

# Sistemi Intelligenti Reinforcement Learning: Q-learning

Alberto Borghese

Università degli Studi di Milano  
Laboratorio di Sistemi Intelligenti Applicati (AIS-Lab)  
Dipartimento di Scienze dell'Informazione  
[borghese@dsi.unimi.it](mailto:borghese@dsi.unimi.it)



A.A. 2009-2010

1/30

<http://homes.dsi.unimi.it/~borghese/>



## Sommario



SARSA

Q-learning

A.A. 2009-2010

2/30

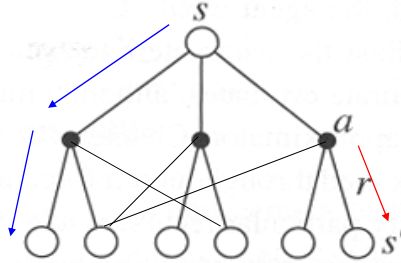
<http://homes.dsi.unimi.it/~borghese/>



## Le value function



$$V^\pi(s) = E_\pi \{ R_t \mid s_t = s \} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right\} = \left[ \sum_{a_j} \pi(a_j, s) \right] \sum_{s'} P_{s \rightarrow s' | a_j} \left[ R_{s \rightarrow s' | a_j} + \gamma V^\pi(s') \right]$$



$$Q^\pi(s, a) = E_\pi \{ R_t \mid s_t = s, a_t = a \} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right\}$$

$$= \sum_{s'} P_{s \rightarrow s' | a_j} \left[ R_{s \rightarrow s' | a_j} + \gamma V^\pi(s') \right]$$

A.A. 2009-2010

3/30

<http://homes.dsi.unimi.it/~borghese/>



## Serve davvero la Value Function?





La Value Function deriva dalla visione della Programmazione Dinamica.

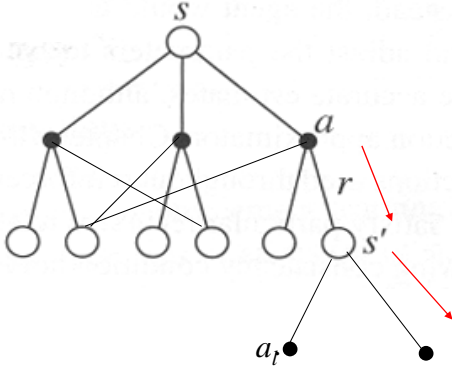
Ma è proprio necessario conoscere la Value function (che riguarda gli stati)? In fondo a noi interessa determinare la Policy.

A.A. 2009-2010

4/30

<http://homes.dsi.unimi.it/~borghese/>





**Calcolo ricorsivo della value function Q**



$$Q^\pi(s, a) = E_\pi \{ R_t \mid s_t = s, a_t = a \} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right\}$$

$$= \sum_{s'} P_{s \rightarrow s' | a} \left[ R_{s \rightarrow s' | a} + \gamma \sum_l \pi(s', a_l) Q^\pi(s', a_l) \right]$$

A.A. 2009-2010 5/30 http://homes.dsi.unimi.it/~borghese\



**Q Functions**

$$\pi^*(s) = \arg \max_a \sum_{s'} P_{s \rightarrow s' | a}^a [R_{s \rightarrow s' | a}^a + \gamma V^\pi(s')] = \arg \max_a Q(s, a)$$

V = Cumulative reward of being in s and choosing a<sub>j</sub>:  $Q^\pi(s, a_j) = \sum_{s'} P_{s \rightarrow s' | a_j} [R_{s \rightarrow s' | a_j} + \gamma V^\pi(s')]$

**Idea chiave:**

- Unire il rinforzo che si ottiene passando da uno stato al successivo in un'unica funzione

$$Q(s, a) = [R_{s \rightarrow s' | a}^a + \gamma V^\pi(s')]$$

- Questa funzione valuta la bontà dell'azione e non più dello stato (a = π(s)).
- A questo punto posso massimizzare Q senza conoscere separatamente il reward istantaneo e la value function come:
 
$$\pi^*(s) = \operatorname{argmax}_a Q(s, a)$$

Q = Cumulative reward of being in s and taking action a.

