Apprendimento per rinforzo

Reinforcement Learning

Definizione del Problema

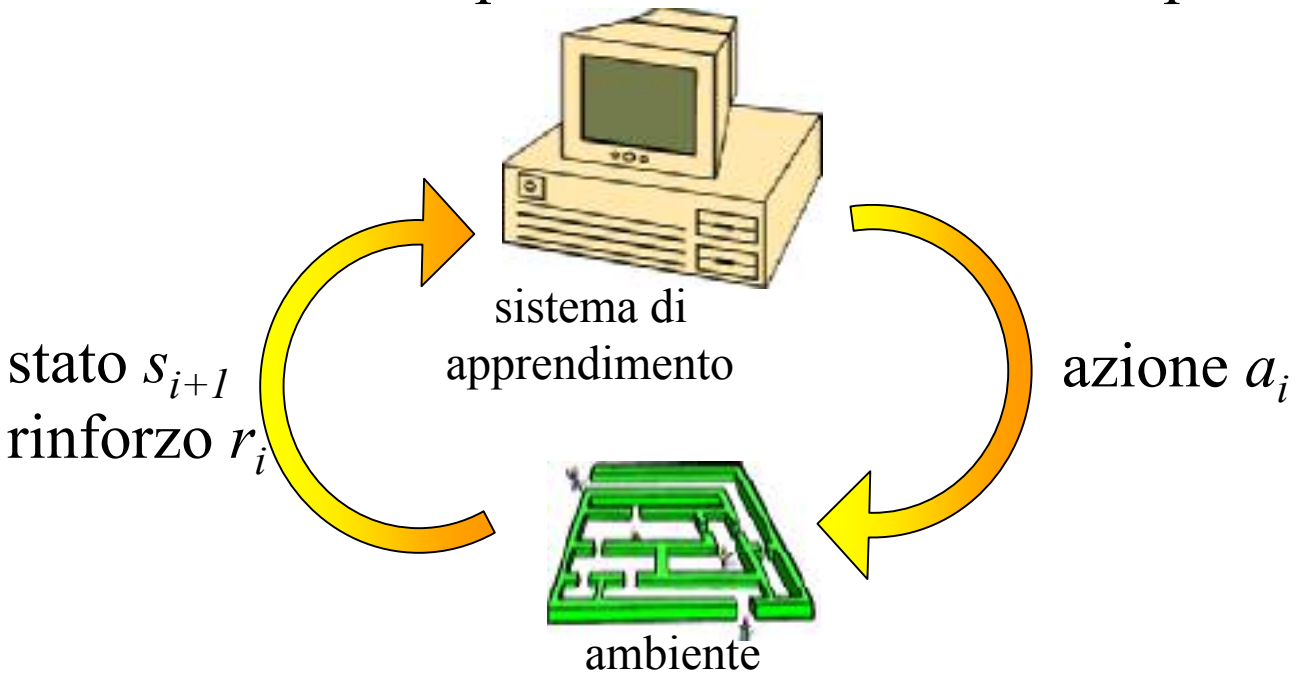
- Non sempre è possibile modellare un problema di apprendimento come la scelta ottimale di una funzione di classificazione
- L'**apprendimento per rinforzo** modella tipicamente il caso di un **agente che percepisce ed agisce in un certo ambiente**, il cui obiettivo è di **imparare a fare "la cosa ottimale"**, o la cosa che lo avvicina di più al suo obiettivo, **dato un certo stato dell'ambiente**
- Sistemi ad agenti
  - Robot
  - Assistenti web
  - ecc.

