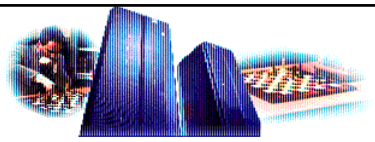
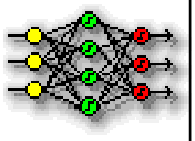




Sistemi Intelligenti
Reinforcement Learning: Eligibility Trace

Alberto Borghese

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

Sommario

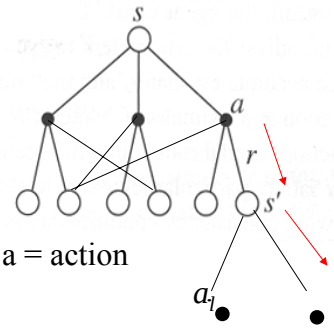
- Q-learning
- The eligibility trace
- Q(λ), SARSA(λ)

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.

Calcolo ricorsivo della Value function

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

$$V^{\pi}(s') = E_{\pi}\{R_{t+1} \mid s_{t+1} = s'\}$$

Legame?

Policy Next-state

$$P_{s->s'|a} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$$

$$V^{\pi}(s) = \sum_{a_j}\pi(a_j,s)\sum_{s'}P_{s\to s'|a_j}R_{s\to s'|a_j} + E_{\pi}\left\{\gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \,\middle|\, s_t = s\right\}$$

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

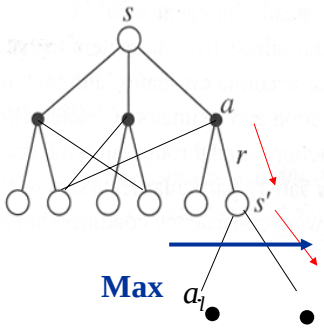
Bellman's equation

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.

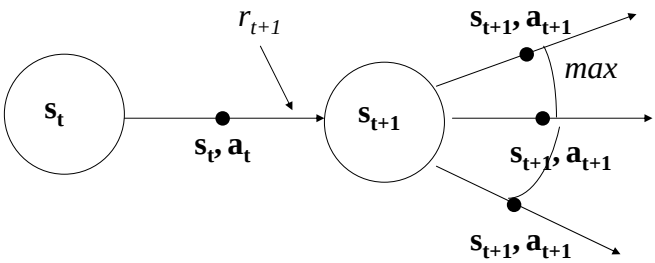


Q-learning algorithm (progetto)

```
Q(s,a) = rand();      // ∀s, ∀a, Q(s,a) = 0 evetualmente
Repeat                        // for each episode
{        s = s0;
         Repeat               // for each step of the single episode
         {
                  a = Policy(s);          // eventualmente ε-greedy
                  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);
         }                    // until last state
}                             // until the end of learning
```

Max with respect to a_next

Rappresentazione grafica



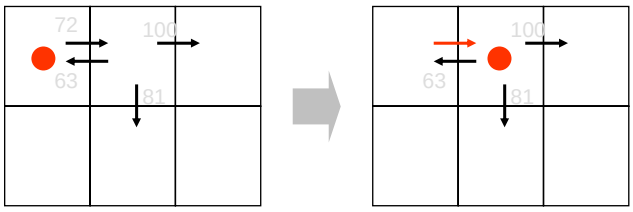
$Q(s_t, a_t)$ $Q(s_{t+1}, a_{t+1})$

One step for **Q Iteration**

Viene migliorata la policy al tempo t+1.

Example 1 – *Q* Learning Update

$\gamma = 0.9$

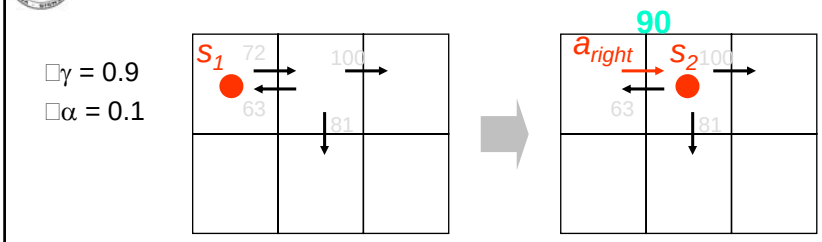


0 reward received

Esempio tratto dai lucidi del corso di Brian C. Williams su RL.
Modificati dalle slide di: Manuela Veloso, Reid Simmons, & Tom Mitchell, CMU

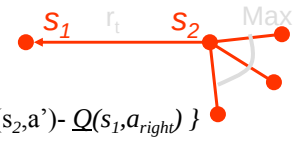
Apprendimento della funzione valore Q. Versione Q-learning.

