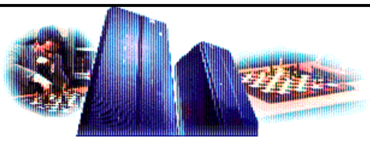
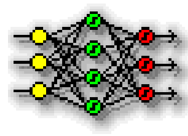




Sistemi Intelligenti Reinforcement Learning: Value Iteration and Temporal Differences

Alberto Borghese

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

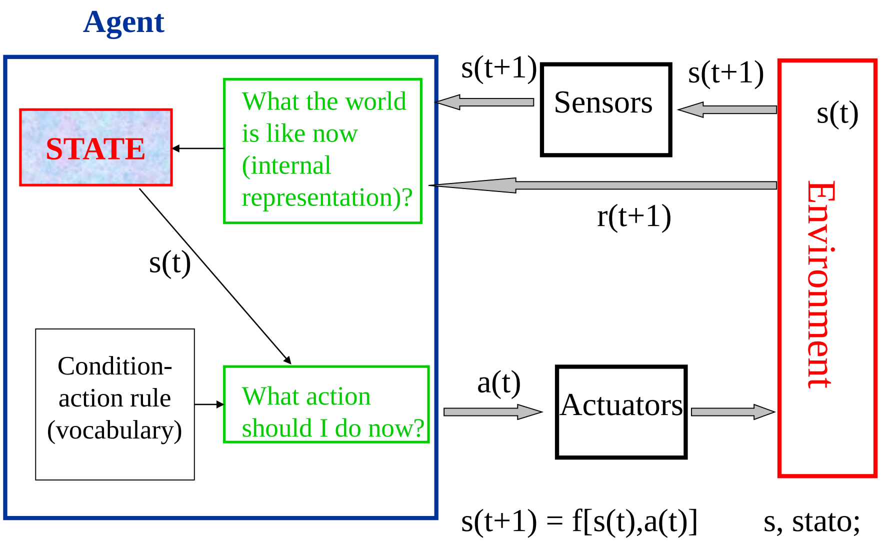
Sommario

Value iteration

Esempi

Learning with temporal differences

Schematic diagram of an agent



s(t+1) = f[s(t),a(t)]
a(t) = g[s(t)]

s, stato; a = uscita

s, stato;
a, ingresso

Calcolo ricorsivo della Value function ottima::confronti

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

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

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

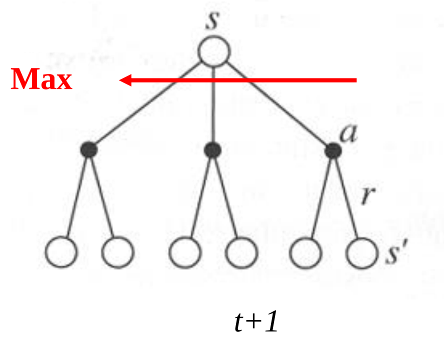
Politica greedy: scelgo l'azione ottimale.
Ha senso per il robot raccogli-lattine?

V*(s) – Osservazioni

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

Per ogni stato devo valutare:
• L'azione migliore ad un passo

Come valuto?
• analizzando reward a lungo termine



Policy iteration

Iterazione tra:

• Calcolo iterativo della Value function (iterative policy evaluation)
• Miglioramento della policy (policy improvement)

$$\pi_0 \rightarrow V^{\pi 0} \rightarrow \pi_1 \rightarrow V^{\pi 1} \rightarrow \pi_2 \rightarrow V^{\pi 2} \rightarrow \ldots\ldots.. \rightarrow \pi^* \rightarrow V^*$$

Converge velocemente ad una buona politica
(cf. Software Sommaruga)

Algoritmo

Inizialization

V(s) = 0;
π(s,a) = random (e.g. equiprobabile);

Repeat
    point 2.
    point 3.
until policy_stable

Algoritmo – point2

Policy evaluation – versione per trial

Repeat
  th = 0; // small value;
  for s = 1:N
    V_temp = $\sum_{a_j} \pi(s, a_j) \sum_{s'} \Pr_{s->s'|a_j} \left[R_{s->s'|a_j} + \gamma V(s') \right]$