Definizione del Modello

- Un agente si muove in un ambiente rappresentabile mediante un insieme **S** di stati
- L'agente percepisce un vettore di ingresso, o di percezione, ***e*** che informa l'agente circa lo stato $\mathbf{s_i} \in S$ in cui si trova
- **A** è un insieme di azioni eseguibili dall'agente
- L'esecuzione di un'azione $\boldsymbol{a_i} \in A$ produce una transizione di stato
- **R** è un insieme di ricompense che l'agente può ricevere:
  - $r_i = r(e_i, a_i)$ dove $r_i \in R$, $e_i = f(s_i)$, $a_i \in A$

Obiettivo dell'Apprendimento per Rinforzo

- L'obiettivo è di apprendere una politica ottima $\pi : S \rightarrow A$ che selezioni le azioni successive a ogni stato in modo da massimizzare le ricompense accumulate nel tempo



$$s_0 \xrightarrow{a_0} {}_{r_0} s_1 \xrightarrow{a_1} {}_{r_1} s_2 \xrightarrow{a_2} {}_{r_2} \ldots\ldots$$

Il compito di apprendimento

- Apprendere una politica $\pi : S \rightarrow A$ tale che sia possibile accumulare la ricompensa maggiore nel tempo (secondo π, per ogni stato in S è data un'azione, e dunque lo stato successivo)
- Definiamo $V^{\pi}(s_t)$ come il **valore cumulativo** acquisito tramite una politica arbitraria partendo da uno stato arbitrario s_t:

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

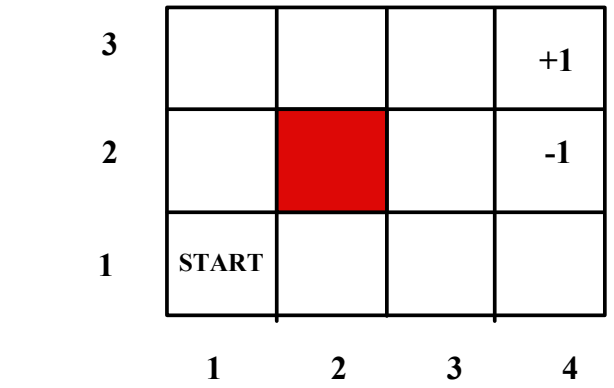
- $0 \leq \gamma < 1$ è una costante che determina il valore relativo di ricompense non immediate. Ad esempio, se una ricompensa r_{t+i} è ricevuta dopo i passi dall'attuale stato, viene "scontata" di un fattore γ^i.
- Il compito di apprendimento è ottenere una politica ottima π^* tale che:

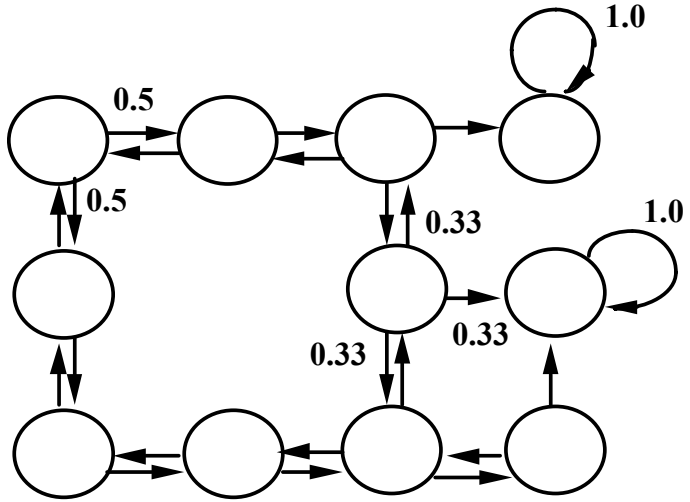
$$\pi^* = \arg\max_{\pi} V^{\pi}(s), \quad \forall s \in S$$

Varianti

- **Ricompensa ritardata**: a volte la ricompensa non corrisponde a singole azioni, ma al raggiungimento di qualche obiettivo o sotto-obiettivo (es. per un robot calciatore: fare goal o fare una “bella azione”)
- **Esplorazione**: l’apprendista può restare all’interno di ambienti (stati) noti o può decidere di esplorare stati non noti
- **Percezione parziale**: l’agente può non percepire l’intera realtà (ad es. per via di ostacoli visivi o perché non può vedere dietro di sé)

Esempio: Apprendimento passivo in ambiente noto





- Un robot deve muoversi in una scacchiera. Le azioni possibili sono N, S, E, O (nord,sud, est,ovest)
- Lo stato (1,1) è lo stato di partenza, (3,4) e (2,4) sono terminali con **ricompense** +1 e -1. Lo stato (2,2) è inaccessibile. In ogni stato le **probabilità** di transizione verso altri stati sono uguali. Le **percezioni** coincidono con l'identificazione dello stato in cui il robot si trova.

