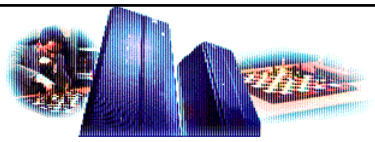
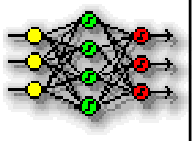




Sistemi Intelligenti
Reinforcement Learning: Temporal Difference

Alberto Borghese

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

Sommario

Temporal differences

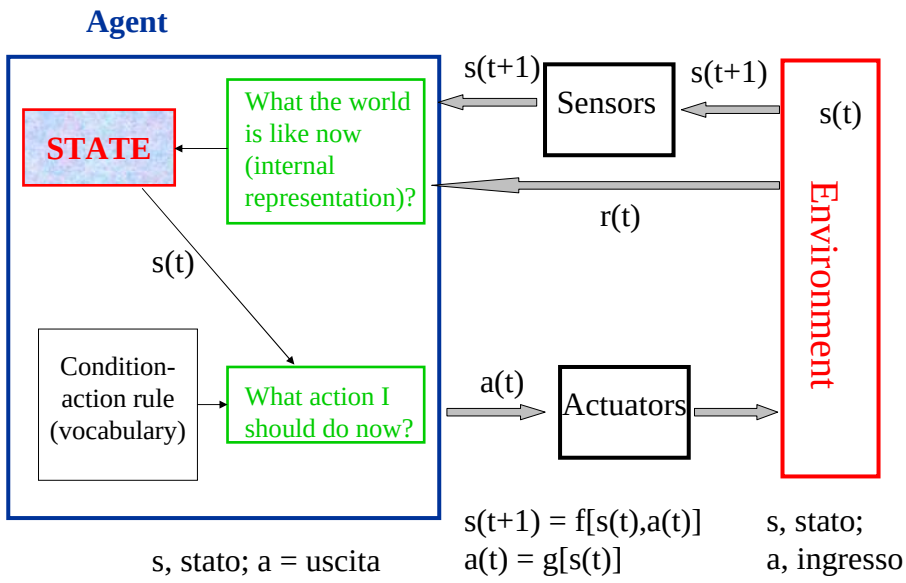
SARSA

1° annual RL competition @ NIPS 2006

Results at:

http://rlai.cs.ualberta.ca/RLAI/rlc.html

Schematic diagram of an agent



How About Learning the Value Function?

Facciamo imparare all'agente la value function, per una certa politica: V^{π}:

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

È una funzione dello stato.

Una volta imparata la value function, V^{*}, l'agente seleziona la policy ottima passo per passo, "one step lookahead":

$$\pi^{*}(s) = \arg\max_{a} \sum_{s'} P^{a}_{s \to s'} \left[R^{a}_{s \to s'} + \gamma V^{\pi}(s')\right]$$

Full backup, for all states

Value function iteration

Facciamo imparare all'agente la value function, per una certa politica: V^{π}, analizzando quello che succede in uno step temporale:

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

L'apprendimento della policy si può inglobare nella value iteration:

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

Full backup, for all states

Asynchronous DP

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

Full backup, single state, s, all future states s'

Fino a questo punto, è noto un modello dell'ambiente:
•R(.)
•P(.)

Environment modeling -> Value function computation -> Policy optimization.

Alcuni richiami: DP update

Iterazione tra:
• Calcolo della Value function

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

•Miglioramento della policy

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

Non sono noti

Background su Temporal Difference (TD) Learning

Al tempo t abbiamo a disposizione:

$r_{t+1} = r'$ $\quad R_{s \to s'|a_j}$

$s_{t+1} = s'$ $\quad P_{s \to s'|a_j}$

Reward certo
Transizione certa
vengono misurati dall'ambiente

Come si possono utilizzare per apprendere?

