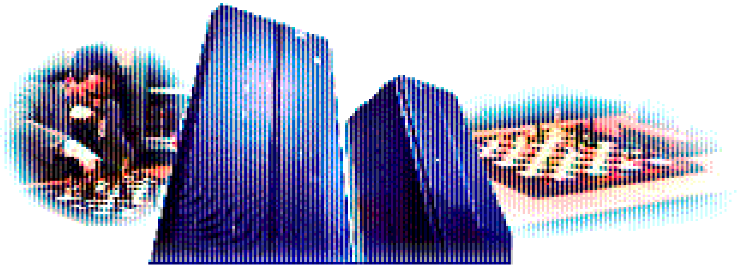
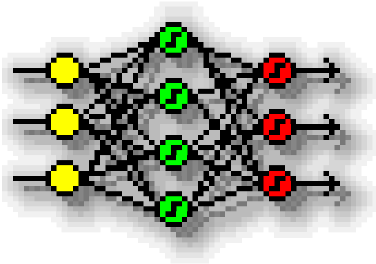




Sistemi Intelligenti Reinforcement Learning: Q-learning

Alberto Borghese

Università degli Studi di Milano
Laboratorio di Sistemi Intelligenti Applicati (AIS-Lab)
Dipartimento di Informatica
borghese@di.unimi.it

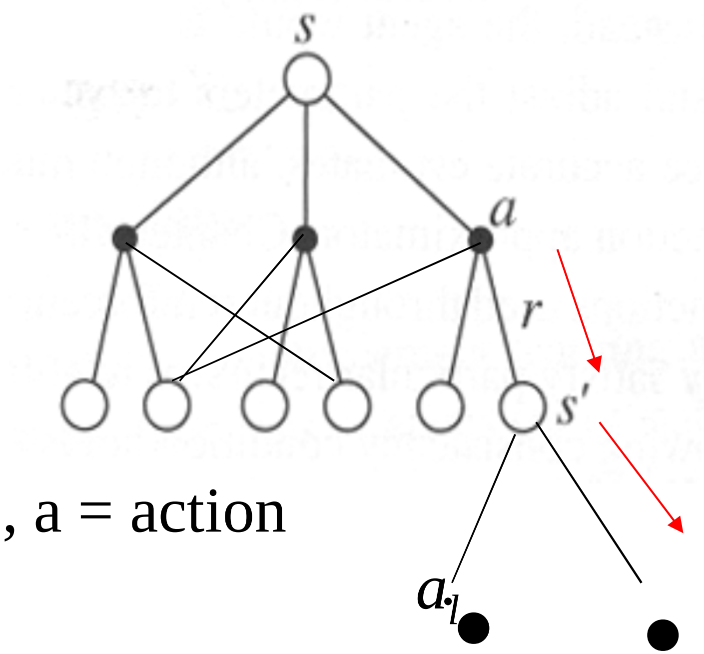
Sommario

Q-learning

Come apprendere Q: SARSA

$$Q^{\pi}(s_t,a_t)=Q^{\pi}(s_t,a_t)+\alpha\left[r_{t+1}+\gamma Q^{\pi}(s_{t+1},a_{t+1})-Q^{\pi}(s_t,a_t)\right]$$

1) Apprendiamo il valore di Q per una policy data, π, (on-policy).



2) Dopo avere appreso la funzione Q, possiamo modificare la policy in modo da migliorarla (**policy improvement**)

S = state, a = action, r = reward, s = state, a = action

On-policy learning.

Value iteration

$$Q^{\pi}{}_{k+1}(s,a) = \sum_{s'} P_{s \to s'|a_j} \left\{ R_{s \to s'|a_j} + \gamma \left[\sum_{a'_j} \pi\left(a_j, s'\right) \right] Q^{\pi}{}_{k+1}(s', a'_j) \right\}$$

Invece di considerare una policy stocastica, consideriamo l'azione migliore in base al reward atteso a lungo termine per quella azione a':

$$Q_{k+1}(s,a) = \sum_{s'} P_{s \to s'|a} \left[R_{s \to s'|a} + \gamma \max_{a'} Q_k(s', a') \right]$$