Q-learning

- Ogni casella rappresenta uno **stato**, ogni mossa è un'**azione**.
- La funzione $r(s, a)$ fornisce una **ricompensa** solo negli stati in cui essa è prevista, e zero in tutti gli altri stati.
- Indichiamo con $\delta(s, a)$ lo stato in cui il sistema transita se parte da s ed esegue l'azione a e indichiamo V^{π^*} con V^*
- Allora possiamo scegliere l'azione per lo stato s sulla base della ricompensa immediata più il valore cumulativo per lo stato raggiunto sulla base dell'azione intrapresa:

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

Sfortunatamente, questo ha senso solo se il sistema conosce la funzione r e la funzione di transizione $\delta(s, a)$!

Q-learning (2)

- L'input del sistema sono sequenze di transizioni, dette **episodi**. Un episodio inizia in uno stato scelto a caso, e termina quando l'agente raggiunge, dopo una serie di azioni, uno stato obiettivo.
- Definiamo la quantità: $Q(s, a) = r(s, a)+\gamma V^*(\delta(s, a))$
- Possiamo perciò riscrivere la precedente equazione:

$$V^*(s) = \max_{a'} Q(s, a')$$

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

- Il sistema deve dunque stimare le quantità Q, indicheremo $\hat{Q}$ con le stime di Q. La precedente definizione ricorsiva di Q consente di stimare le $\hat{Q}$ mediante **approssimazioni iterative**.
- Inizialmente i valori di stima possono essere riempiti casualmente

Q-learning (3)

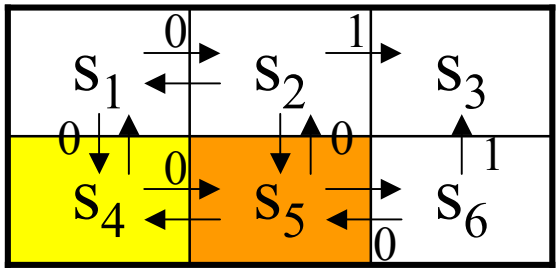
- Per ogni (s, a) inizializza $\hat{Q}(s,a)$ a zero (o a un valore *random*)
- Per ogni epoca
  - Seleziona lo stato iniziale s
  - Esegui ripetutamente:
    - Seleziona un'azione possibile a in s ed eseguila
    - Ricevi una ricompensa r
    - Osserva il nuovo stato s' cui si arriva eseguendo a
    - Aggiorna le stime come segue: $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
    - $s \leftarrow s$'

Q-learning: Esempio

- **Spostamenti di un agente in una griglia:**
  - Il sistema inizialmente pone a zero tutte le stime di $\hat{Q}$.
  - In tal modo, non potrà fare cambiamenti nella tabella finché, eventualmente, non raggiunge uno stato che prevede una ricompensa (primo episodio)
    $(1,1) \rightarrow (2,1) \rightarrow (3,1) \rightarrow (3,2) \rightarrow (3,3) \rightarrow$ (**3,4**)
  - Questo ha l'effetto di far aggiornare il valore di $\hat{Q}$ per la sola transizione che porta allo stato obiettivo, ad esempio s_k (3,3 nell'esempio)

    ($Q((3,3),a_{right})$ nell'esempio)
  - Nel successivo episodio, se l'agente passa per s_k, il valore non nullo di $\hat{Q}$ consentirà di aggiornare il valore di $\hat{Q}_k$ per qualche transizione due celle prima della cella obiettivo (cioè (3,4)), e, se il numero di episodi è sufficiente, l'informazione si propaga all'indietro per tutti gli stati.