Example 1 – *Q* Learning Update



□γ = 0.9
□α = 0.1

0 reward received in the transition



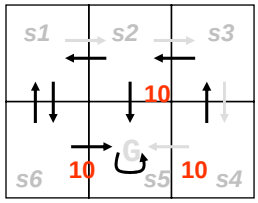
$\underline{Q}(s_1,a_{right}) \leftarrow \underline{Q}(s_1,a_{right}) + \alpha\{ r(s_1,a_{right}, s_2) + \gamma \max_{a'} \underline{Q}(s_2,a') - \underline{Q}(s_1,a_{right}) \}$
$\leftarrow 72 + \alpha [0 + 0.9 \max \{63, 81, 100\} - \underline{Q}(s_1,a_{right})]$
$\leftarrow 72 + \alpha(90 - 72) = 1.8$

In grigio i valori di Q(s,a). Nessun reward istantaneo.
Correzione di $\underline{Q}(s_1,a_{right})$
La correzione va a 0 quando $\underline{Q}(s_1,a_{right}) = 90$

Example 2: Q-Learning Iterations: Episodic

- Start at upper left; Selected policy: move clockwise; Table initially 0; γ = 0.8.
  Possibili transizione sono segnate con frecce nere e grigie.

Reward istanteo in rosso



α = 1

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

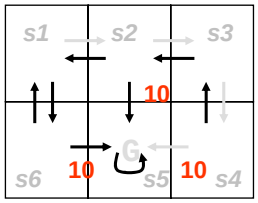
E.g. videogioco.

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; $\gamma = 0.8$

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

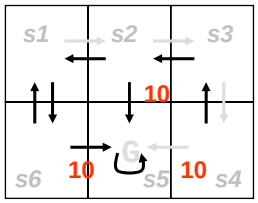


Q(S1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	

Q-Learning Iterations

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

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

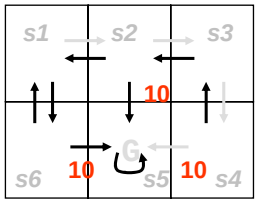


Q(S1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	$r + \gamma \max_{a'} \{\underline{Q}(s5a)\} =$ $10 + 0.8 \times 0 = \mathbf{10}$

Q-Learning Iterations

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

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

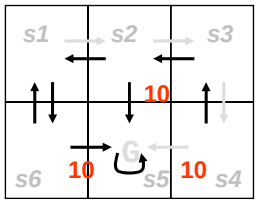


Q(S1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	$r + \gamma \max_{a'} \{\underline{Q}(s5,a)\} =$ 10 + 0.8 x 0 = 10
0	0	$r + \gamma \max_{a'} \{\underline{Q}(s4,W), \underline{Q}(s4,N)\} = 0 +$ 0.8 x max{10,0) = 8	

Q-Learning Iterations

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

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

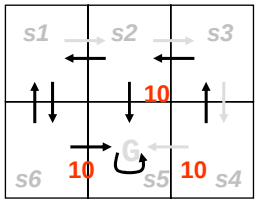


Q(S1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	$r + \gamma \max_{a'} \{\underline{Q}(s5,loop)\} =$ 10 + 0.8 x 0 = 10
0	0	$r + \gamma \max_{a'} \{\underline{Q}(s4,W), \underline{Q}(s4,N)\} = 0 +$ 0.8 x max{10,0) = 8	10
0	$r + \gamma \max_{a'} \{\underline{Q}(s3,W), \underline{Q}(s3,S)\} = 0 +$ 0.8 x max{0,8) =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

Q-Learning Iterations

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

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

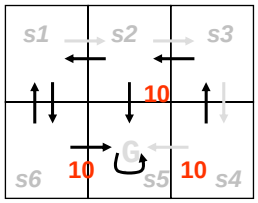


Q(s1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	$r + \gamma \max_{a'} \{\underline{Q}(s5,loop)\} = 10 + 0.8 \times 0 = 10$
0	0	$r + \gamma \max_{a'} \{\underline{Q}(s4,W), \underline{Q}(s4,N)\} = 0 + 0.8 \times \max\{10,0\} = 8$	10
0	$r + \gamma \max_{a'} \{\underline{Q}(s3,W), \underline{Q}(s3,S)\} = 0 + 0.8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

