optimal action-value function** $q_*(s,a)$ è la action-value function massimale fra tutte le policy.
>
> $$q_*(s,a) = \max_\pi q_\pi(s,a)$$

Optimal policy

- Define a partial ordering over policies

$$\pi \geq \pi' \text{ if } V_\pi(s) \geq V_{\pi'}(s), \forall s$$

Theorem

For any Markov Decision Process

- There exists an optimal policy $\pi*$ that is better than or equal to all other policies, $\pi * \geq \pi, \forall \pi$
- All optimal policies achieve the optimal value function, $V_{\pi_*}(s) \geq V_*(s)$
- All optimal policies achieve the optimal action-value function, $q_{\pi_*}(s,a) \geq q_*(s,a)$

Finding an optimal policy

- An optimal policy can be found by maximising over q∗(s,a)

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\text{argmax}}\ q_*(s,a) \\ 0 & \textit{otherwise} \end{cases}$$

- There is always a deterministic optimal policy for any MDP
- If we know q∗(s,a), we immediately have the optimal policy

Come otteniamo il valore q_*?

Bellman optimality equation for v_*

$$v_*(s) = \max_a q_*(s,a)$$

Bellman optimality equation for q_*

$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s')$$

Q-Learning

Può essere di due tipi:

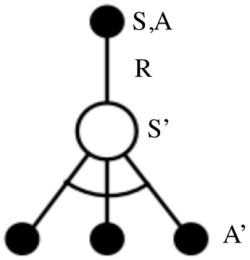
- **On-policy** learning: "Learn on the job"
- **Off-policy** learning: "Impara dall'esperienza di qualcun'altro"

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal,\cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

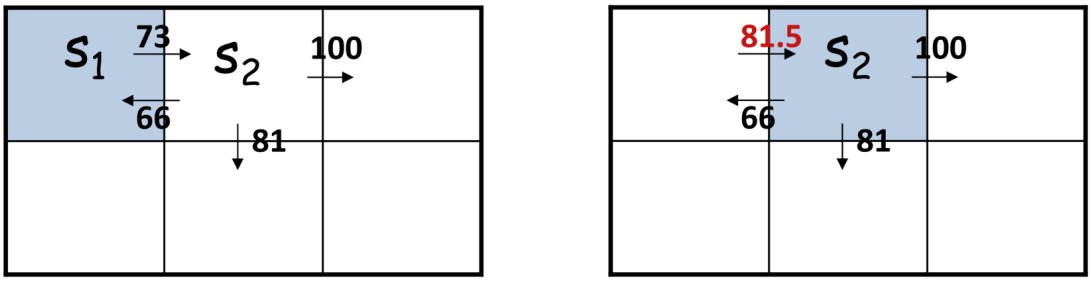
Q-Learning Control Algorithm



$$Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma \max_{a'} Q(S',a') - Q(S,A)\right)$$

- Q-learning control converges to the optimal action-value function, Q(s, a) → q*(s, a)

Example



$\gamma = 0.9, \quad \alpha = 0.5, \quad r = 0$ for non-terminal states

$$\begin{aligned} Q(s_1, right) &= Q(s_1, right) + \alpha\left(r + \gamma \max_{a'} Q(s_2, a') - Q(s_1, right)\right) \\ &= 73 + 0.5(0 + 0.9 \max\{66, 81, 100\} - 73) \\ &= 73 + 0.5(17) \\ &= 81.5 \end{aligned}$$

$$Q(C, Wait) = (1-\alpha)Q(C, Wait) + \alpha[R + \gamma \max_{a} Q(B, a)]$$