Q-Learning: Esempio (γ=0.9)

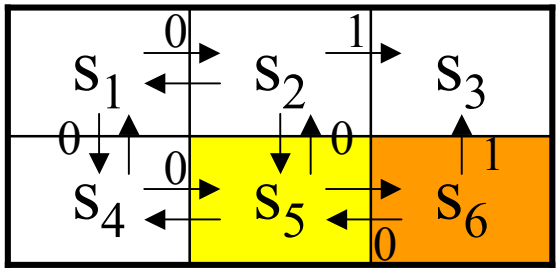


$s=s_4$, $s'=s_5$

- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue:
  $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- $s \leftarrow s'$

	N	S	O	E
s_1	0	0	0	0
s_2	0	0	0	0
s_3	0	0	0	0
s_4	0	0	0	**0+0.9*0 =0**
s_5	0	0	0	0
s_6	0	0	0	0

Q-Learning: Esempio (γ=0.9)

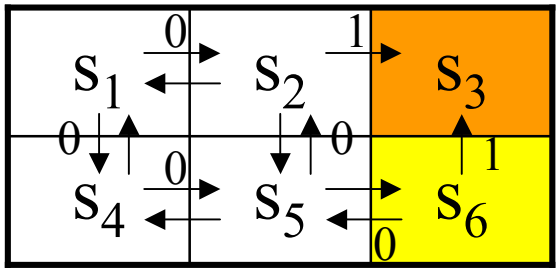


$s=s_5$, $s'=s_6$

- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue:
  $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- $s \leftarrow s'$

	N	S	O	E
s_1	0	0	0	0
s_2	0	0	0	0
s_3	0	0	0	0
s_4	0	0	0	**0**
s_5	0	0	0	**0+0.9*0 =0**
s_6	0	0	0	0

Q-Learning: Esempio (γ=0.9)

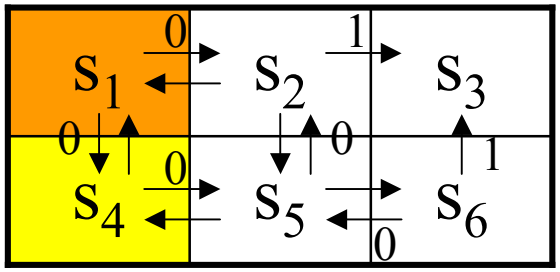


$s=s_6$, $s'=s_3$

- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue:
  $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- **GOAL RAGGIUNTO: FINE PRIMO EPISODIO**

	N	S	O	E
s_1	0	0	0	0
s_2	0	0	0	0
s_3	0	0	0	0
s_4	0	0	0	**0**
s_5	0	0	0	**0**
s_6	**1+0.9*0 =1**	0	0	0

Q-Learning: Esempio (γ=0.9)

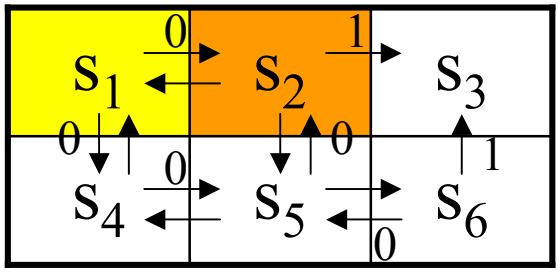


s=s4, s'=s1

- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue:
  $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- $s \leftarrow s'$

	N	S	O	E
s_1	0	0	0	0
s_2	0	0	0	0
s_3	0	0	0	0
s_4	**0+0.9*0 =0**	0	0	**0**
s_5	0	0	0	**0**
s_6	**1**	0	0	0

Q-Learning: Esempio (γ=0.9)

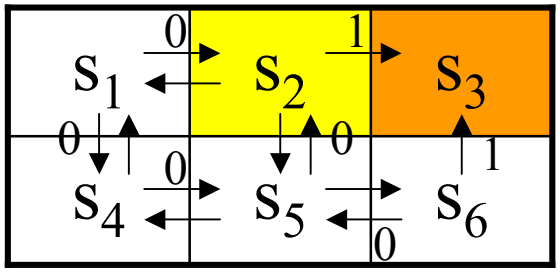


$s=s_1$, $s'=s_2$

- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue:
  $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- $s \leftarrow s'$