A.A. 2009-2010 6/30 http://homes.dsi.unimi.it/~borghese\



## Equazioni di ottimalità di Bellman

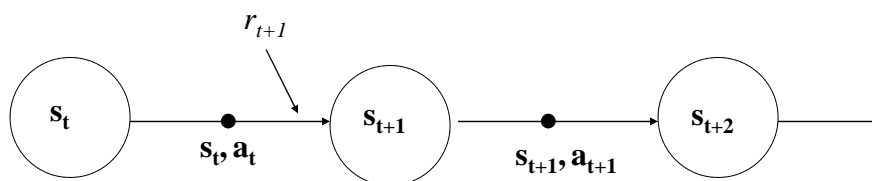
$V^*(s)$  di uno stato, quando viene scelta la policy ottima, deve essere uguale al valore atteso del reward per l'azione migliore per lo stato  $s$ .

$$V^*(s) = \max_{a_j} \sum_{s'} P_{s \rightarrow s' | a_j} [R_{s \rightarrow s' | a_j} + \gamma V^*(s')] ]$$

$$Q^*(s, a_j) = \sum_{s'} P_{s \rightarrow s' | a_j} [R_{s \rightarrow s' | a_j} + \gamma \max_{a'} Q^*(s', a')] ]$$



## Relazione tra $Q_t(\cdot)$ e $Q_{t+1}(\cdot)$ : rappresentazione grafica



$V(s_t)$

$V(s_{t+1})$

One step for Iterative policy Evaluation

---

$Q(s_t, a_t)$

$Q(s_{t+1}, a_{t+1})$

One step for **Q-based** policy Evaluation

---



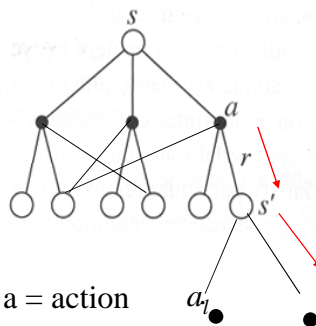
## Come apprendere Q: SARSA



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

1) Apprendiamo il valore di Q per una policy data (*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  
 è di tipo TD(0)



## SARSA Algorithm (progetto)



```

Q(s,a) = rand(); // ∀s, ∀a, eventualmente Q(s,a) = 0
Repeat // for each episode
{
  s = s0;
  Repeat // for each step of the single episode
  {
    a = Policy(s); // ε-greedy??
    s_next = NextState(s,a);
    reward = Reward(s,s_next,a);
    a_next = Policy(s_next); // ε-greedy?
    Q(s,a) = Q(s,a) + α [reward + γ Q(s_next, a_next) - Q(s,a)];
    s = s_next;
  } // until last state
} // until the end of learning
  
```

- 1) Apprendiamo il valore di Q per una policy data (on-policy).
- 2) Dopo avere appreso la funzione Q, possiamo modificare la policy in modo da migliorarla.

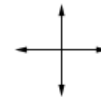
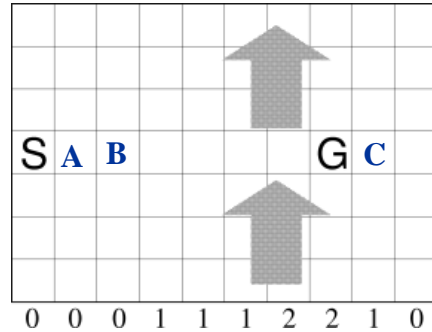
Come integrare i due passi?



## Esempio



From Start to Goal.



standard moves

Upwards wind

$Q(s,a)$  iniziale = 0.  
 $r = 0$  se  $s' = G$ ; altrimenti  $r = -1$ .  
 $\pi(s,a)$  data.

