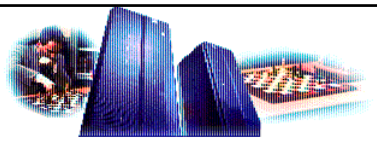
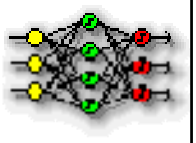




Sistemi Intelligenti
Reinforcement Learning: Q-learning

Alberto Borghese

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

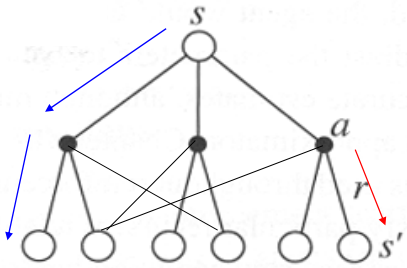
Sommario

SARSA

Q-learning

Le value function

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



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

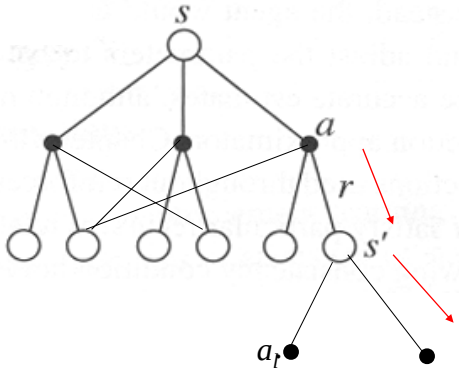
$$= \sum_{s'}P_{s\to s'|a_j}\left[R_{s\to s'|a_j} + \gamma V^{\pi}(s')\right]$$

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.

Calcolo ricorsivo della value function Q



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

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

Q Functions

$$\pi^*(s) = \arg\max_a \sum_{s'} P^a_{s->s'}\left[R^a_{s->s'} + \gamma V^{\pi}(s')\right] = \arg\max_a Q(s,a)$$

V = Cumulative reward of being in *s* and choosing a_j. $Q^{\pi}(s,a_j) = \sum_{s'} P_{s\to s'|a_j}\left[R_{s\to s'|a_j} + \gamma V^{\pi}(s')\right]$

Idea chiave:

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

$$Q(s,a) = \left[R^a_{s->s'} + \gamma V^{\pi}(s')\right]$$

- 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:

$$\pi^*(s) = \text{argmax}_a\ Q(s,a)$$

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

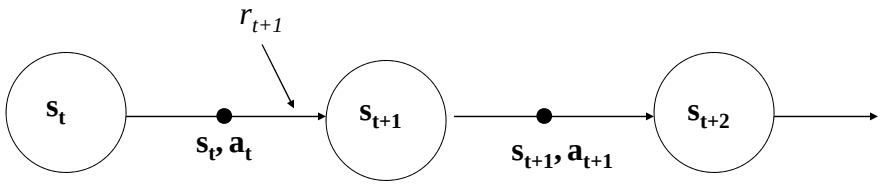
Equazioni di ottimalità di Bellman

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.

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

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

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



$V(s_t)$ $V(s_{t+1})$

One step for Iterative policy Evaluation

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

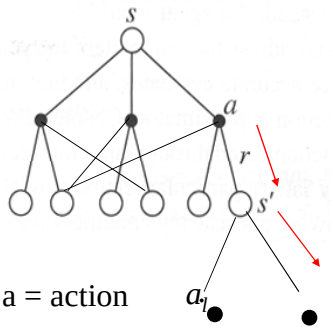
One step for **Q-based** policy Evaluation

Come apprendere Q: SARSA

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

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)