	N	S	O	E
s_1	0	0	0	**0+0.9*0 =0**
s_2	0	0	0	0
s_3	0	0	0	0
s_4	**0**	0	0	**0**
s_5	0	0	0	**0**
s_6	**1**	0	0	0

Q-Learning: Esempio (γ=0.9)

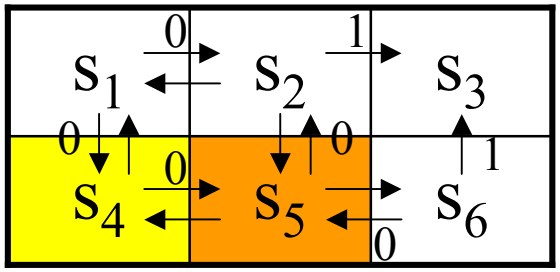


$s=s_2$, $s'=s_3$

- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue:
  $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- **GOAL RAGGIUNTO: FINE SECONDO EPISODIO**

	N	S	O	E
s_1	0	0	0	**0**
s_2	0	0	0	**1+0.9*0 =1**
s_3	0	0	0	0
s_4	**0**	0	0	**0**
s_5	0	0	0	**0**
s_6	**1**	0	0	0

Q-Learning: Esempio (γ=0.9)

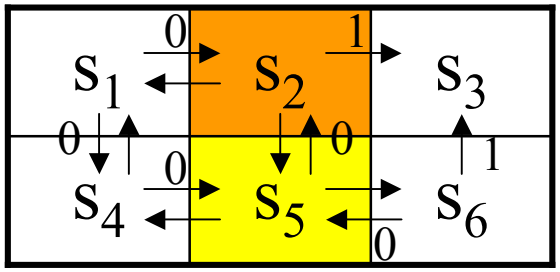


$s=s_4$, $s'=s_5$

- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue:
  $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- $s \leftarrow s'$

	N	S	O	E
s_1	0	0	0	**0**
s_2	0	0	0	**1**
s_3	0	0	0	0
s_4	**0**	0	0	**0+0**
s_5	0	0	0	**0**
s_6	**1**	0	0	0

Q-Learning: Esempio (γ=0.9)

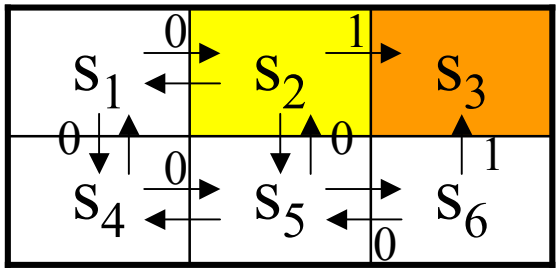


$s=s_5$, $s'=s_2$

- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue:
  $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- $s \leftarrow s'$

	N	S	O	E
s_1	0	0	0	**0**
s_2	0	0	0	**1**
s_3	0	0	0	0
s_4	**0**	0	0	**0**
s_5	**0+0.9*1 =0.9**	0	0	**0**
s_6	**1**	0	0	0

Q-Learning: Esempio (γ=0.9)

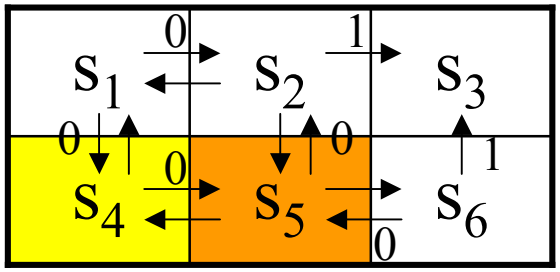


$s=s_2$, $s'=s_3$

- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue: $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- **GOAL RAGGIUNTO: FINE TERZO EPISODIO**