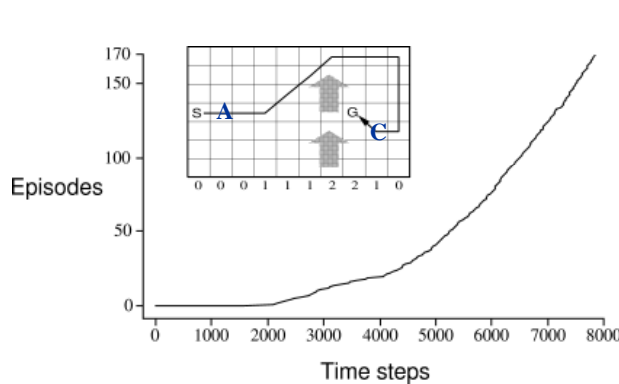
A.A. 2009-2010

11/30

<http://homes.dsi.unimi.it/~borghese/>



## Esempio - risultato



Policy  $\pi$ , greedy or  $\epsilon$ -greedy

$\epsilon = 0.1$

$\alpha = 0.5$

$\gamma = 1$

Per trial or per epoch

Al termine, policy improvement.

Correzione di Q ad un passo:

$$Q(S, \text{east}) = 0 + 0.5 [-1 + 0 - 0] = -0.5$$

$$Q(A, \text{east}) = 0 + 0.5 [-1 + 0 - 0] = -0.5$$

$$Q(C, \text{west}) = 0 + 0.5 [0 + 0 - 0] = 0; \quad (\text{NB } c' \grave{e} \text{ il vento verso l'alto di } 1)$$

A.A. 2009-2010:  $Q(s_t, a_t) = Q(s_t, a_t) + \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$  [ni.it/~borghese/](http://homes.dsi.unimi.it/~borghese/)



# Sommario

SARSA

Q-learning



# Come apprendere Q: SARSA

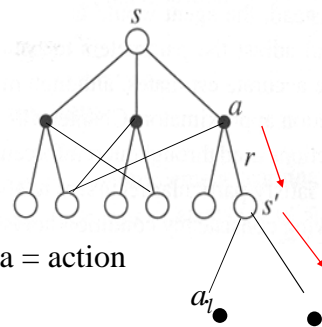
$$Q(s_t, a_t) = Q^\pi(s_t, a_t) + \alpha [r_{t+1} + \gamma Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t)]$$

1) Apprendiamo il valore di Q per una policy data (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.**





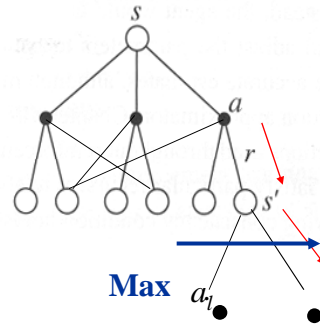
## Off-policy Temporal Difference: Q-learning



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

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.



A.A. 2009-2010

15/30

<http://homes.dsi.unimi.it/~borghese/>



## Q-learning algorithm (progetto)



```
Q(s,a) = rand(); // ∀s, ∀a, Q(s,a) = 0 eventualmente
Repeat // for each episode
{
  s = s0; a = Policy(s); // eventualmente ε-greedy
  Repeat // for each step of the single episode
  {
    s_next = NextState(s,a);
    reward = Reward(s,s_next,a);
    a_next = Policy(s_next); // eventualmente ε-greedy
    Q(s,a) = Q(s,a) + α [reward + γ max Q(s_next, a_next) - Q(s,a)];
    s = s_next;
    UpdatePolicy(a_next, s_next);
    a = a_next;
  } // until last state
} // until the end of learning
```