Q-Learning Iterations: improving policy

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

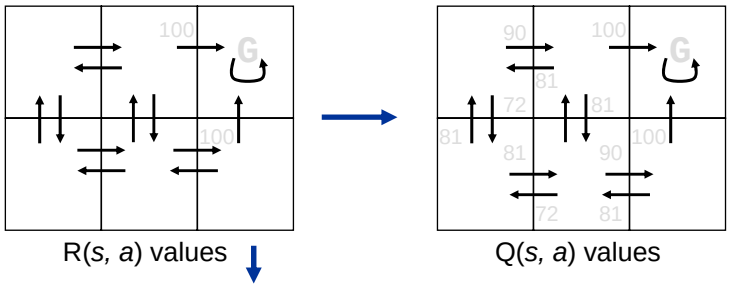
$$\underline{Q}(s, a) \leftarrow r + \gamma \max_{a'} \underline{Q}(s', a')$$



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

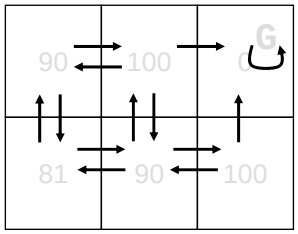
Ricalcolo $Q(s_1, E) = r + \gamma \max_{a'} \{\underline{Q}(s2,E), Q(s2,W), Q(s2, S)\} =$
$r + \gamma \max_{a'} \{6.4, 0.0, 10.0\}$ ➔ **South = $\pi(s_2)$!**

Example 3
Comparison functions V and Q

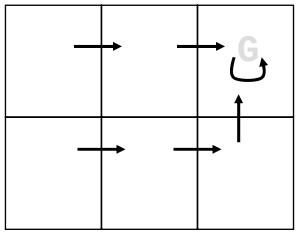


R(*s*, *a*) values

Q(*s*, *a*) values



V*(*s*) values



One Optimal Policy

Sommario

- Q-learning
- The eligibility trace
- Q(λ), SARSA(λ)

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.

Formulazione di TD(0) per Q-learning

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

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

$$V_{k+1}(s_t)=V_k(s_t)+\alpha\left[r_{t+1}+\gamma V_k(s_{t+1})-V_k(s_t)\right]$$

$$\Delta V(s_t)=+\alpha\,\delta_k \qquad \delta_k=\left[r_{t+1}+\gamma V_k(s_{t+1})-V_k(s_t)\right]$$

Cosa rappresenta la Eligibility trace

Buffer di memoria: contiene traccia di eventi passati (stati visitati, azioni…); la traccia evapora nel tempo.

Quando viene calcolato un errore usando metodi basati su TD, la eligibility trace suggerisce quali variabili aggiornare (credit assignement).

Amplia l'orizzonte temporale sul quale fare l'aggiornamento a più di 1 passo.

View of the idea – forward view

Considero il reward su orizzonti temporali più ampi:

$R_t = r_{t+1} + \gamma\, Q(s_{t+1}, a_{t+1})$
$R_t = r_{t+1} + \gamma\, r_{t+2} + \gamma^2\, V(s_{t+2}, a_{t+2})$
$R_t = r_{t+1} + \gamma\, r_{t+2} + \gamma^2\, r_{t+3} + \gamma^3\, V(s_{t+3}, a_{t+3})$
……..

$$R_t = \frac{1}{M+1}\sum_{k=0}^{M}\gamma^k\left[r_{t+1+k} + \gamma Q(s_{t+1+k}, a_{t+1+k})\right]$$

Aggiornamento della value function: $\Delta Q(s,a) = \alpha\, \delta_t$

$$\delta_t = R_t - Q_t(s_t, a_t) = \left(\frac{1}{M+1}\sum_{k=0}^{M}\gamma^k\left[r_{t+1+k} + \gamma Q(s_{t+1+k}, a_{t+1+k})\right] - Q(s_t, a_t)\right)$$

Come arrivare all'elegibility trace

Considero il reward su orizzonti temporali più ampi:

$R_t = r_{t+1} + \gamma\, Q(s_{t+1}, a_{t+1})$
$R_t = r_{t+1} + \gamma\, r_{t+2} + \gamma^2\, V(s_{t+2}, a_{t+2})$
$R_t = r_{t+1} + \gamma\, r_{t+2} + \gamma^2\, r_{t+3} + \gamma^3\, V(s_{t+3}, a_{t+3})$
……..