	N	S	O	E
s_1	0	0	0	**0**
s_2	0	0	0	**1+0**
s_3	0	0	0	0
s_4	**0**	0	0	**0**
s_5	**0.9**	0	0	**0**
s_6	**1**	0	0	0

Q-Learning: Esempio (γ=0.9)

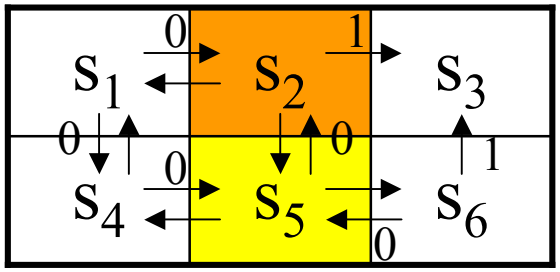


$s=s_4$, $s'=s_5$

- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue:
  $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- $s \leftarrow s'$

	N	S	O	E
s_1	0	0	0	**0**
s_2	0	0	0	**1**
s_3	0	0	0	0
s_4	**0**	0	0	**0+0.9*0.9 =0.81**
s_5	**0.9**	0	0	**0**
s_6	**1**	0	0	0

Q-Learning: Esempio (γ=0.9)

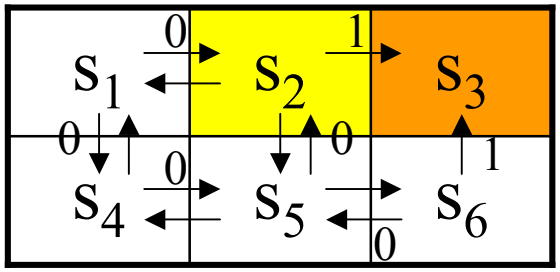


$s=s_5$, $s'=s_2$

- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue:
  $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- $s \leftarrow s'$

	N	S	O	E
s_1	0	0	0	**0**
s_2	0	0	0	**1**
s_3	0	0	0	0
s_4	**0**	0	0	**0.81**
s_5	**0+0.9*1 =0.9**	0	0	**0**
s_6	**1**	0	0	0

Q-Learning: Esempio (γ=0.9)

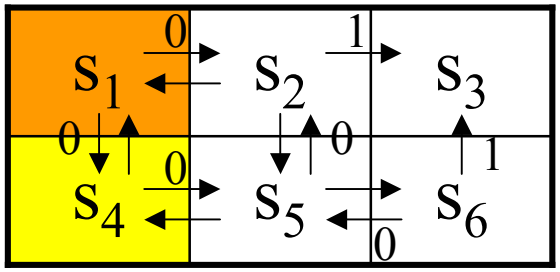


$s=s_2$, $s'=s_3$

- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue: $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- **GOAL RAGGIUNTO: FINE QUARTO EPISODIO**

	N	S	O	E
s_1	0	0	0	**0**
s_2	0	0	0	**1+0**
s_3	0	0	0	0
s_4	**0**	0	0	**0.81**
s_5	**0.9**	0	0	**0**
s_6	**1**	0	0	0

Q-Learning: Esempio (γ=0.9)

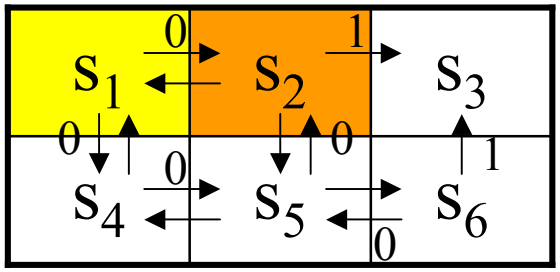


$s=s_4$, $s'=s_1$

- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue:
  $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- $s \leftarrow s'$

	N	S	O	E
s_1	0	0	0	**0**
s_2	0	0	0	**1**
s_3	0	0	0	0
s_4	**0+0**	0	0	**0.81**
s_5	**0.9**	0	0	**0**
s_6	**1**	0	0	0

Q-Learning: Esempio (γ=0.9)