      ΔV = |V(s) – V_temp)|
      V(s) = V_temp;
      th = max(th, ΔV)
  end;
until th < th_max;

Algoritmo – point3

Policy improvement

policy_stable = true;
for s = 1:N   // in alternativa, scelgo uno stato
  a_old = π(s);
  a_new = $\arg\max_a \left(\sum_{s'} \Pr_{s->s'|a} \left[R_{s->s'|a} + \gamma V(s') \right] \right)$;

  if (a_new ≠ a_old)
    policy_stable = false;
end;

Algoritmo – II

Policy evaluation – versione per epoch

Repeat
  Th = 0; // small value;
  for s = 1:N
    V_temp(s,a) = $\sum_{a_j} \pi(s, a_j) \sum_{s'} \Pr_{s->s'|a_j} \left[R_{s->s'|a_j} + \gamma V(s') \right]$

      ΔV = |V(s) – V_temp(s))|
      th = max(th, ΔV)
    end:
  end;
  for s = 1:N
    V(s) = V_temp(s);
  end; end;
until th < th_max;

Max or soft max

Policy improvement

policy_stable = true;
for s = 1:N   // in alternativa, scelgo uno stato
  a_old = π(s);

a_new = $\arg\max_a \left\{ \sum_a \pi(s,a) \sum_{s'} \Pr_{s->s'|a} \left[R_{s->s'|a} + \gamma V(s') \right] \right\}$

if (a_new ≠ a_old)
    policy_stable = false;
end;

Max con policy ε-greedy, soft-max, ...

Iterative policy evaluation sulla value function V(s)

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

Converge al limite a $V^{\pi}(s)$. Come facciamo a troncare?

Value iteration

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

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

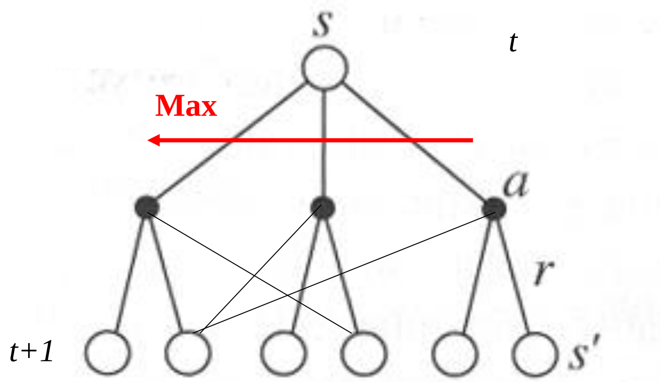
$$V_{k+1}(s) = \max_a \sum_a \pi(a, s) \sum_{s'} P_{s \to s'|a} \left[R_{s \to s'|a} + \gamma V_k(s') \right]$$

