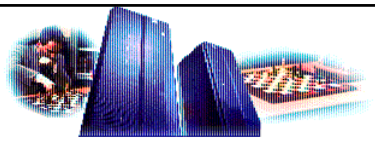
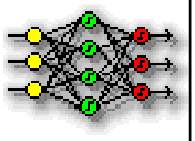




Sistemi Intelligenti
Reinforcement Learning: Policy Iteration

Alberto Borghese

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

Sommario

Teorema del miglioramento della policy

Policy iteration

Value iteration

Metodi asincroni

Calcolo iterativo della Value Function

Per ogni stato s, estratto a caso, analizziamo una singola transizione.

Equazione di Bellman per "iterative policy evaluation":

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

Mi fido di $V_k(s')$ (Backup)

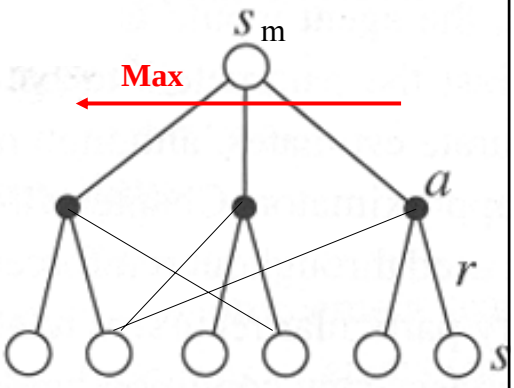
$$\lim_{k\to\infty}\{V_k(s)\}=V^{\pi}(s)$$

V*(s) – Osservazioni

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

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

Come valuto?
• analizzando reward a lungo termine



t+1

Miglioramento della policy

Tutti gli stati sono valutati in funzione di una policy data.

Condizioni di funzionamento dell'agente:
Policy deterministica: a = $\pi(s)$.
Ambiente stocastico.

Cosa succede se cambiamo la policy per un certo stato s? a' $\neq \pi(s_m)$.
Cosa viene influenzato?

Come faccio a valutare se miglioro la policy o no?

Effetto del cambiamento della policy

Cambia, a, cambiano i possibili stati successivi ad s, $\{s_{t+1}\}$, ed il reward a lungo termine:

$$Q^{\pi}(s_m, a_{new}) = E_{\pi}\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s_m, a_t = a_{new} \neq \pi(s_m)\right\} =$$

$$\sum_{s'} P^{a_{new}}_{s_m -> s'}\left[R^{a_{new}}_{s_m -> s'} + \gamma V^{\pi}(s')\right]$$

?

$$Q^{\pi}(s_m, a_{new}) >=< Q^{\pi}(s_m, a = \pi(s_m)) = V^{\pi}(s_m) \quad \forall s\,?$$

Il costo a lungo termine può essere maggiore (minore) solamente se aumenta (diminuisce) il costo totale "visto" ad un passo (costo del passo + costo successivo).

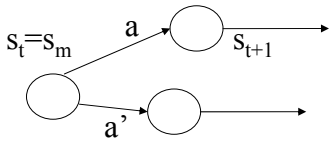
Enunciato del teorema del miglioramento della policy

$$Q^{\pi}(s,a) = \sum_k P_{s \to s_k | a} \left[R_{s \to s_k | a} + \gamma V^{\pi}(s_k) \right]$$

Ipotesi: $Q^{\pi}(s_m, \pi'(s_m)) \geq V^{\pi}(s_m)$

$$Q^{\pi'}(s, a_{new} = \pi'(s_m, a)) = \sum_k P_{s_m \to s_k | a_{new}} \left[R_{s_m \to s_k | a_{new}} + \gamma V^{\pi}(s_k) \right]$$

Tesi: π' è meglio di π. Cioè: $V^{\pi'}(s) >= V^{\pi}(s)$ $\forall$**s.**



Dimostrazione del teorema del miglioramento della policy

Analizziamo la seguente condizione:

$\pi' = \pi$ $\forall s$ tranne che per s_m per il quale si applica l'azione:
$a_{new} = \pi'(s_m)$

Risulta che il reward a lungo termine è maggiore per $a_{new} = \pi'(s)$.