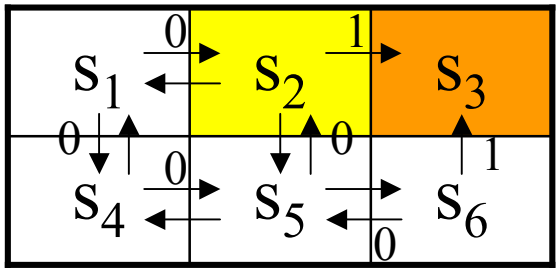


$s=s_1$, $s'=s_2$

- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue:
  $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- $s \leftarrow s'$

	N	S	O	E
s_1	0	0	0	**0+0.9*1 =0.9**
s_2	0	0	0	**1**
s_3	0	0	0	0
s_4	**0**	0	0	**0.81**
s_5	**0.9**	0	0	**0**
s_6	**1**	0	0	0

Q-Learning: Esempio (γ=0.9)



$s=s_2$, $s'=s_3$

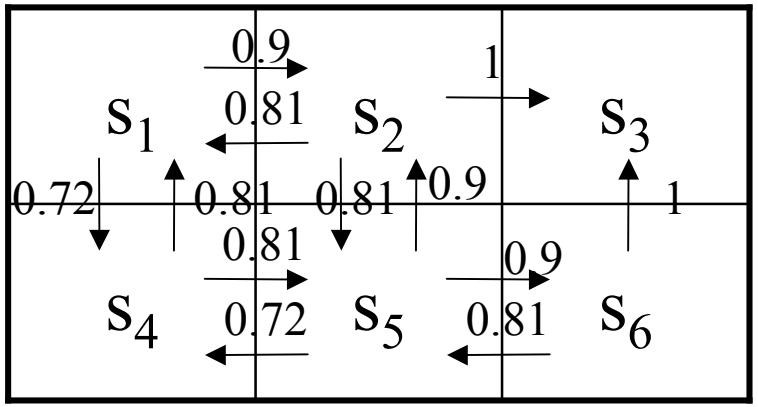
- Seleziona un'azione possibile *a* in *s* ed eseguila
- Ricevi una ricompensa *r*
- Osserva il nuovo stato *s'* cui si arriva eseguendo a
- Aggiorna le stime come segue:
  $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- **GOAL RAGGIUNTO: FINE QUINTO EPISODIO**

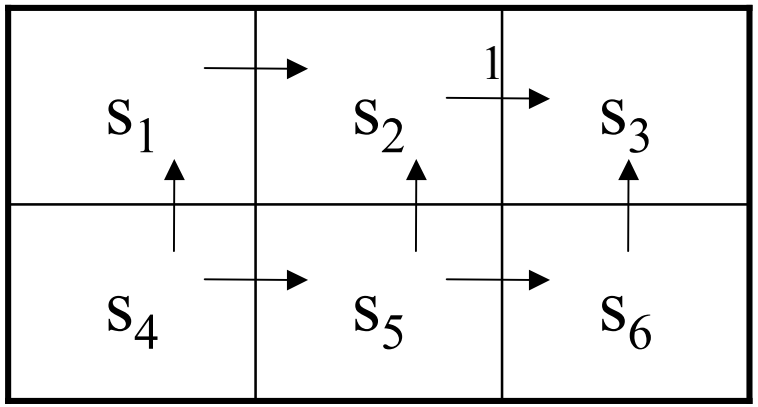
	N	S	O	E
s_1	0	0	0	**0.9**
s_2	0	0	0	**1+0**
s_3	0	0	0	0
s_4	**0**	0	0	**0.81**
s_5	**0.9**	0	0	**0**
s_6	**1**	0	0	0

Q-Learning: Esempio (γ=0.9)

Al termine dell'esecuzione, l'algoritmo converge alla seguente tabella:

	N	S	O	E
s_1	0	**0.72**	0	**0.9**
s_2	0	**0.81**	**0.81**	**1**
s_3	0	0	0	0
s_4	**0.81**	0	0	**0.81**
s_5	**0.9**	0	**0.72**	**0.9**
s_6	**1**	0	**0.81**	0





Q-learning: converge?