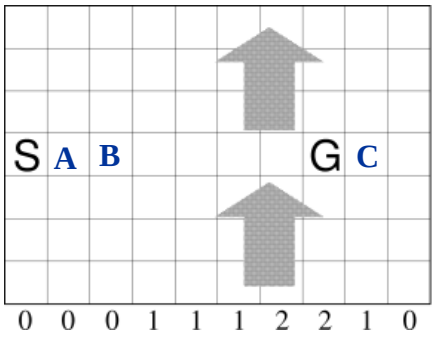
```
Q(s,a) = rand();    // ∀s, ∀a, eventualmente Q(s,a) = 0
Repeat                          // for each episode
{       s = s0;
        Repeat                  // for each step of the single episode
        {
                a = Policy(s);          // ε-greedy??
                s_next = NextState(s,a);
                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.

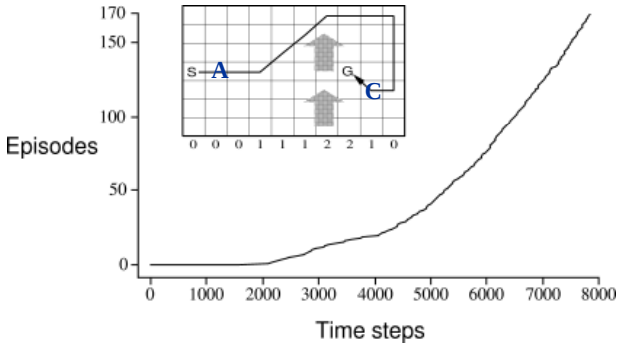


standard moves

Upwards wind

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

Esempio - risultato



Policy π, greedy or ε-greedy

ε = 0.1
α= 0.5
γ = 1

Per trial or per epoch

Al termine, policy improvement.

Correzione di Q ad un passo:
Q(S, east) = 0 + 0.5 [-1 + 0 – 0] = -0.5
Q(A, east) = 0 + 0.5 [-1 + 0 –0] = -0.5
Q(C, west) = 0 + 0.5 [0 + 0 – 0] = 0; (NB c'è il vento verso l'alto di 1)

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

Sommario

SARSA

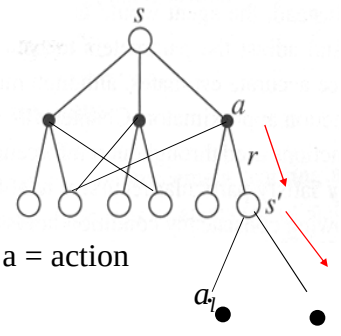
Q-learning

Come apprendere Q: SARSA

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

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

$$V_{k+1}(s) = \left[\sum_{a_j} \pi(a_j, s)\right] \sum_{s'} P_{s \to s'|a_j} \left[R_{s \to s'|a_j} + \gamma V_k(s')\right]$$

Invece di considerare una policy stocastica, consideriamo l'azione migliore:

$$V_{k+1}(s) = \max_a \sum_{s'} P_{s \to s'|a} \left[R_{s \to s'|a} + \gamma V_k(s')\right]$$

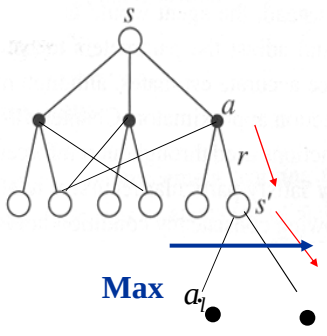
$\forall s$

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.

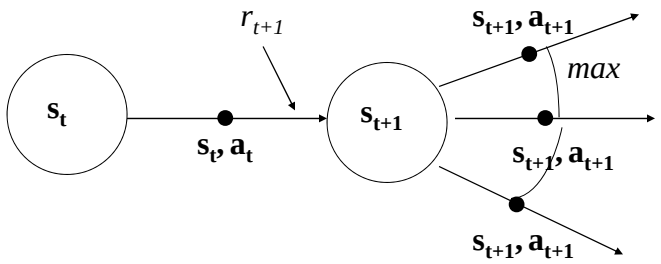


Q-learning algorithm (progetto)