$Q^{\pi'}(s, a_{new} = \pi'(s)) >= Q^{\pi}(s, a = \pi(s)) = V^{\pi}(s)$

Tesi: π' è meglio di π. Cioè: $V^{\pi'}(s) >= V^{\pi}(s)$ $\forall s$.

Dimostrazione del teorema del miglioramento della policy

Hp: $Q^{\pi}(s,\pi'(s)) >= V^{\pi}(s)$ $\forall s$ $\pi'(s,a)$ è migliore per almeno uno stato

$V^{\pi}(s) \leq Q^{\pi}(s,\pi'(s))$

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

$<= E_{\pi'}\{r_{t+1} + \gamma Q^{\pi}(s_{t+1}, \pi'(s_{t+1})) \mid s_t = s\}$

$<= E_{\pi'}\{r_{t+1} + \gamma E_{\pi'}(r_{t+2} + \gamma V^{\pi}(s_{t+2})) \mid s_t = s\}$

$= E_{\pi'}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 V^{\pi}(s_{t+2})) \mid s_t = s\}$

Sostituisco ancora $Q^{\pi}*(.)$

$<= E_{\pi'}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... \mid s_t = s\}$

Th: $V^{\pi}(s) <= V^{\pi'}(s)$

Sommario

Teorema del miglioramento della policy

Policy iteration

Value iteration

Metodi asincroni

Politica ottima

Miglioramento della politica per tutti gli stati.

$$\pi'(s_k) = \arg\max_a Q^\pi(s,a) \qquad \textit{greedy o } \varepsilon\textit{-greedy}$$

$$= \arg\max_a E\{r_{t+1} + \gamma V^\pi(s') \mid s_t = s, a_t = a\}$$

$\forall s$

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

Policy improvement

Si può estendere al caso di comportamento stocastico dell'agente nel qual caso: $\pi(s,a)$ è una probabilità.

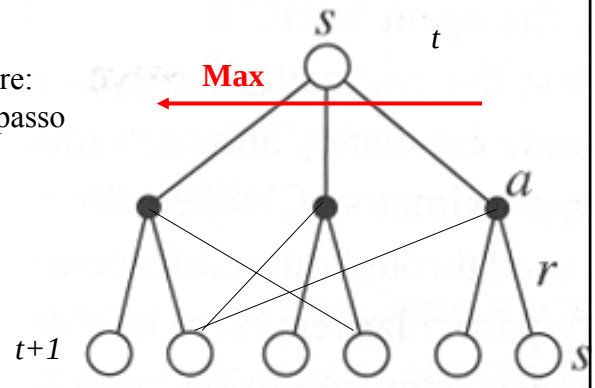
Policy ottima

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

$$\pi^*(s) = \arg\max_a Q^\pi(s,a) \qquad \forall s$$

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

Algoritmo (progetto per esame)

Repeat until policy-stable

1. Initialization
   $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$
2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} \left[\mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s')\right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)
3. Policy Improvement
   policy-stable $\leftarrow$ true
   For each $s \in \mathcal{S}$:
   $b \leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V(s')\right]$
   If $b \neq \pi(s)$, then *policy-stable* $\leftarrow$ false
   If *policy-stable*, then stop; else go to 2

Esercizio: autonoleggio

2 locazioni di autonoleggio

In ogni locazione, se quando arriva un cliente l'auto è disponibile, si guadagnano 10 euro.

Se l'auto non è disponibile si perde la gestione della locazione.

Le auto diventano disponibili il giorno dopo che sono state restituite dopo il noleggio.

Si possono portare le auto da un autonoleggio all'altro di notte al costo di 2 Euro per ogni auto.

Supponiamo che il numero di auto richieste e restituite in ognuno dei 2 autonoleggi sia rappresentato da una distribuzione di Poisson (probabilità che vengano richieste n auto: $(\lambda^n / n!)e^{-\lambda}$, dove λ è il valore atteso (media) di auto richieste o restituite.