$\forall$s

Visualizzazione grafica

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

Full back-up



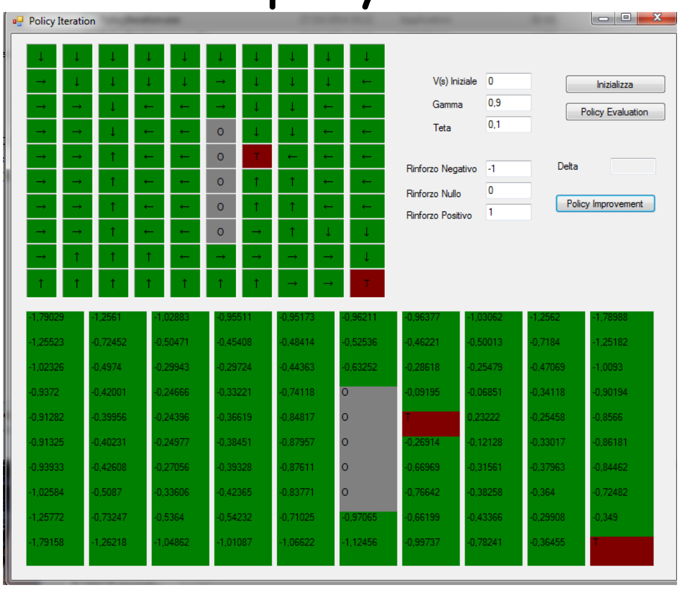
Sommario

Value iteration

Esempi

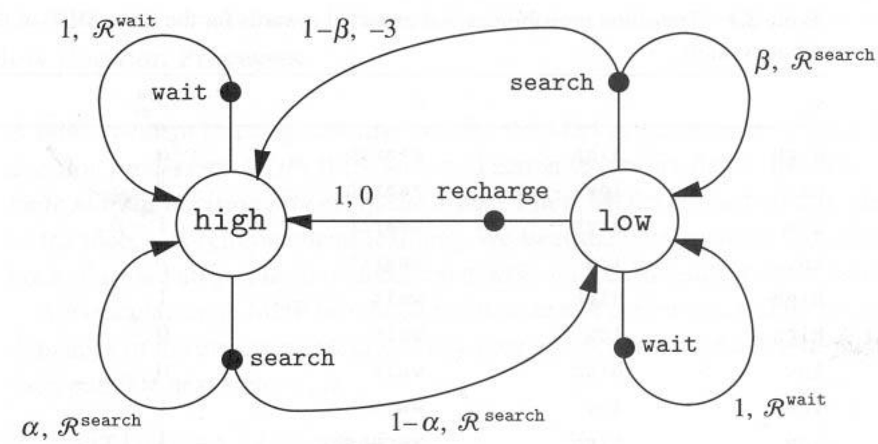
Learning with temporal differences

Iterative policy evaluation



 Forlivesi PolicyIteration Labirinto

Robot cerca-lattine



Esempio: robot – Policy deterministica

Q(h,search) = Pr(h->l, search) x [R(h->h, search) + γ x Q(h,search)]
Pr(h->h, search) x [R(h->l, search) + γ Q(l,wait)]
Q(h,search) = 0.4x[3+0.8xQ(h,search)]+0.6x[3+0.8Q(l,wait)]

Q(l,wait) = Pr(l->l, wait) x [R(l->l, wait)+0.8 x Q(l,wait)]
Q(l,wait) = 1 x [1+0.8 Q(l,wait)]

Policy iniziale deterministica:

STATO: Q(h,search) ➔
Q(h,s) ≅ 4,4 + 0.7 Q(l,w) ≅ 7.95
STATO: Q(l, wait) ➔
Q(l,wait) = 5

Posso migliorare la policy?

Esempio: robot – miglioramento policy

Miglioro la policy, modificando l'azione associata a s = low:

STATO: high
a = search ➔ Q(h,search) ≅ 4,4 + 0.7 Q(l,recharge) = ??? ≠7.95
STATO: low
a = recharge ➔ Q(l,recharge) = 0 + 0.8 Q(h,search) = ???

Ho stimato correttamente Q(h,search)? No
Applico un passo di iterative policy evaluation, in modalità trial.
STATO: Vl
a = recharge → Q(l,r) = 0.8 Q(h,s) = 0.8 x 7.95 = 6.36
STATO: high
a = search ➔ Q(h,s) ≅ 4.4 + 0.7 Q(l,r) ≅ 4.4 + 0.7 x 6.36 = 8.85
Ho stimato correttamente Q(s,a)? No. Devo iterare la policy evaluation.

Esempio: robot - IV

<u>Asintoticamente</u> calcolo il valore vero delle coppie stato-azione:

STATO: high
a = search ➔ Q(h,s) ≅ 4.4+0.7 Q_1(l,r) = 4.4 +0.7x6.36 = 8.85
STATO: low
a = recharge → Q(l,r) = 0.8 Q(h,s) = 0.8x8.85 = 7.08

Potrei ottenere gli stessi valori ottenuti asintoticamente, risolvendo il sistema lineare:
Q(h,s) = 4.4 + 0.7 Q(l,r) = 10
Q(l,r) = 0.8 Q(h,s) = 8