- Si dimostra che l'algoritmo **converge se**:
  - primo, il sistema può essere modellato come un processo di Markov (le probabilità di transizione fra stati devono essere note a priori)
  - Secondo, i valori di ricompensa devono essere limitati da una costante c, ovvero: $|r(s, a)| \leq c$
  - Terzo, l'agente deve selezionare le azioni in modo tale che ogni coppia (s, a) sia visitata un numero infinito di volte
- In pratica, si dimostra che l'algoritmo funziona in condizioni meno restrittive

Come scegliere le azioni per calcolare le stime di Q?

- L'algoritmo non specifica come scegliere la prossima azione a partire dallo stato *s*
- **Strategia ovvia:**
  - Scegliere l'azione *a* che massimizza $\hat{Q}(s,a)$
  - **Rischio:** seguire sempre le stesse azioni e tralasciare azioni con valori potenzialmente ancora più alti
- **Strategia probabilistica:**
  - Si sceglie l'azione *a* dallo stato s con la seguente probabilità (k > 0 determina l'influenza della selezione verso alti valori di Q):

$$P(a \mid s) = \frac{k^{\hat{Q}(s,a)}}{\sum_{a' \in A} k^{\hat{Q}(s,a')}}$$

Ricompense e Azioni Non Deterministiche

- Consideriamo il caso in cui la funzione ricompensa $r(s, a)$ e la funzione di transizione $\delta(s, a)$ abbiano esiti probabilistici
  - Producono una distribuzione di probabilità sugli esiti dato lo stato s e l'azione a ed estraggono casualmente un esito sulla base di questa distribuzione
- Dobbiamo generalizzare il valore V^{π} della politica π al valore atteso (sugli esiti non deterministici) dei compensi accumulati nel tempo:

$$V^{\pi}(s_t) = E\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i}\right]$$

- dove r_{t+i} è sempre generata seguendo la politica π a partire dallo stato s

Q-Learning per il caso non deterministico (1)

$$
\begin{aligned}
Q(s,a) &= E[r(s,a) + \gamma V^*(\delta(s,a))] \\
&= E[r(s,a)] + \gamma E[V^*(\delta(s,a))] \\
&= E[r(s,a)] + \gamma \sum_{s' \in S} P(s'|s,a) V^*(s')]
\end{aligned}
$$

- Dato che, come prima:

$$V^*(s) = \max_{a'} Q(s,a')$$

- Possiamo riscrivere:

$$Q(s,a) = E[r(s,a)] + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a' \in A} Q(s',a')$$

Q-Learning per il caso non deterministico (2)

- Per assicurare la convergenza, dobbiamo modificare infine anche la regola di allenamento dell'algoritmo
- A tal fine, abbiamo bisogno di far "decadere" nel tempo la stima rivista a favore del valore precedente della stima di Q:

$$\hat{Q}_n(s,a) \leftarrow (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \alpha_n[r + \max_{a' \in A} \hat{Q}_{n-1}(s',a')]$$

- Dove *s* e *a* sono lo stato e l'azione aggiornate durante l'*n*-esima iterazione e $visite_n(s, a)$ è il numero totale di volte che la coppia (s,a) è stata visitata finora (inclusa l'*n*-esima iterazione):

$$\alpha_n = \frac{1}{1 + visite_n(s,a)}$$

Applicazioni

- **TD-Gammon**: Tesauro
  - Il miglior programma di backgammon
- **Elevator Control**: Crites & Barto
  - Controllore di ascensori a elevate prestazioni
- **Dynamic Channel Assignment**: Singh & Bertsekas, Nie & Haykin
  - Prestazioni elevate nell'assegnare canali radio a chiamate di telefonia mobile
- **Traffic light control:** Wiering et al., Choy et al.
  - Elevate prestazioni nel controllo dei semafori per ottimizzare il flusso del traffico
- **Water systems control:** Bhattacharya et al.
  - Elevate prestazioni nel controllo dei livelli d'acqua dei sistemi idrici regionali