Supponiamo che non ci possano essere più di 20 auto in uno dei 2 atuonoleggi (ciascuna auto aggiuntiva viene inviata al centro di raccolta nazionale della compagnia).

Supponiamo anche che un massimo di 5 auto possa essere spostato in una singola notte.

Questo problema is può formulare con un MDP dove lo stato è rappresentato dal numero di auto presenti in ciascuno dei 2 autonoleggi al termine di una giornata e le azioni il numero di auto che vengono spostate durante la notte.

Consideriamo $\gamma = 0.9$. Il reward sarà il reward accumulato durante il giorno + notte.

Partiamo dalla policy $\pi(s,a) = 0$: nessuna auto viene mossa. Siamo in grado di migliorarla? Come?

Sommario

Teorema del miglioramento della policy

Policy iteration

Value iteration

Metodi asincroni

Calcolo iterativo della Value Function

Per ogni stato s, analizziamo una singola transizione.

Equazione di Bellman per "***iterative policy evaluation***":

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

$$\lim_{k \to \infty} \{V_k(s)\} = V^{\pi}(s)$$

Politica ottima

Miglioramento della politica per tutti gli stati.

$\forall s$

$$\begin{aligned}
\pi'(s_k) &= \arg\max_a Q^{\pi}(s,a) \\
&= \arg\max_a Q^{\pi}(s,a) \\
&= \arg\max_a E\{r_{t+1} + \gamma V^{\pi}(s') \mid s_t = s, a_t = a\} \\
&= \arg\max_a \sum_{s'} P^a_{s->s'} \left[R^a_{s->s'} + \gamma V^{\pi}(s')\right]
\end{aligned}$$

Si può estendere al caso di comportamento stocastico dell'agente nel qual caso: $\pi(s,a)$ è una probabilità.


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


Iterative policy evaluation - problema

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

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

Value iteration

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

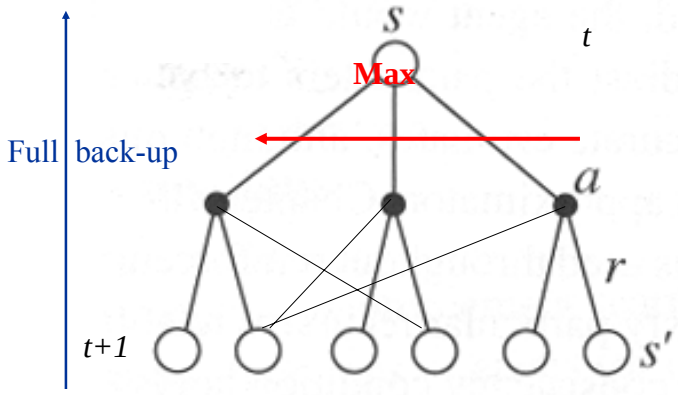
Invece di considerare una policy stocastica, consideriamo l'azione migliore:

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

$\forall s$

Visualizzazione grafica

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



Confronto con l'equazione di Bellman

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

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.

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

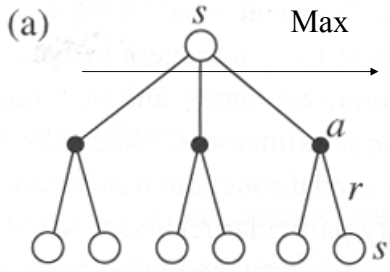
$$V_{k+1}(s)=\max_{a_j}\sum_{s'}P_{s\to s'|a}\left[R_{s\to s'|a_j}+\gamma V_k(s')\right]$$

Confronto con l'equazione di backup

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

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

Nel caso della Value Iteration viene considerata solamente l'azione che fornisce il valore massimo.



Algoritmo di value iteration

V(s) = 0 ∀s compreso TS

Repeat
{ Δ = 0;
for s = 1:NS // Per ogni stato eccetto il TS
{ V_buffer = V(s);

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

Δ = max(Δ, |V(s) – V_buffer|);
}
} until (Δ < Th);

Output a deterministic policy such that:

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

Esempio

Consideriamo uno scommettitore che scommette su testa e croce.

