

Sistemi Intelligenti Reinforcement Learning: Temporal Differences

Alberto Borghese

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



A.A. 2005-2006

1/28

<http://homes.dsi.unimi.it/~borgnese/>



Sommario



Policy Iteration

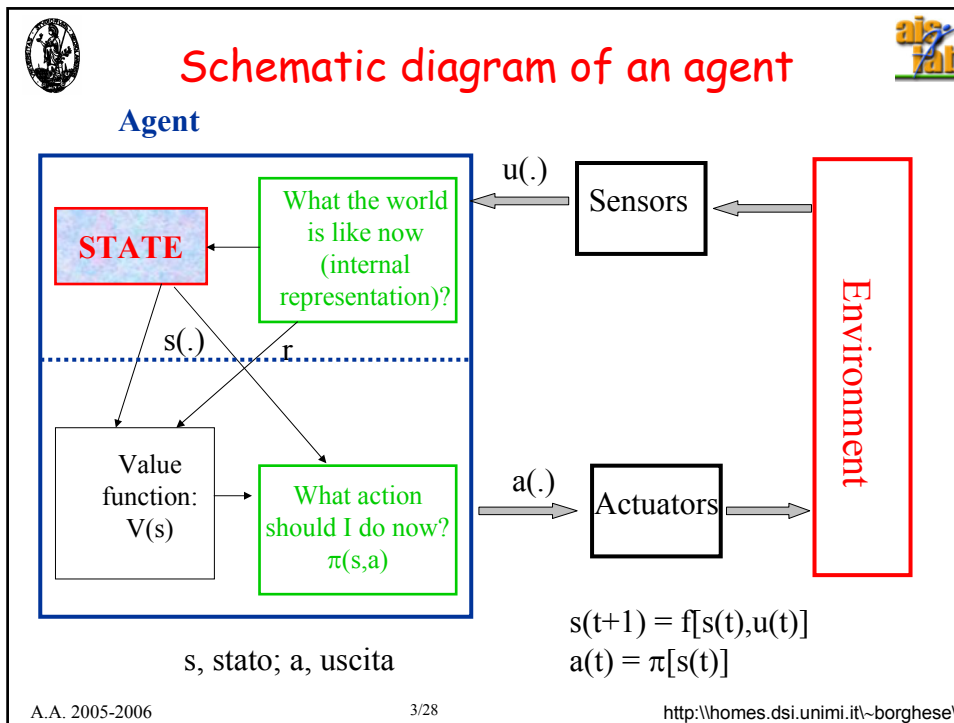
Generalized Policy Iteration

Temporal differences

A.A. 2005-2006

2/28

<http://homes.dsi.unimi.it/~borgnese/>



-
- Alcuni richiami: l'Agente**
- Può scegliere un'azione sull'ambiente tra un insieme continuo o discreto.
 - L'azione dipende dalla situazione. La situazione è riassunta nello stato del sistema (policy: $a = \pi(s)$).
 - L'agente monitora continuamente l'ambiente (input) e modifica continuamente il suo modello dell'ambiente.
 - A partire dal modello dell'ambiente, l'agente può costruirsi una funzione valore.
 - Supponiamo problemi ad orizzonte finito. Ogni agente esegue ripetutamente il task. Ogni esecuzione viene definita episodio (o trial).
- A.A. 2005-2006 4/28 <http://homes.dsi.unimi.it/~borghese/>



Calcolo della Value Function



Per ogni stato s , analizziamo una singola transizione.

Equazione di Bellman per “iterative policy evaluation”:

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

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



Politica ottima



Miglioramento della politica per tutti gli stati.

$$\begin{aligned} \pi^*(s_k) &= \arg \max_a Q^\pi(s, a) \\ &= \arg \max_a Q^\pi(s, a) \\ \forall s &= \arg \max_a E\{r_{t+1} + \gamma V^\pi(s') \mid s_t = s, a_t = a\} \\ &= \arg \max_a \sum_{s'} P_{s \rightarrow s' | a}^a [R_{s \rightarrow s' | a}^a + \gamma V^\pi(s')] \end{aligned}$$

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



Policy improvement



Iterazione tra:

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



Problema: Value iteration



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

Converge al limite. Come facciamo a troncare?

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

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

$\forall s$



Confronto con l'equazione di Bellman



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

$V^*(s)$ 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 .

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

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

A.A. 2005-2006

9/28

<http://homes.dsi.unimi.it/~borghese/>



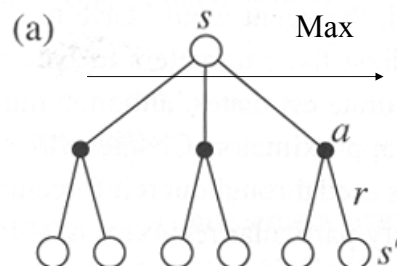
Confronto con l'equazione di backup



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

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

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



A.A. 2005-2006

10/28



Algoritmo (progetto per esame)



A.A. 2005-2006

11/28

<http://homes.dsi.unimi.it/~borghese/>



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 di 100 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: $s = \{1, 2, \dots, 99\}$

L'azione è quanto viene scommesso ad ogni lancio: $a = \{1, 2, \dots, \min(s, 100 - s)\}$

Il reward istantaneo = 0 tranne quando raggiunge 100 Euro, nel qual caso reward = +1.

Conosciamo la probabilità p con cui esce testa (la moneta può essere truccata e quindi le due condizioni testa/croce possono essere non equiprobabili).