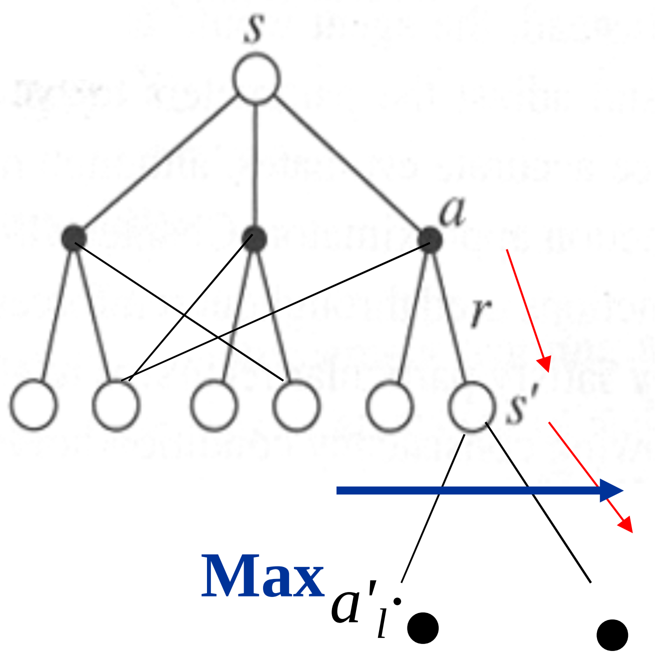
$\forall s$

Off-policy Temporal Difference: Q-learning

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\right]$$

Non imparo semplicemente la funzione valore Q, ma la funzione valore Q ottima.

In s, scelgo un ramo del grafo, e poi **decido** ad un passo come continuare, guardando il reward a lungo termine stimato per le diverse azioni.

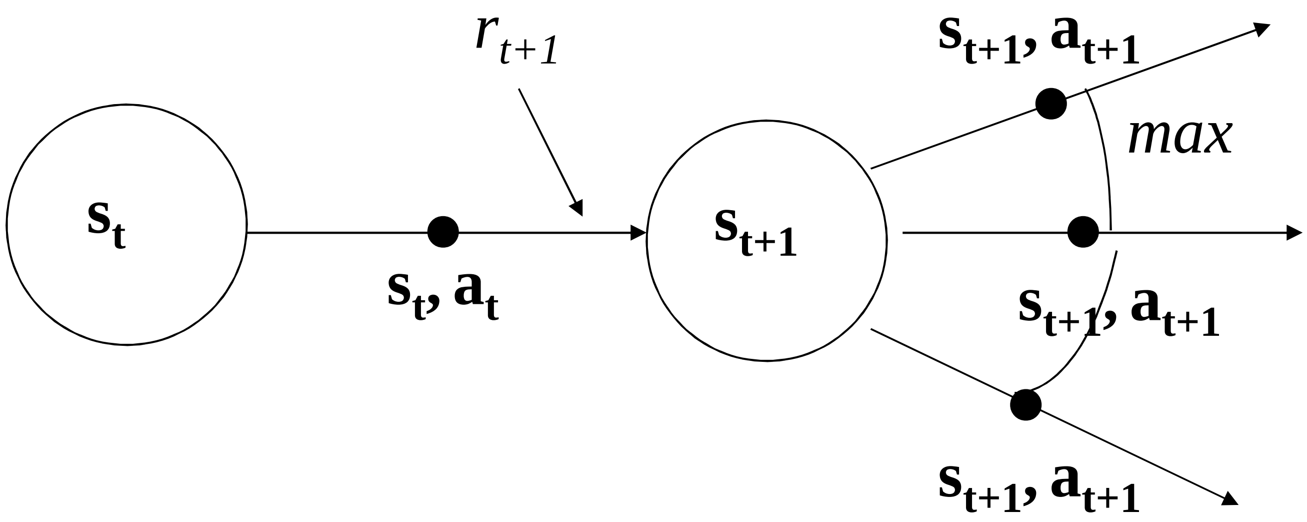


Q-learning algorithm (progetto)

```
Q(s,a) = 0;          // ∀s, ∀a,
Policy data
Repeat                                                  // for each episode
{       s = s0; a = Policy(s); PolicyStable = true;    // eventualmente ε-greedy
        Repeat                                          // for each step of the single episode
        {       s_next = NextState(s,a);
                reward = Reward(s, s_next, a);
                a_next_pol = Policy(s_next);                 // on policy
                a_next = argmax(Q(s_next, a);
                               a

                if (a_next_pol != a_next)
                {    UpdatePolicy(s_next, a_next); PolicyStable = false; }
                endif;
                Q(s,a) = Q(s,a) + α [reward + γ Q(s_next, a_next) – Q(s,a)];
                s = s_next;
                a = a_next;          // a = Policy(s = s_next)
        }   // until last state
}   // until the end of learning
```