Se esce testa, vince tanti Euro quanti ne ha scommesso. Se esce croce, perde gli Euro che ha scommesso.

Il gioco termina quando lo scommettitore arriva ad accumulare 100 Euro di vincita o perde tutto il suo capitale.

Il suo capitale di partenza è variabile tra 1 e 99 Euro. Ad ogni lancio della moneta, lo scommettitore può decidere quanto scommettere sul fatto che esca testa.

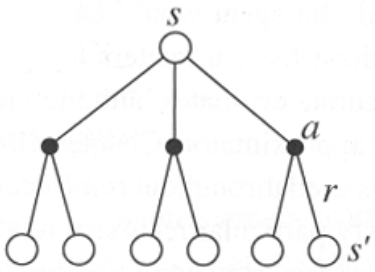
Undiscounted, episodico, MDP.

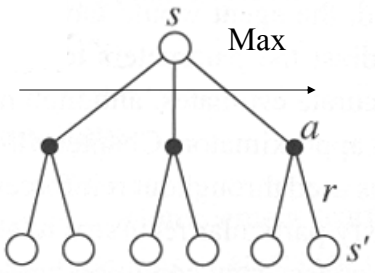
Lo stato è il capitale accumulato dal giocatore: s = {1, 2, …… , 99}.
L'azione è quanto viene scommesso ad ogni lancio: a = {1, 2, … min (s, 100 – s)}.
Conosciamo la probabilità p con cui esce testa (la moneta può essere truccata e quindi le due condizioni testa/croce possono essere non equiproababili).
La dinamica dell'ambiente non dipende da a ed è rappresentata dalla transizione di uno stato all'altro che a sua volta è funzione del fatto che esca testa o meno.
Il reward istantaneo = 0 tranne quando raggiunge 100 Euro, nel qual caso reward = +1.

Vogliamo massimizzare la probabilità di vincere. Si può fare tramite Value iteration.

1° passo di Value iteration

$P_{s->s'}$ = {testa, croce} non dipende da a. Sono indipendenti.



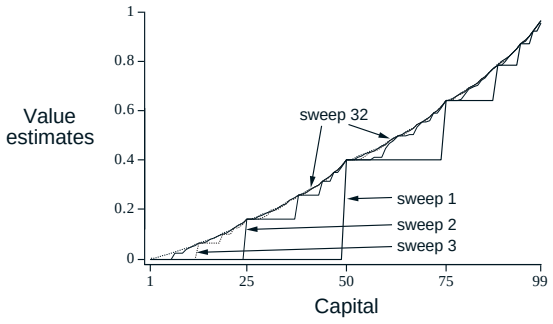


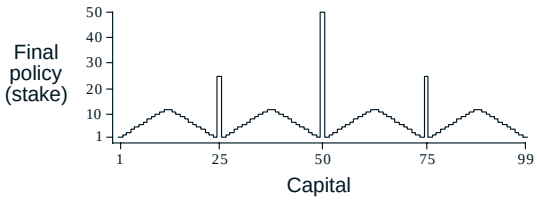
$s < 50$ | $s \geq 50$

$V_{k+1}(s) = 0.4\ (0 + \gamma\ V(s')) = 0$ | $V_{k+1}(s) = 0.4\ (1 + \gamma\ V(s')) = 0.4$

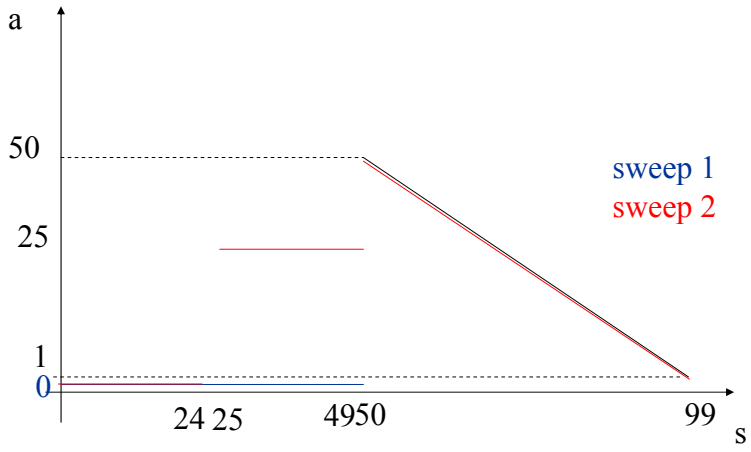
$\forall a$ | Per a scelto in modo da arrivare a 100€ in caso di vincita.