TD(0) update

Ad ogni istante di tempo di ogni trial aggiorno la Value function:

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

Da confrontare con la iterative policy evaluation:

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

E con il valore di uno stato sotto la policy π(s,a):

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\{r_{t+1} + \gamma V^{\pi}(s') \mid s_t = s\}$$

Quanto vale α?

Confronto con il setting associativo

$$Q_{k+1} = Q_k - \frac{Q_k}{N_{k+1}} + \frac{r_{k+1}}{N_{k+1}} = \boxed{Q_k + \alpha[r_{k+1} - Q_k]}$$

Occupazione di memoria minima: Solo Q_k e k.
NB k è il numero di volte in cui è stata scelta a_j.

Questa forma è la base del RL. La sua forma generale è:

NewEstimate = OldEstimate + StepSize [Target – OldEstimate]
*NewEstimate = OldEstimate + StepSize * Error.*

StepSize = α = 1/k *a = cost*

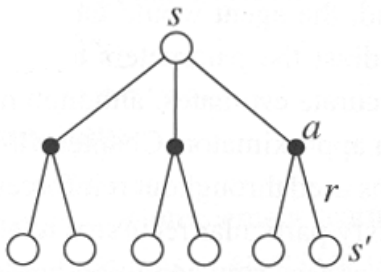
Qual è la differenza introdotta dall'approccio DP?

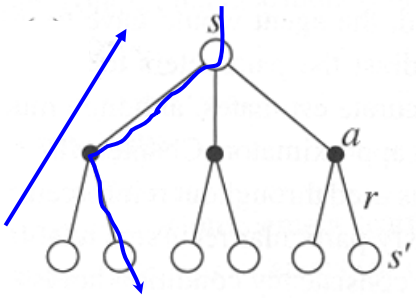
Setting α value

$\alpha(s_t, a_t, s_{t+1}) = 1/k(s_t, a_t, s_{t+1})$, where k represents the number of occurrences of s_t, a_t, s_{t+1}. With this setting the estimated Q tends to the expected value of Q(s,a).

Per semplicità si assume solitamente $\alpha < 1$ costante. In questo caso, Q(s,a) assume il valore di una media pesata dei reward a lungo termine collezionati da (s,a), con peso: $(1-\alpha)^k$: *exponential recency-weighted average*.

Sample backup





Full backup

Single sample is evaluated

Algoritmo per TD(0) – Progetto per esame (da completare con scelta della policy)

Inizializziamo V(s) = 0.
Inizializziamo la policy: π(s,a).da valutare
Repeat
{ s = s0;
Repeat // For each state until terminal state, analyze an episode
{ a = π(s);
s_next = NextState(s, a);
reward = Reward(s, s_next, a);
V(s) = V(s) + α [reward + γ V(s_next) – V(s)];
s = s_next;
} until TerminalState
} Until convergence of V(s) for policy π(s,a)

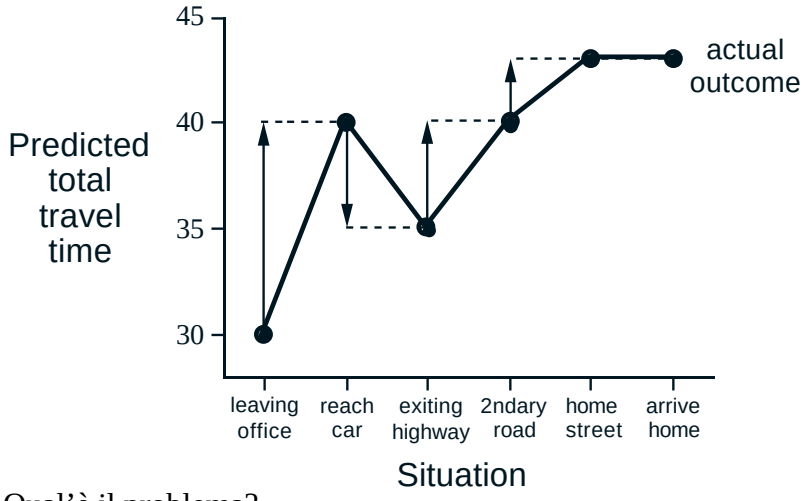
Esempio: valutazione della policy mediante TD

Obbiettivo: predirre la durata del percorso per tornare a casa.