Rappresentazione grafica



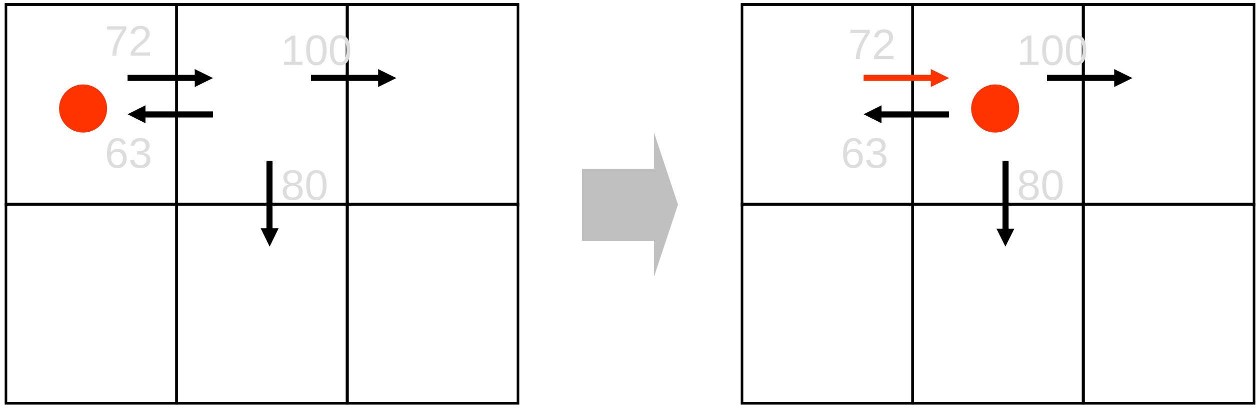
$Q(\mathbf{s_t}, \mathbf{a_t})$ $Q(\mathbf{s_{t+1}}, \mathbf{a_{t+1}})$

One step for **Q Iteration**

Viene migliorata la policy al tempo t+1.

Example 1 - Q Learning Update

$\gamma = 0.9$



0 reward received in the transition

Esempio tratto dai lucidi del corso di Brian C. Williams su RL.

Modificati dalle slide di: Manuela Veloso, Reid Simmons, & Tom Mitchell, CMU

Apprendimento della funzione valore Q. Versione Q-learning. Q(A,dx) = ?

A	B	C
D	E	F

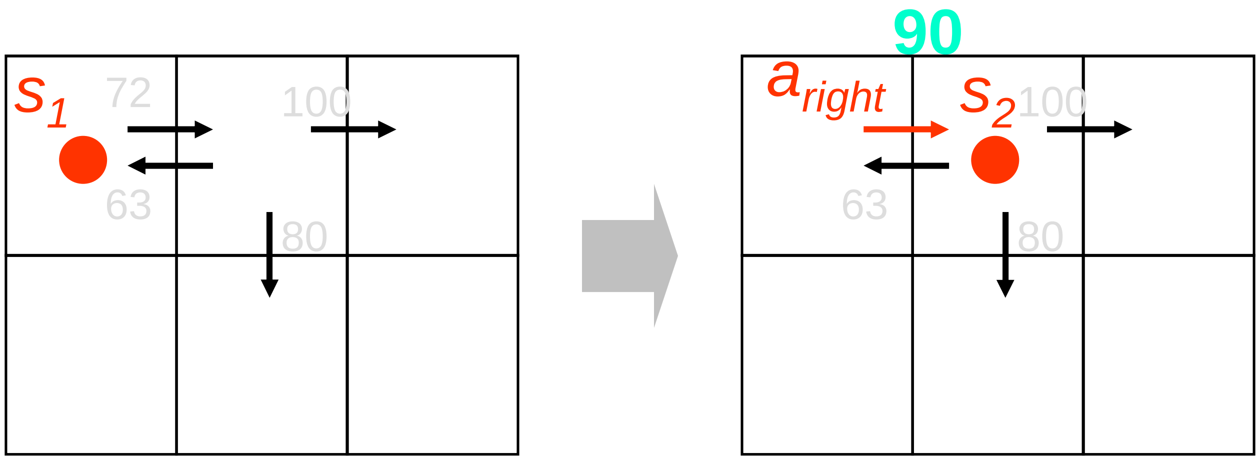
In grigio i valori di Q(s,a).
Nessun reward istantaneo.

