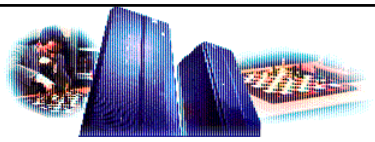
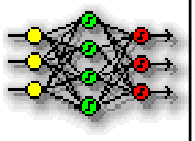




Sistemi Intelligenti
Reinforcement Learning: Temporal Differences

Alberto Borghese

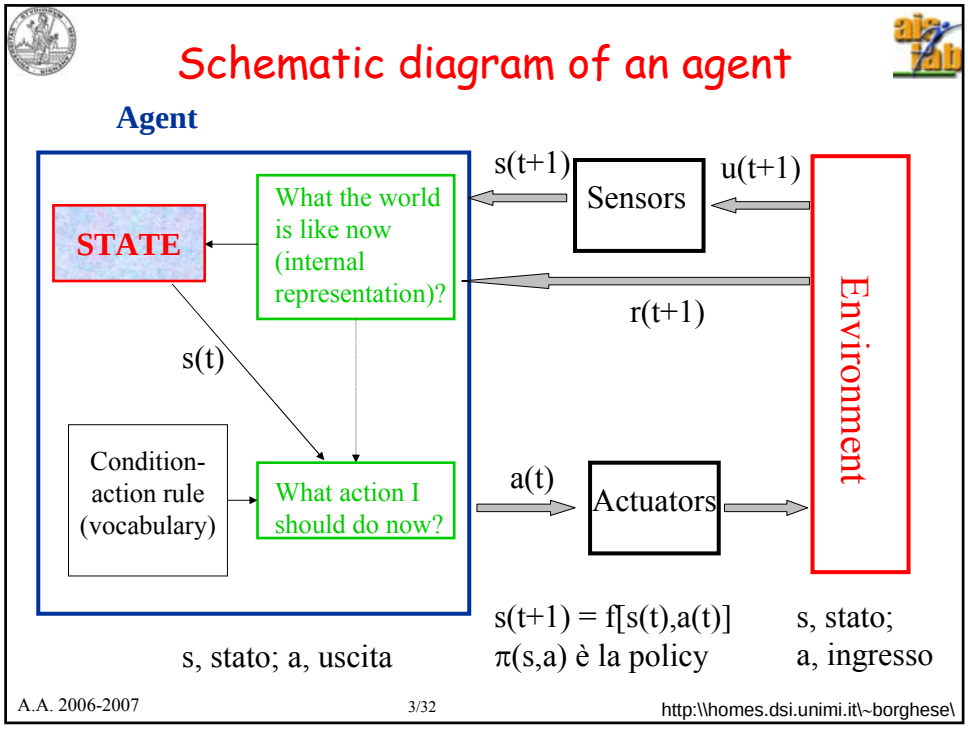
Università degli Studi di Milano
Laboratorio di Sistemi Intelligenti Applicati (AIS-Lab)
Dipartimento di Scienze dell'Informazione
borghese@dsi.unimi.it