A questi valori si arriva solo asintoticamente

Sommario

Value iteration

Esempi

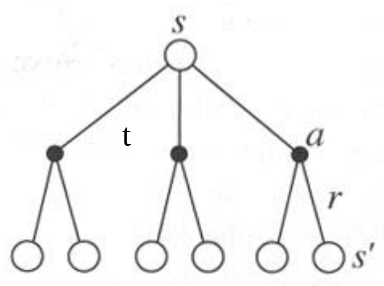
Learning with temporal differences

Tecnica full-backup

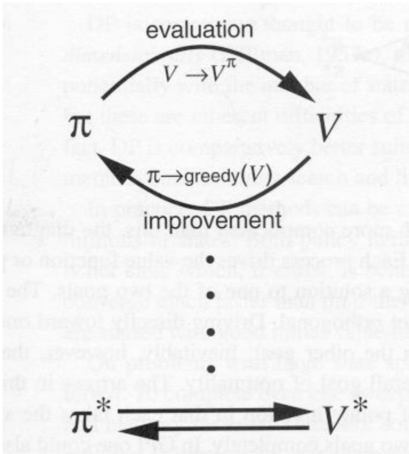
π(s,a) fissata

Back-up

t+1



Conosciamo $V_k(s_t)$ $\forall s_t$, anche per s'_{t+1} quindi:
Analizziamo la transizione da s_t,a_t -> (s'_{t+1})
Calcoliamo un nuovo valore di V per s: $V_{k+1}(s_t)$ congruente con:
$V_k(s_{t+1})$ ed r_{t+1}
Full backup se esaminiamo tutti gli s', e tutte le a (cf. DP).
Da s' mi guardo indietro ed aggiorno V(s).



Schema di Apprendimento

Generalized Policy iteration $\begin{cases} \text{Policy iteration} \\ \text{Value iteration} \end{cases}$

Competizione e cooperazione -> V corretta e policy ottimale.

World RL competition

Started in NIP2006.

It became very popular and started a workshop on its own. Last official challenge at COLT was at 2014. Still on-going research and specific challenges.

Visit: http://www.rl-competition.org/

Project Malmo competition. https://www.microsoft.com/en-us/research/project/project-malmo/ (2019)

How About Learning the Value Function?

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

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

È una funzione dello stato.

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

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

Full backup, for all states

Value iteration

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

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

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

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

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

Problema legato alla conoscenza della risposta dell'ambiente

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

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.

Osservazioni

Iterazione tra:
• Calcolo della Value function

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

•Miglioramento della policy

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

Non sono noti

Background su Temporal Difference (TD) Learning

Al tempo t abbiamo a disposizione:

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

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

Reward certo
Transizione certa
vengono misurati dall'ambiente

Come si possono utilizzare per apprendere?

Confronto con il setting associativo

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

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 = $\alpha = 1/k$ $\qquad$ *a = cost*
Rewards weight w = 1 $\qquad$ *Weight of i-th reward at time k: w =* $(1-a)^{k-i}$

Qual è la differenza introdotta dall'approccio DP?

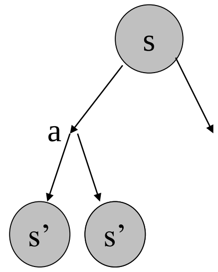
Un possibile aggiornamento

In iterative policy evaluation ottengo questo aggiornamento:

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

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

$$V_{k+1}(s) = \left[r' + \gamma V_k(s) \right]$$



Qual'è il problema?

Un possibile aggiornamento di Q(s,a)

$$Q_{k+1} = Q_k - \frac{Q_k}{N_{k+1}} + \frac{r_{k+1}}{N_{k+1}} = \quad Q_k + \alpha\left[r_{k+1} - Q_k \right] = Q_k + \Delta Q_k$$

Quanto vale α?

$$V_k(s) = V_k(s) + \Delta V_k(s)$$

Come calcolo $\Delta V_k(s)$?

TD(0) update

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

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

Da confrontare con la iterative policy evaluation:

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

Sample backup

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

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

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*

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*.