```
Q(s,a) = rand();      // ∀s, ∀a, Q(s,a) = 0 evetualmente
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;             // a = Policy(s)
        }               // until last state
}                       // until the end of learning
```

Max with respect to a_next

Rappresentazione grafica



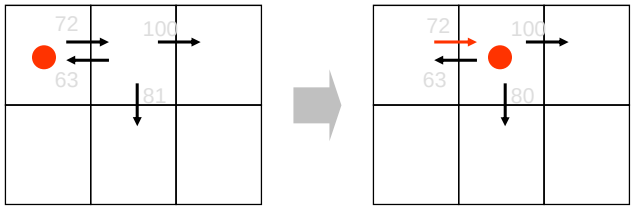
$Q(\mathbf{s_t}, \mathbf{a_t})$ $Q(\mathbf{s_{t+1}}, \mathbf{a_{t+1}})$

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

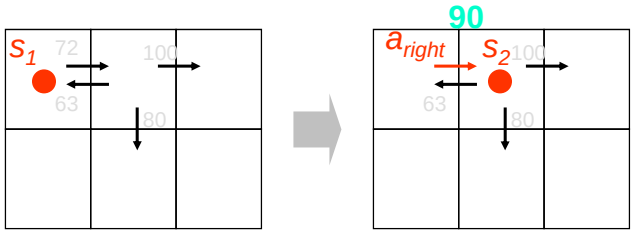
Apprendimento della funzione valore Q. Versione Q-learning. Q(A,dx) = ?

A	B	C
D	E	F

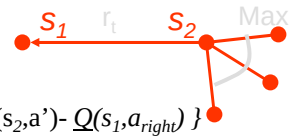
In grigio i valori di Q(s,a).
Nessun reward istantaneo.

Example 1 – *Q* Learning Update

$\gamma = 0.9$
$\alpha = 0.1$
a(s2) = down



0 reward received in the transition



$\underline{Q}(s_1,a_{right}) \leftarrow \underline{Q}(s_1,a_{right}) + \alpha\{ r(s_1,a_{right},, s_2) + \gamma \max_{a'} \underline{Q}(s_2,a') - \underline{Q}(s_1,a_{right}) \}$
$\leftarrow 72 + \alpha [0 + 0.9 \max \{63, 80, 100\} - \underline{Q}(s_1,a_{right})]$
$\leftarrow 72 + \alpha(90 - 72) = 72 + 1.8 = 73.8$

Correzione di $\underline{Q}(s_1,a_{right})$
Correzione dell'azione in s2 da down a right
La correzione va a 0 quando $\underline{Q}(s_1,a_{right}) = 90$

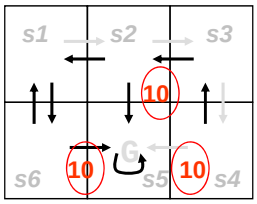
Example 2: Q-Learning Iterations: Episodic

- 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



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

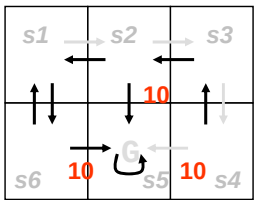
E.g. videogioco.

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

Q-Learning Iterations

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

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

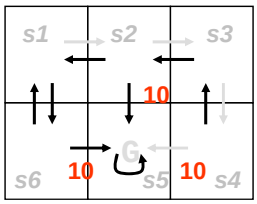


Q(S1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	

Q-Learning Iterations

- Start at upper left – move clockwise; γ = 0.8

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

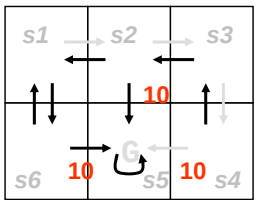


Q(S1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	$r + \gamma \max_{a'} \{\underline{Q}(s5a)\}$= 10 + 0.8 x 0 = **10**

Q-Learning Iterations