Stima della Value function





Policy selezionata



Ci sono diversi stati per cui la Q(s,.) assume lo stesso valore per diverse azioni.

Sommario

Teorema del miglioramento della policy

Policy iteration

Value iteration

Metodi asincroni

Problemi

Framework: DP

L'analisi di tutti gli stati può essere computazionalmente molto pesante: ***curse of dimensionality*** (of the state space).
Nel frattempo la policy non evolve.

La risposta dell'ambiente è nota (ne è noto il modello statistico).

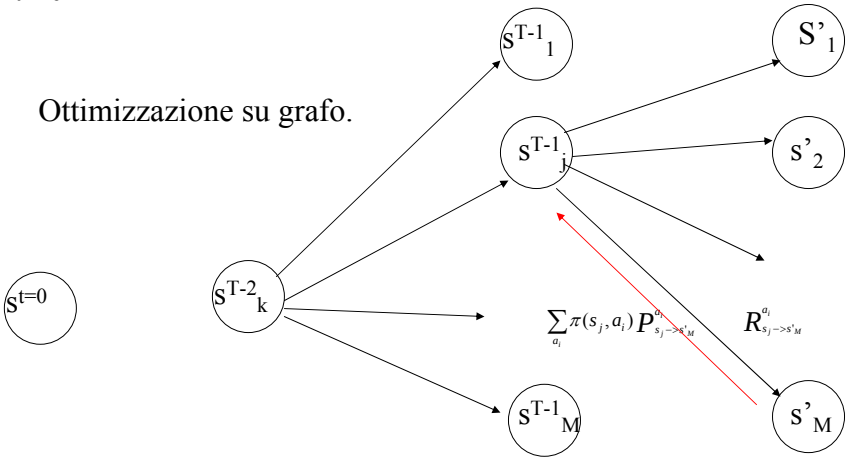
Per problemi ad orizzonte finito -> ottimizzazione su grafo.

Problemi ad orizzonte finito

t= 0

t= T

Ottimizzazione su grafo.



Backwards ottimization

$V(s'_M) = r^T(s'_M)$

Asynchronous solution

Asynchronous DP.

Si può applicare ad un agente in azione. Vengono aggiornati solamente gli stati visitati dall'agente durante la sua azione.

Possiamo scegliere l'ordine in cui gli stati sono visitati. Non è più richiesto un ciclo.

Evoluzione della metodologia

Policy iteration:

Iterative policy evaluation

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

$$\lim_{k \to \infty} \{V_k(s)\} = V^{\pi}(s)$$

∀s

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

Value iteration:

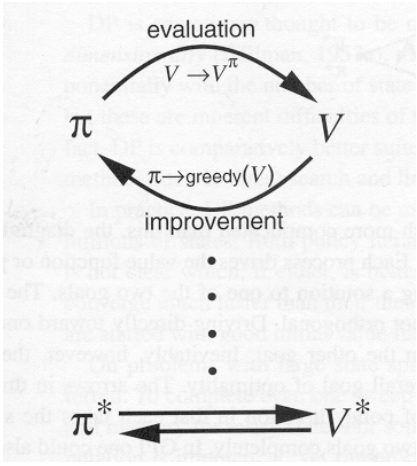
$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P_{s \to s'|a} \left[R_{s \to s'|a} + \gamma V_k(s')\right]$$

∀s

Asynchonous solution:

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

1 stato



Riassunto

Generalized Policy iteration { Policy iteration, Value iteration, Asynchronous DP

Competizione e cooperazione -> V corretta e policy ottimale.

Sommario

Teorema del miglioramento della policy

Policy iteration

Value iteration

Metodi asincroni