Stato	Tempo trascorso	Tempo previsto	Tempo totale
Esco dall'ufficio	0	30	30
Salgo in auto (nevica)	5	35	40
Esco dall'autostrada	20	15	35
Strada secondaria (camion davanti!)	30	10	40
Strada di casa	40	3	43
Entro in casa	43	0	43

V(s) è l'expected "*Time-to-go*" α = cost.

Modifiche di V(s) run-time



Qual'è il problema?

Alcuni passi di iterazione per TD(0)

$V(0) = V(0) + \alpha\,(r_1 + \gamma V(1) - V(0)) = 30 + \alpha\,(5 + 35 - 30) = 30 + \alpha * \Delta$

Stima iniziale del tempo di percorrenza totale: 30m
Tempo di percorrenza fino all'auto: 5m
Stima del tempo di percorrenza dal parcheggio: 35m

$V(1) = V(1) + \alpha\,(r_1 + \gamma V(2) - V(1)) = 35 + \alpha\,(20 + 15 - 35) = 35 + \alpha * \Delta$

Stima iniziale del tempo di percorrenza dal parcheggio: 35m
Tempo di percorrenza fino ad uscita autostrada: 20m
Stima del tempo di percorrenza dall'uscita autostrada: 15m

Alcuni passi di iterazioni per TD(0) partendo da V(s) = 0

$V_2(0) = V_1(0) + \alpha\,(r_1 + \gamma V_1(1) - V_1(0)) = 0 + \alpha\,(5 + 0 - 0) = 0 + \alpha * 5$

Stima iniziale del tempo di percorrenza totale: $(\alpha * 5)$ m
Tempo di percorrenza fino all'auto: 5m
Stima del tempo di percorrenza dal parcheggio: 0m

$V_2(1) = V_1(1) + \alpha\,(r_2 + \gamma V_2(2) - V_2(1)) = \alpha*5 + \alpha\,(20+\alpha*20-\alpha*5) =$
$= 25\alpha + 15\alpha^2$ (e.g. per $\alpha = 0.6$, $V_2(1) = 15 + 5.4 = 20.4$m)

$V_3(0) = V_2(0) + \alpha\,(r_1 + \gamma V(2) - V(1)) = 5\alpha + \alpha(5\alpha + 25\alpha + 15\alpha^2 - 5\alpha) =$
$= 5\,\alpha + 25\alpha^2 + 15\alpha^3$ (e.g. per $\alpha = 0.6$, $V_3(0) = 3 + 9 + 3.24 =$
15.24m

Proprietà del metodo TD

Non richiede conoscenze a priori dell'ambiente.
L'agente stima dalle sue stesse stime precedenti (bootstrap).
Si dimostra che il metodo converge asintoticamente.

Batch vs trial learning.

Converge!!

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

Single backup, single state, s_t, single future state s_{t+1}

Rimpiazza iterative Policy evaluation.
Rimane il passo di Policy iteration (improvement).

Sommario

Temporal differences

SARSA

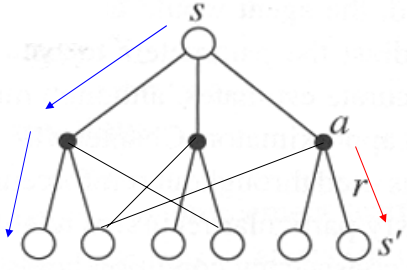
Serve davvero la Value Function?

La Value Function deriva dalla visione della Programmazione Dinamica.