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

A.A. 2005-2006

12/28

<http://homes.dsi.unimi.it/~borghese/>



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.

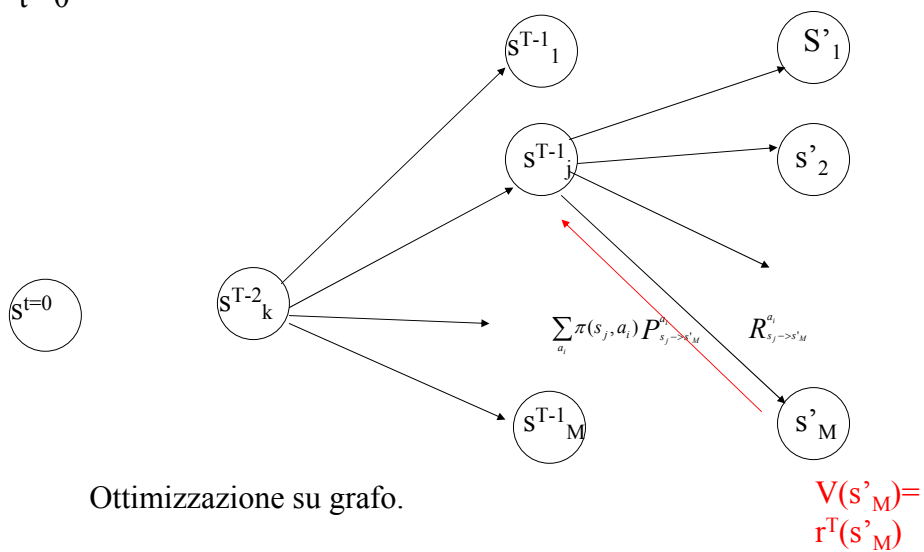


Problemi ad orizzonte finito



t=0

t=T





Sommario



Policy Iteration

Generalized Policy Iteration

Temporal differences



Problemi



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).



Asynchronous solution



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

Asynchronous DP.

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



Evoluzione della metodologia



Policy iteration:

Iterative policy evaluation

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

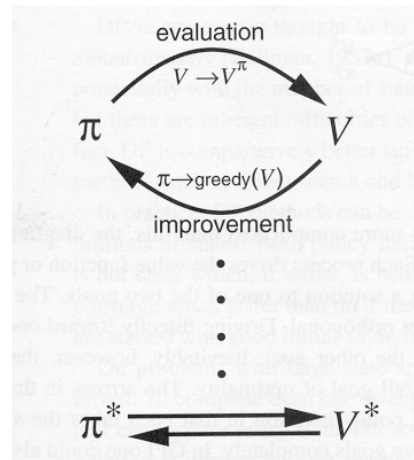
$$\forall s \quad \pi'(s_k) = \arg \max_a \sum_{s'} P_{s \rightarrow s' | a}^a [R_{s \rightarrow s' | a} + \gamma V^\pi(s')]$$

Value iteration:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P_{s \rightarrow s' | a} [R_{s \rightarrow s' | a} + \gamma V_k(s')] \quad \forall s$$

Asynchronous solution:

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



Generalized Policy iteration

{ Policy iteration
Value iteration
Asynchronous DP

Riassunto

Competizione e cooperazione -> V corretta e policy ottimale.



Sommario

Policy Iteration

Generalized Policy Iteration

Temporal differences



Agente in ambiente non noto



Il funzionamento stocastico dell'ambiente è noto. Vogliamo rimuovere questo vincolo. Come?

Occorre che sia l'agente durante l'interazione con l'ambiente a costruirsi il modello stesso dell'ambiente.

Value function computation -> Policy optimization

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



Alcuni richiami: update Value function ricorsiva



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

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:

$$\begin{aligned} \text{NewEstimate} &= \text{OldEstimate} + \text{StepSize} [\text{Target} - \text{OldEstimate}] \\ \text{NewEstimate} &= \text{OldEstimate} + \text{StepSize} * \text{Error}. \end{aligned}$$

$$\text{StepSize} = \alpha = 1/k$$

Qual è la differenza introdotta dall'approccio DP?



Alcuni richiami: DP update



Iterazione tra:

- Calcolo della Value function

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

- Miglioramento della policy

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

Non sono noti



TD(0) update



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

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

Da confrontare con la value iteration della DP:

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

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

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



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



```
Inizializziamo  $V(s) = 0$ .
Inizializziamo la policy:  $\pi(s,a)$ 
Repeat
{
   $s = s_0$ ;
  Repeat // For each state until terminal state
  {
     $a = \pi(s)$ ;
     $s\_next = NexState(s, a)$ ;
     $reward = Reward(s, s\_next, a)$ ;
     $V(s) = V(s) + \alpha [reward + \gamma V(s\_next) - V(s)]$ ;
     $s = s\_next$ ;
  } until TerminalState
} Until convergence of  $V(s)$  for policy  $\pi(s,a)$ 
```

Quanto vale α ?

A.A. 2005-2006

25/28

<http://homes.dsi.unimi.it/~borghese/>



Esempio: valutazione della policy mediante TD



Obiettivo: predire la durata del percorso per tornare a casa.

Stato	Tempo trascorso	Tempo previsto	Tempo totale
Esco dall'ufficio	0	30	30
Salgo in auto (neve)	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

A.A. 2005-2006

26/28

<http://homes.dsi.unimi.it/~borghese/>



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.



Sommario



Policy Iteration

Generalized Policy Iteration

Temporal differences