- Start at upper left – move clockwise; γ = 0.8

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

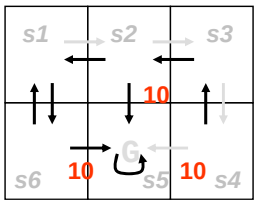


Q(S1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	$r + \gamma \max_{a'} \{\underline{Q}(s5,a)\}$= 10 + 0.8 x 0 = 10
0	0	$r + \gamma \max_{a'} \{\underline{Q}(s4,W), \underline{Q}(s4,N)\}$ = 0 + 0.8 x max{10,0) = 8	

Q-Learning Iterations

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

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

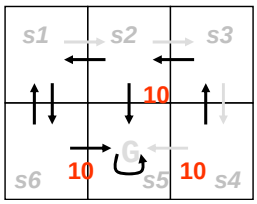


Q(S1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	$r + \gamma \max_{a'} \{\underline{Q}(s5,loop)\}$= 10 + 0.8 x 0 = 10
0	0	$r + \gamma \max_{a'} \{\underline{Q}(s4,W), \underline{Q}(s4,N)\}$ = 0 + 0.8 x max{10,0) = 8	10
0	$r + \gamma \max_{a'} \{\underline{Q}(s3,W), \underline{Q}(s3,S)\}$ = 0 + 0.8 x max{0,8) =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

Q-Learning Iterations: improving policy

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

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



Calcolo $Q(s_2, S) = r + \gamma \max_{a'} \{\underline{Q}(s5,loop)\} = 10 + 0.8 \times 0 = \mathbf{10}$

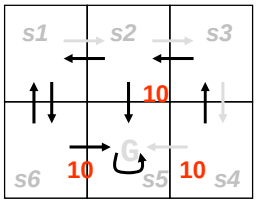
Ricalcolo $Q(s_1, E) = r + \gamma \max_{a'} \{\underline{Q}(s2,E), Q(s2,W), Q(s2, S)\} =$
$r + \gamma \max_{a'} \{6.4, 0.0, 10.0\}$ ➔ **South = $\pi(s_2)$!**

Q-Learning Iterations

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

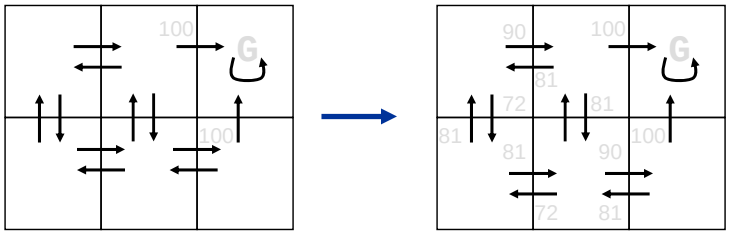
$$\underline{Q}(s, a) \leftarrow r + \gamma \max_{a'} \underline{Q}(s', a')$$

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



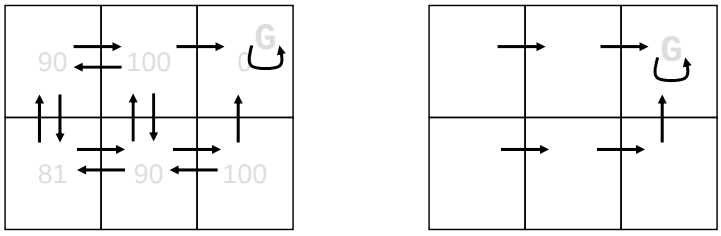
Q(s1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	$r + \gamma \max_{a'} \{\underline{Q}(s5,loop)\} =$ 10 + 0.8 x 0 = 10
0	0	$r + \gamma \max_{a'} \{\underline{Q}(s4,W), \underline{Q}(s4,N)\} = 0 +$ 0.8 x max{10,0) = 8	10
0	$r + \gamma \max_{a'} \{\underline{Q}(s3,W), \underline{Q}(s3,S)\} = 0 +$ 0.8 x max{0,8) =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

Example 3 - Comparison of functions V and Q ($\gamma = 0.9$)



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

SARSA

Q-learning