Ma è proprio necessario conoscere la Value function? In fondo a noi interessa determinare la Policy.

Le value function

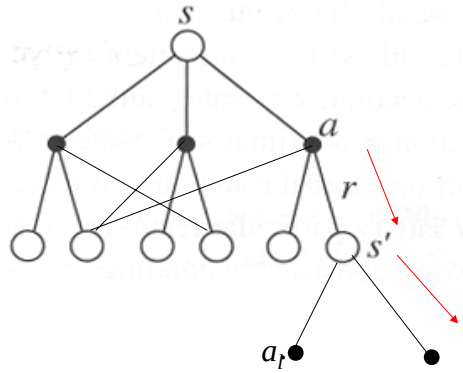
$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \,\middle|\, s_t = s\right\} = \left[\sum_{a_j}\pi(a_j, s)\right]\sum_{s'}P_{s\to s'|a_j}\left[R_{s\to s'|a_j} + \gamma V^{\pi}(s')\right]$$



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

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

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*.

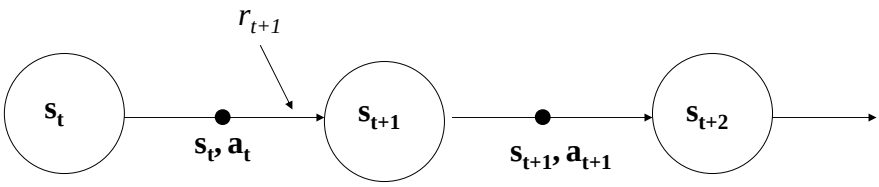
Equazioni di ottimalità di Bellman

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

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

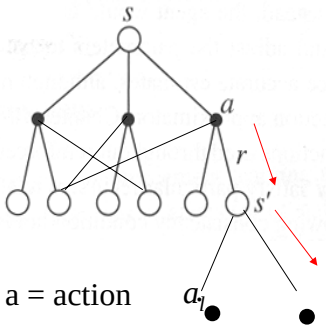
One step for **Q-based** policy Evaluation

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

SARSA Algorithm (progetto)

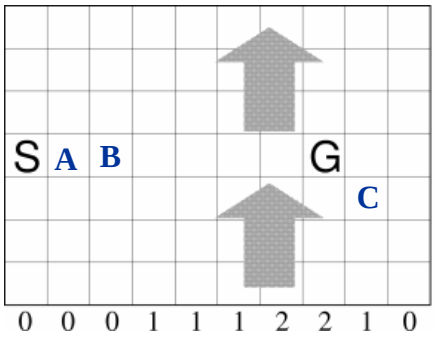
```
Q(s,a) = rand();    // ∀s, ∀a, eventualmente Q(s,a) = 0
Repeat                          // for each episode
{       s = s0;
        Repeat                  // for each step of the single episode
        {
                a = Policy(s);          // ε-greedy??
                s_next = NextState(s,a);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

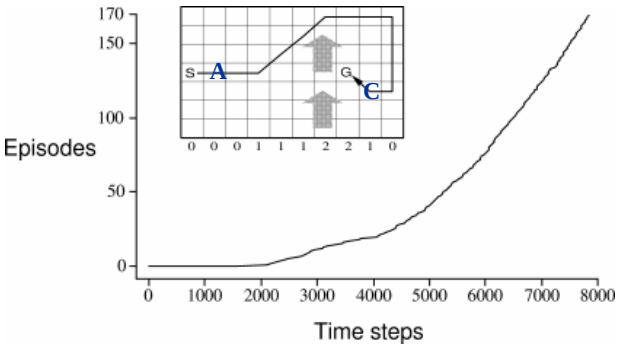


standard moves

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

Per trial
or
Per epoch

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 –(-0.5)] = -0.5
Q(C, west) = 0 + 0.5 [0 + 0 – 0] = 0;

Al termine, policy improvement.

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

Sommario

Temporal differences

SARSA