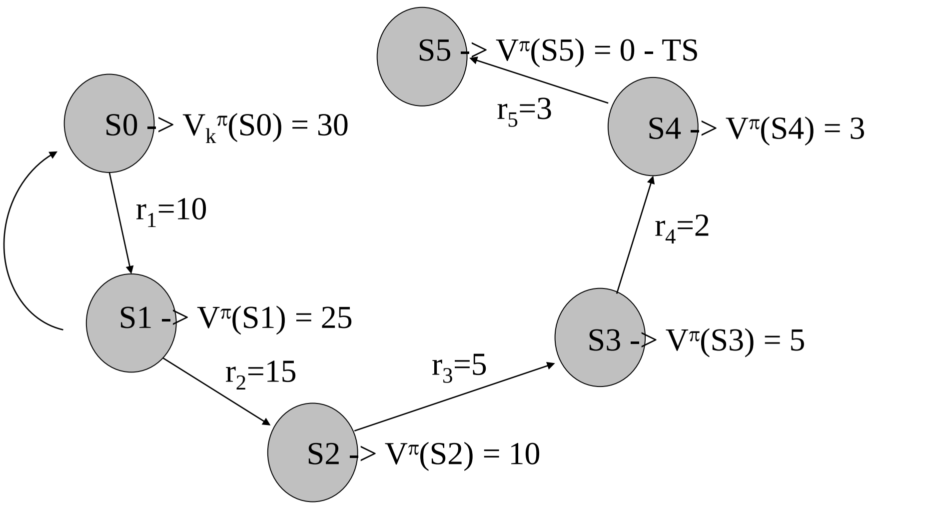
Esempio: valutazione della policy TD

	Stato	Tempo percorrenza stimato del segmento	Tempo percorrenza attuale del segmento	Tempo totale previsto in precedenza $V_k(s)$	Tempo totale previsto aggiornato $V_{k+1}(s)$	Increase or decrease V(s)
S0	Esco dall'ufficio	0	0	30	30 + (10 + 25 – 30) = 35	>
S1	Salgo in auto	5	10	25	25 + (15 + 10 - 25) = 25	=
S2	Esco dall'autostrada	15	15	10	10+ (10 + 5 - 10) = 15	>
S3	Esco su strada secondaria	5	10	5	5+ (2 + 3 - 5) = 5	=
S4	Entro Strada di casa	2	2	3	3 + (3 + 0 - 3) = 3	=
S5	Parcheggio	3	3	0	0	

V(s) è l'expected "Time-to-Go" - $\gamma = 1$ $\alpha = 1$

Learning $V_\pi(s)$

S0 -> $V_{k+1}^{\pi}(S0)$ = 30 + (10+25-30) = 35



Come i diversi reward istantanei modificano $V^\pi(s)$?

Alcuni passi di apprendimento di TD(0)

Alcuni passi di iterazione per TD(0)

V(0) = V(0) + α (r1 + γV(1) - V(0)) = 30 + α (5 + 35 − 30) = 30 + α * Δ 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) + α (r1 + γV(2) - V(1)) = 35 + α (20 + 15 − 35) = 35 + α * Δ
Stima iniziale del tempo di percorrenza dal parcheggio: 35m
Tempo di percorrenza fino ad uscita autostrada: 20m
Tempo di percorrenza fino ad uscita autostrada: 20m
Stima del tempo di percorrenza dall'uscita autostrada: 15m

Ruolo di α

$V(1) = V(1) + \alpha\,(r_1 + \gamma V(2) - V(1)) = 30 + \alpha\,(10+25 - 30) = 30 - \alpha*5$

Stima iniziale del tempo di percorrenza dal parcheggio: 30m
Tempo per raggiungere l'auto: 10m
Stima del tempo di percorrenza dall'uscita Dal parcheggio: 25m

$\alpha < 1$.

If $\alpha << 1$ aggiorno molto lentamente la value function.

If $\alpha = 1/k$ aggiorno la value function in modo da tendere al valore atteso. Devo memorizzare le occorrenze dello stato s.

If α = cost. Aggiorno la value function, pesando maggiormente i risultati collezionati dalle visite dello stato più recenti.

Sommario

Value iteration

Esempi

Learning with temporal differences