Max with respect to a\_next

A.A. 2009-2010

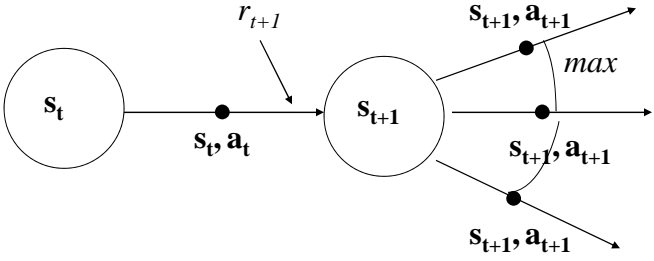
16/30

<http://homes.dsi.unimi.it/~borghese/>



## Rappresentazione grafica





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

**One step for Q Iteration**

---

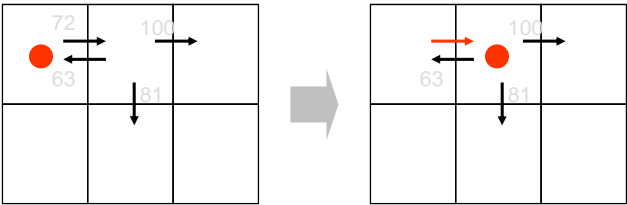
Viene migliorata la policy al tempo t+1.

A.A. 2009-2010
17/30
<http://homes.dsi.unimi.it/~borghese/>

## 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.

A.A. 2009-2010
18/30
<http://homes.dsi.unimi.it/~borghese/>

## Example 1 - Q Learning Update

$\gamma = 0.9$   
 $\alpha = 0.1$

$s_1$  72    100

←    →

63    ↓ 81

→


90

$a_{right}$   $s_2$  100

←    →

63    ↓ 81

0 reward received in the transition



$$Q(s_1, a_{right}) \leftarrow Q(s_1, a_{right}) + \alpha \{ r(s_1, a_{right}, s_2) + \gamma \max_{a'} Q(s_2, a') - Q(s_1, a_{right}) \}$$

$$\leftarrow 72 + \alpha [0 + 0.9 \max \{63, 81, 100\} - Q(s_1, a_{right})]$$

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



Correzione di  $Q(s_1, a_{right})$   
La correzione va a 0 quando  $Q(s_1, a_{right}) = 90$

A.A. 2009-2010

19/30

<http://homes.dsi.unimi.it/~borghese/>

## 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.

$\alpha = 1$

Reward istanteo in rosso e cerchiato

$s_1$      $s_2$      $s_3$

←    ←    ←

↑↑    ↓ (10)    ↓↓

(10)    ←    (10)     $s_4$

$s_6$     (10)     $s_5$

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

E.g. videogioco.

Q(s1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0			

A.A. 2009-2010

20/30

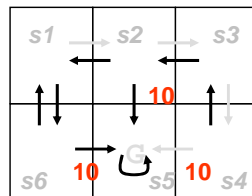
<http://homes.dsi.unimi.it/~borghese/>



## Q-Learning Iterations

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

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



$Q(s1,E)$	$Q(s2,E)$	$Q(s3,S)$	$Q(s4,W)$
0	0	0	

A.A. 2009-2010

21/30

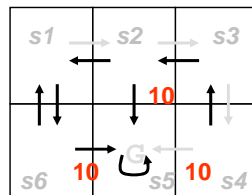
<http://homes.dsi.unimi.it/~borghese/>



## 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	$r + \gamma \max_{a'} \{Q(s5a)\} = 10 + 0.8 \times 0 = 10$

A.A. 2009-2010

22/30

<http://homes.dsi.unimi.it/~borghese/>

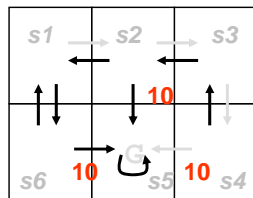


# Q-Learning Iterations



- 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	$r + \gamma \max_{a'} \{Q(s5,a)\} = 10 + 0.8 \times 0 = 10$
0	0	$r + \gamma \max_{a'} \{Q(s4,W), Q(s4,N)\} = 0 + 0.8 \times \max\{10,0\} = 8$	

A.A. 2009-2010

23/30

<http://homes.dsi.unimi.it/~borghese/>

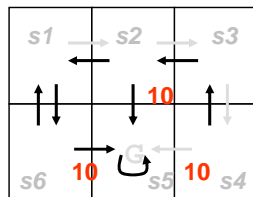


# Q-Learning Iterations



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

A.A. 2009-2010

24/30

<http://homes.dsi.unimi.it/~borghese/>

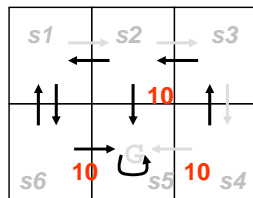


## Q-Learning Iterations: improving policy



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

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



Calcolo  $Q(s_2, S) = r + \gamma \max_{a'} \{Q(s_5, \text{loop})\} = 10 + 0.8 \times 0 = 10$