Example 1 – $\underline{Q}$ Learning Update

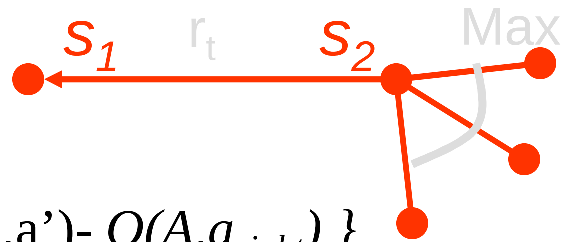
$\gamma = 0.9$

$\alpha = 0.1$

a(s2) = down



0 reward received in the transition



$$\underline{Q}(A,a_{right}) \leftarrow \underline{Q}(A,a_{right}) + \alpha\{ r(A,a_{right},, B) + \gamma \max_{a'} \underline{Q}(B,a') - \underline{Q}(A,a_{right}) \}$$

$$\leftarrow 72 + \alpha\, [0 + 0.9 \max\{63, 80, 100\} - \underline{Q}(A,a_{right})\,]$$

$$\leftarrow 72 + \alpha(90 - 72) = 72 + 1.8 = 73.8$$

Correzione di $\underline{Q}(A,a_{right})$

Correzione dell'azione in B da down a right

La correzione di $\underline{Q}(A,a_{right})$ va a 0

quando $\underline{Q}(A,a_{right}) = 90$

$$Q(B,a_{down}) = 80$$

$$Q(B,a_{right}) = 100$$

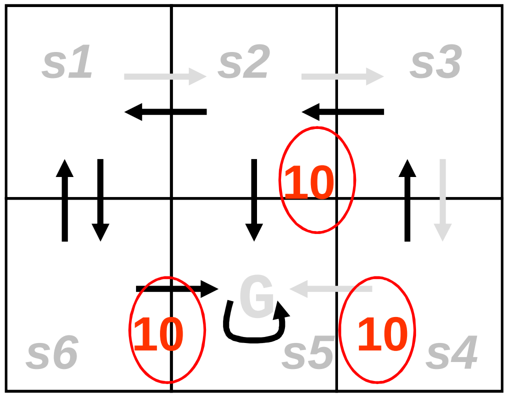
$$Q(B,a_{left}) = 63$$

Example 2: Q-Learning Iterations: Episodic

- Start at upper left; Initial selected policy: move clockwise; Table initially 0; $\gamma = 0.8$.

Possibili transizione sono segnate con frecce nere e grigie.

Reward istanteo in rosso e cerchiato



$\alpha = 1$

$$Q(s_t, a_t) \leftarrow \left[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \right]$$

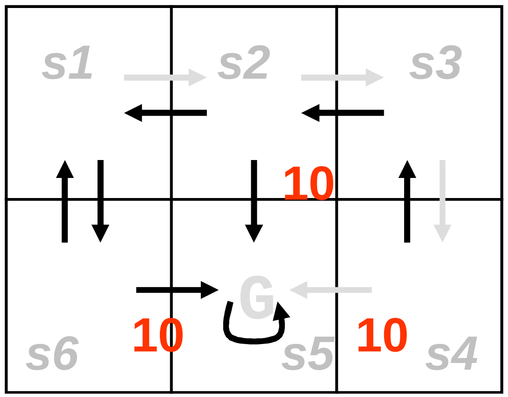
E.g. videogioco.
In G rimango in G - loop