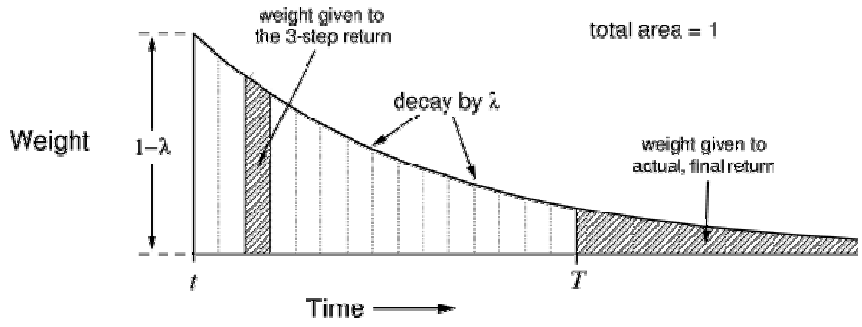
Ne faccio una media pesando di più, mediante λ, i reward nell'immediato futuro:

$R^{\lambda}_t = K * (R_t^{(1)} + \lambda\, R_t^{(2)} + \lambda^2\, R_t^{(3)} + \ldots..)$

$R^{\lambda}_t = (1 - \lambda) * \sum_{k=1}^{+\infty} \lambda^{k-1} R_t^{(k)}$ Dipende da λ e da γ!

La somma dei pesi deve essere = 1

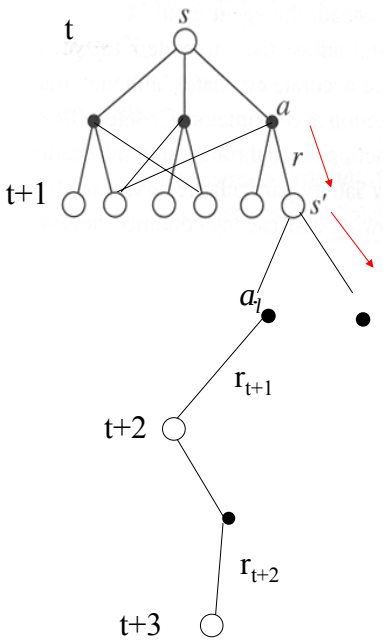
Visualizzazione grafica



$\lambda = 0$ TD(0) $\Delta V_t = \alpha\, [R_t^{\lambda} - V_t(s_t)]$

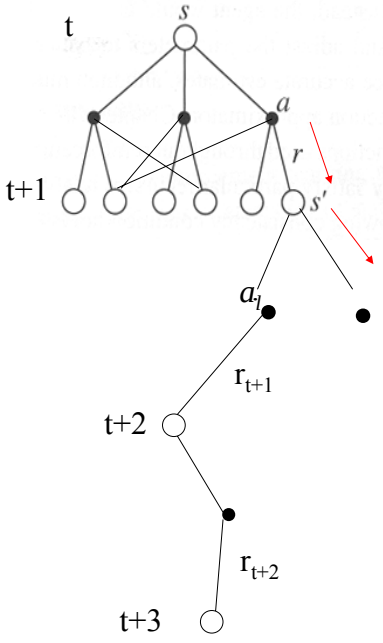
λ regola la velocità di decremento del peso del reward.

Problemi: TD(λ) in questa forma è non causale.



Rewards

Look forwards 3 steps

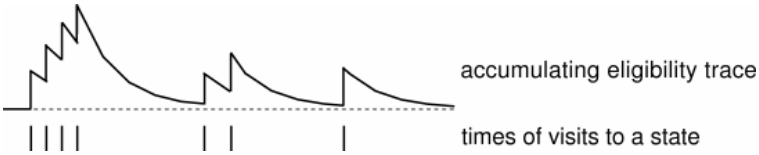


Rewards

Look backwards 3 steps


Eligibility trace

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & \text{if } s = s_t \text{ and } a = a_t; \\ \gamma\lambda e_{t-1}(s,a) & \text{otherwise.} \end{cases} \qquad \text{for all } s, a$$




Come utilizzare la eligibility trace

TD(0) Learning:

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

Errore: δ_t

$$Q_{k+1}(s_t,a_t) = Q_k(s_t,a_t) + \alpha\delta_t$$ Per 1 coppia (s,a)

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha\delta_t e_t(s,a)$$ Per tutte le coppie (s,a)

Eleggibilità: $e_t(s,a)$

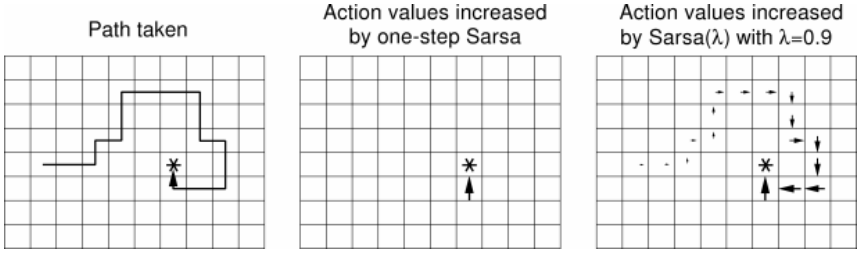
Sommario

- Q-learning
- The eligibility trace
- Q(λ), SARSA(λ)