Ricalcolo  $Q(s_1, E) = r + \gamma \max_{a'} \{Q(s_2, E), Q(s_2, W), Q(s_2, S)\} = r + \gamma \max_{a'} \{6.4, 0.0, 10.0\} \rightarrow \text{South} = \pi(s_2)!$



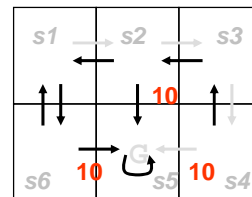
## Q-Learning Iterations



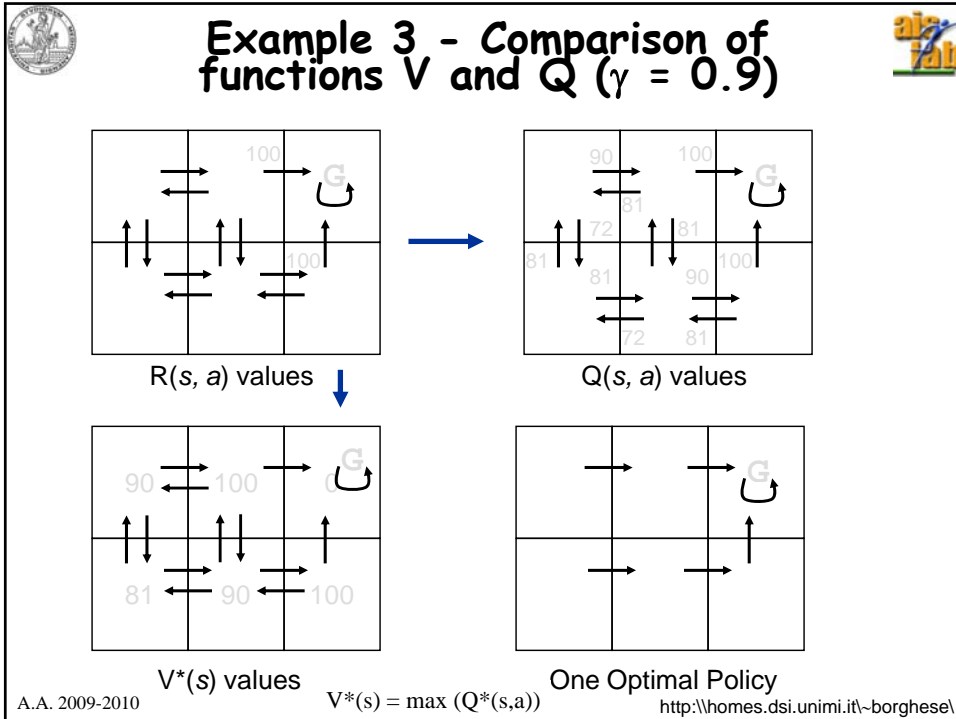
- Start at upper left – move clockwise;  $\gamma = 0.8$

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

NB in B the new policy drives the agent towards the E state (loop).



$Q(s_1, E)$	$Q(s_2, E)$	$Q(s_3, S)$	$Q(s_4, W)$
0	0	0	$r + \gamma \max_{a'} \{Q(s_5, \text{loop})\} = 10 + 0.8 \times 0 = 10$
0	0	$r + \gamma \max_{a'} \{Q(s_4, W), Q(s_4, N)\} = 0 + 0.8 \times \max\{10, 0\} = 8$	10
0	$r + \gamma \max_{a'} \{Q(s_3, W), Q(s_3, S)\} = 0 + 0.8 \times \max\{0, 8\} = 6.4$	8	10
8	6.4	8	10



## 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.

A.A. 2009-2010 28/30 <http://homes.dsi.unimi.it/~borghese/>



## Esempio SW



- Labirinto
- Gatto & Topo



## Sommario



SARSA

Q-learning