Q(S1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0			

Q-Learning Iterations

- Start at upper left – move clockwise; table initially 0; $\gamma = 0.8$; $\alpha = 1$

$$\underline{Q}(s, a) \leftarrow r + \gamma \max_{a'} \underline{Q}(s',a')$$

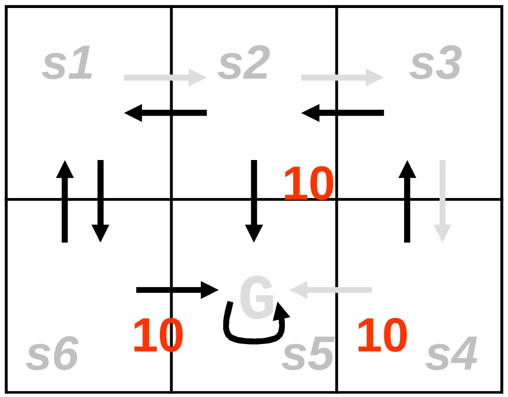


Q(S1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	

Q-Learning Iterations

- Start at upper left – move clockwise; $\gamma = 0.8$

$$\underline{Q}(s, a) \leftarrow r + \gamma \max_{a'} \underline{Q}(s',a')$$

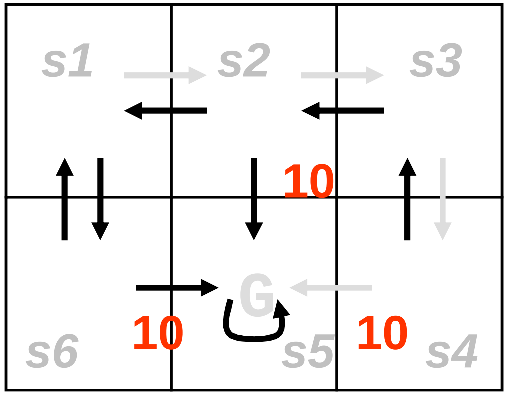


Q(S1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	$r + \gamma \max_{a'} \{\underline{Q}(s5,a) =$ $10 + 0.8 \times 0 = \mathbf{10}$

Q-Learning Iterations

- Start at upper left – move clockwise; $\gamma = 0.8$

$$\underline{Q}(s, a) \leftarrow r + \gamma \max_{a'} \underline{Q}(s',a')$$

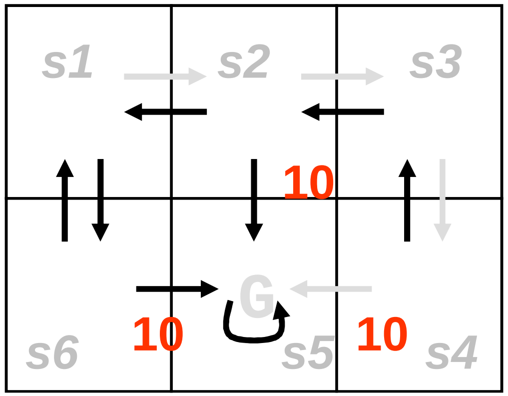


Q(S1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	$r + \gamma \underline{Q}(s5,loop)$ = 10 + 0.8 x 0 = **10**
0	0	$r + \gamma \max_{a'} \{\underline{Q}(s4,W), \underline{Q}(s4,N)\}$ = 0 + 0.8 x max{10,0) = **8**	

Q-Learning Iterations

- Start at upper left – move clockwise; $\gamma = 0.8$

$$\underline{Q}(s, a) \leftarrow r + \gamma \max_{a'} \underline{Q}(s',a')$$

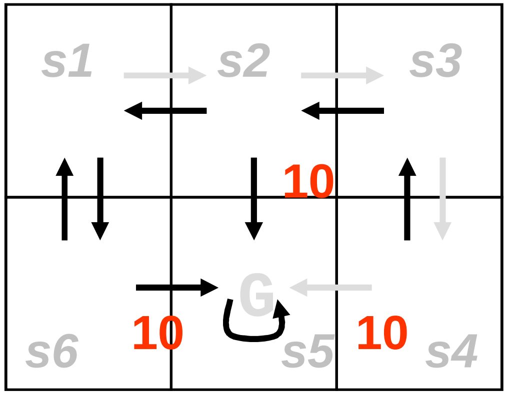


Q(S1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	$r + \gamma\ \{\underline{Q}(s5,loop)-Q(s4,W)\}=$ $10 + 0.8 \times 0 - 0 = \mathbf{10}$
0	0	$r + \gamma\ \underline{Q}(s4,W)$ $= 0 + 0.8 \times 10 = \mathbf{8}$	10
0	$r + \gamma \max_{a'} \{\underline{Q}(s3,W), \underline{Q}(s3,S)\}$ $= 0 + 0.8 \times \max\{0,8\} = \mathbf{6.4}$		