Algoritmo

Initialize $Q(s,a)$ arbitrarily and $e(s,a) = 0$, for all s, a
Repeat (for each episode):
  Initialize s, a
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
    $e(s, a) \leftarrow e(s, a) + 1$
    For all s, a:
      $Q(s, a) \leftarrow Q(s, a) + \alpha \delta e(s, a)$
      $e(s, a) \leftarrow \gamma \lambda e(s, a)$
    $s \leftarrow s'$; $a \leftarrow a'$
  until s is terminal

Esempio



Con il semplice costo di una variabile per ogni coppia stato-azione, ho un aggiornamento graduale della funzione valore di più stati.

Q(s,a) inizializzati ad un valore leggermente negativo.
r = 0 per ogni stato prossimo, tranne lo stato finale, per il quale r = +1.

Watkin's Q(λ)

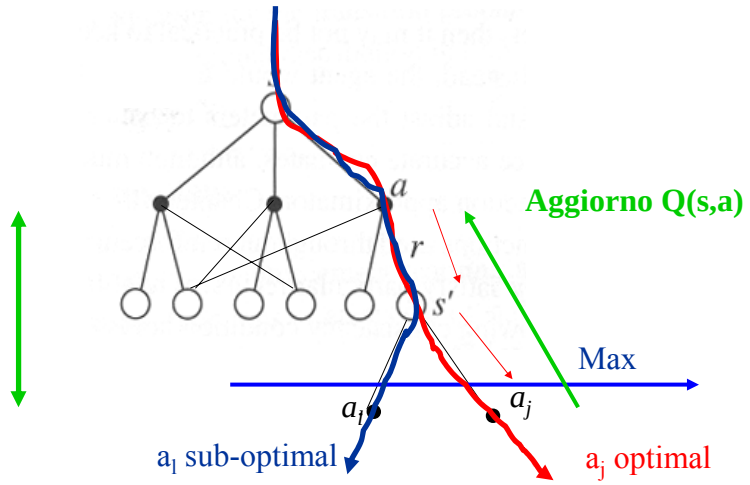
$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\right]$$

Quanto posso guardare in avanti (look-ahead)?

Suppongo di scegliere a' = a_{t+1} azione esplorativa.

Posso sempre calcolare $Q(s_t, a_t)$, scegliendo il max($Q(s_{t+1}, a_{t+1})$). Questo vuole dire ipotizzare di scegliere a_{max} = argmax(max(Q(st+1, at+1)), che in questo caso: $a_{max} \neq a'$.
Ma poi devo ripartire da capo.

Analisi grafica delle mosse ε-eplorative



Q-learning

$$e_t(s,a) = \mathcal{I}_{ss_t} \cdot \mathcal{I}_{aa_t} + \begin{cases} \gamma\lambda e_{t-1}(s,a) & \text{if } Q_{t-1}(s_t,a_t) = \max_a Q_{t-1}(s_t,a); \\ 0 & \text{otherwise,} \end{cases}$$

Aggiorno Q:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\,\delta_t\, e_t(s,a),$$

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1},a') - Q_t(s_t,a_t).$$

Scelta di a:

Se scelgo a_{max}, continuo come SARSA, altrimenti e(s,a) = 0.

Algoritmo

Initialize $Q(s,a)$ arbitrarily and $e(s,a) = 0$, for all s, a
Repeat (for each episode):
  Initialize s, a
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $a^* \leftarrow \arg\max_b Q(s', b)$ (if a' ties for the max, then $a^* \leftarrow a'$)
    $\delta \leftarrow r + \gamma Q(s', a^*) - Q(s, a)$
    $e(s,a) \leftarrow e(s,a) + 1$
    For all s, a:
      $Q(s,a) \leftarrow Q(s,a) + \alpha \delta e(s,a)$
      If $a' = a^*$, then $e(s,a) \leftarrow \gamma \lambda e(s,a)$
                  else $e(s,a) \leftarrow 0$
    $s \leftarrow s'$; $a \leftarrow a'$
  until s is terminal

Sommario

- Q-learning
- The eligibility trace
- Q(λ), SARSA(λ)