Q-Learning Iterations

- Start at upper left – move clockwise; $\gamma = 0.8$

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

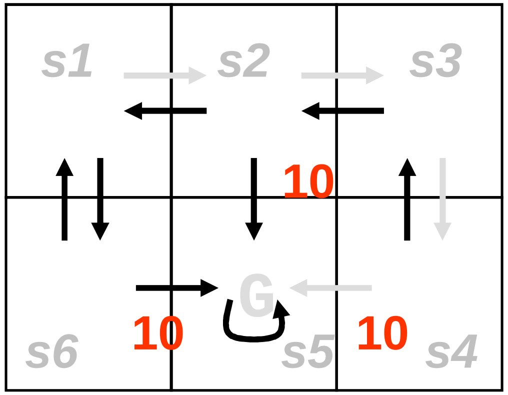


Q(S1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	$r + \gamma$ {Q(s5,loop)-Q(s4,W)}= 10 + 0.8 x 0 – 0 = **10**
0	0	$r + \gamma$ Q(s4,W) = 0 + 0.8 x 10 = **8**	10
0	$r + \gamma \max_{a'}$ {Q(s3,W), Q(s3,S)} = 0 + 0.8 x max{0,8} = **6.4**		
$r + \gamma \max_{a'}$ {Q(s2,W), Q(s2,S)} = 0 + 0.8 x max{6.4, 0} = **5.12**			

Q-Learning Iterations: improving policy

- Start at upper left – move clockwise; $\gamma = 0.8$; $\alpha = 1$

$$\underline{Q}(s, a) \leftarrow r + \gamma \max_{a'} \underline{Q}(s',a')$$



Mossa ε-greedy in s_2 (invece che a = E, scelto a = S):
calcolo $Q(s_2, S) = r + \gamma \max_{a'} \{\underline{Q}(s5,a')\} = 10 + 0.8 \times 0 = \mathbf{10}$

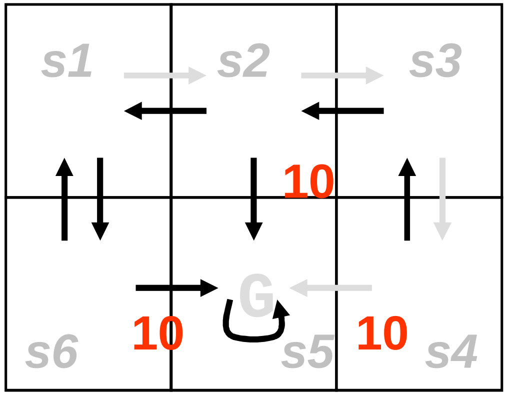
Episodio successivo:
Ricalcolo $Q(s_1, E) = r + \gamma \max_{a'} \{\underline{Q}(s2,E), Q(s2,W), Q(s2, S)\} =$
$r + \gamma \max_{a'} \{6.4, 0.0, 10.0\}$ ➔ **South = $\pi(s_2)$! Policy changed**

Q-Learning Iterations

- Start at upper left – move clockwise; $\gamma = 0.8$

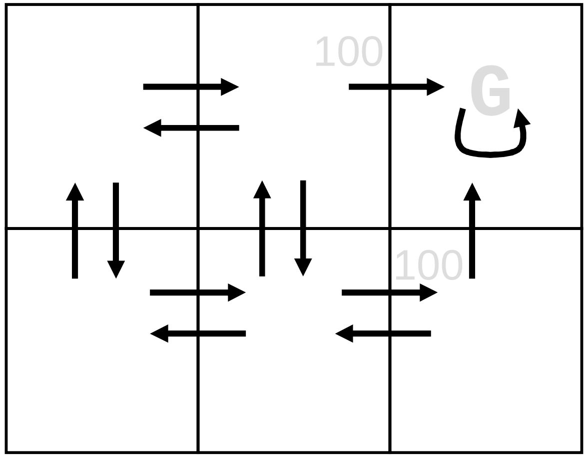
$$\underline{Q}(s, a) \leftarrow r + \gamma \max_{a'} \underline{Q}(s',a')$$

NB in s_2 the new policy drives the agent towards the s_5 state (loop).

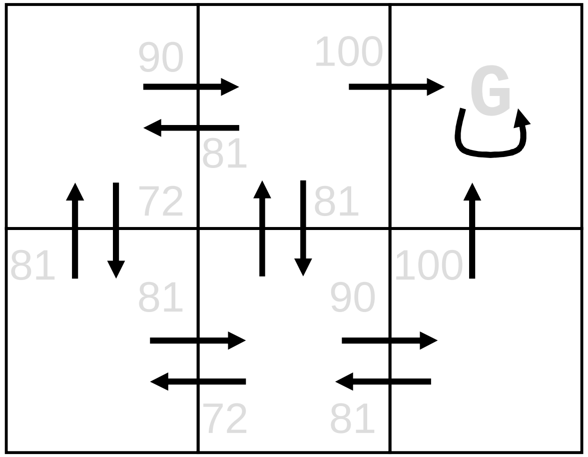


Q(s1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	$r + \gamma \max_{a'} \{\underline{Q}(s5,loop)\} =$ 10 + 0.8 x 0 = 10
0	0	$r + \gamma \max_{a'} \{\underline{Q}(s4,W), \underline{Q}(s4,N)\} = 0 +$ 0.8 x max{10,0} = 8	10
0	$r + \gamma \max_{a'} \{\underline{Q}(s3,W), \underline{Q}(s3,S)\} = 0 +$ 0.8 x max{0,8} = 6.4	8	10
8	6.4	8	10

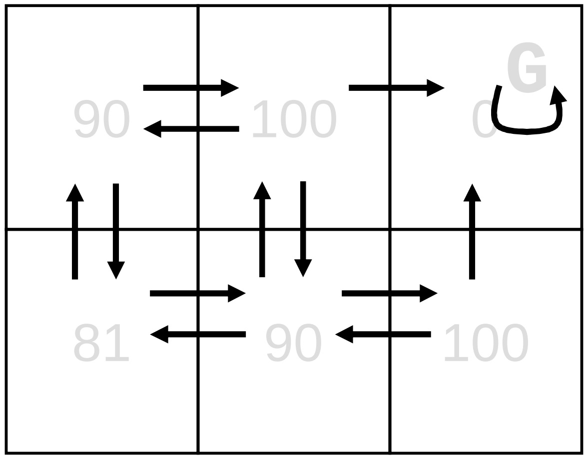
Example 3 - Comparison of functions V and Q (γ = 0.9)



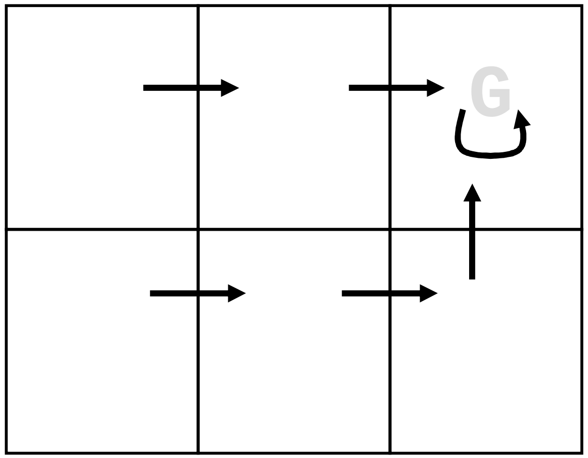
R(*s*, *a*) values



Q(*s*, *a*) values



V*(*s*) values



One Optimal Policy

V*(s) = max (Q*(s,a))

Proprietà del rinforzo

L'ambiente o l'interazione può essere complessa.

Il rinforzo può avvenire solo dopo una più o meno lunga sequenza di azioni (delayed reward).

E.g. agente = giocatore di scacchi.
ambiente = avversario.

Problemi collegati:
- temporal credit assignement.
- structural credit assignement.

L'apprendimento non è più da esempi, ma dall'osservazione del proprio comportamento nell'ambiente.

Esempio SW

- Labirinto

- Gatto & Topo

Sommario

Q-learning