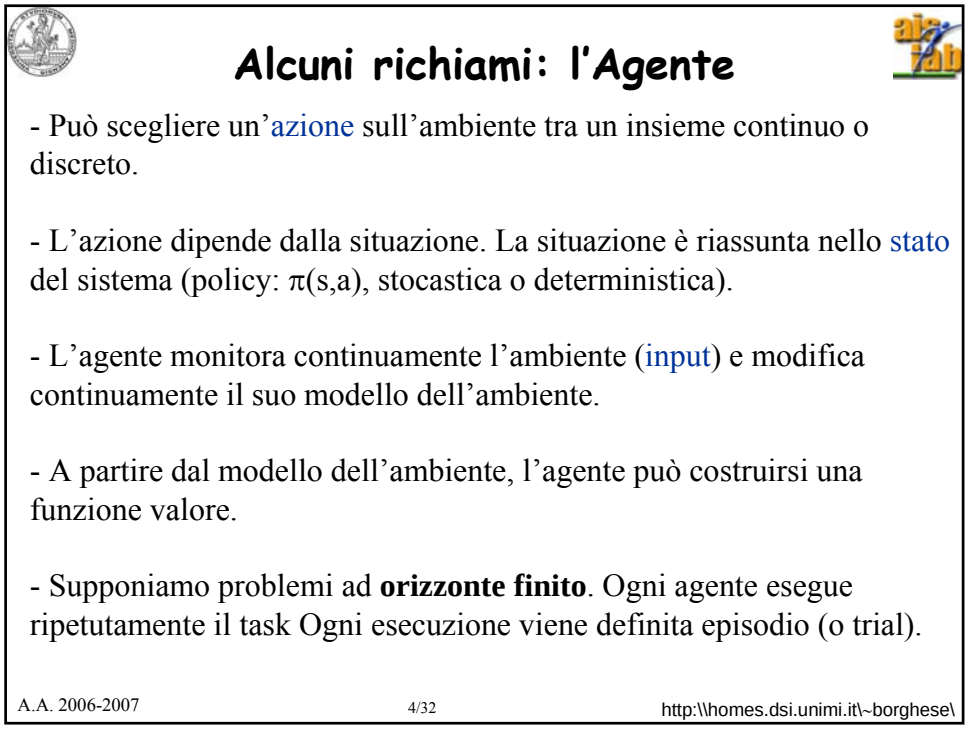
Sommario

Policy Iteration

Generalized Policy Iteration

Temporal differences


Schematic diagram of an agent
Agent
STATE
What the world is like now (internal representation)?
s(t+1)
Sensors
u(t+1)
Environment
r(t+1)
s(t)
Condition-action rule (vocabulary)
What action I should do now?
a(t)
Actuators
s(t+1) = f[s(t),a(t)]
π(s,a) è la policy
s, stato; a, uscita
s, stato; a, ingresso
A.A. 2006-2007
3/32
http:\\homes.dsi.unimi.it\~borghese\


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: π(s,a), stocastica o deterministica).
- 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. 2006-2007
4/32
http:\\homes.dsi.unimi.it\~borghese\

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.

$$\pi'(s_k) = \arg\max_a Q^{\pi}(s,a)$$

$$= \arg\max_a Q^{\pi}(s,a) \quad \forall s$$

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

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

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

Problema: Value iteration

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

Converge al limite. Come facciamo a troncare?

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

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

$\forall s$

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.

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

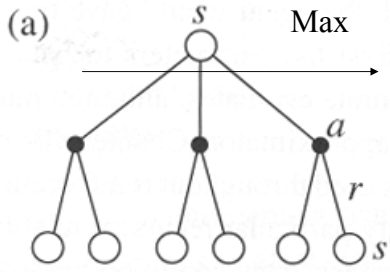
$$V_{k+1}(s)=\max_{a_j}\sum_{s'}P_{s\to s'|a}\left[R_{s\to s'|a_j}+\gamma V_k(s')\right]$$

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.

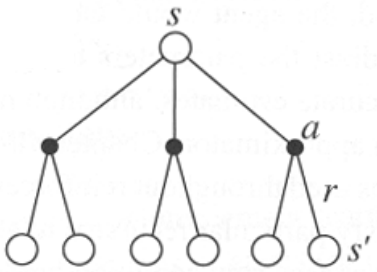
Undiscounted, episodico, MDP.

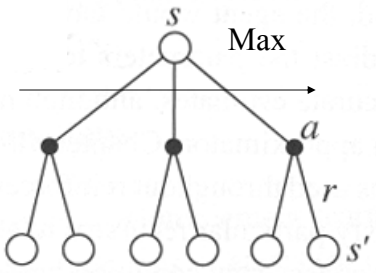
Lo stato è il capitale accumulato dal giocatore: s = {1, 2, …… , 99}.
L'azione è quanto viene scommesso ad ogni lancio: a = {1, 2, … min (s, 100 – s)}.
Il reward istantaneo = 0 tranne quando raggiunge 100 Euro, nel qual caso reward = +1.
Conosciamo la probabilità p con cui esce testa (la moneta può essere truccata e quindi le due condizioni testa/croce possono essere non equiproabili).

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.





$s < 50$

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

$\forall a$

$s \geq 50$

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

Per a scelto in modo da arrivare a 100€ in caso di vincita.

Problemi

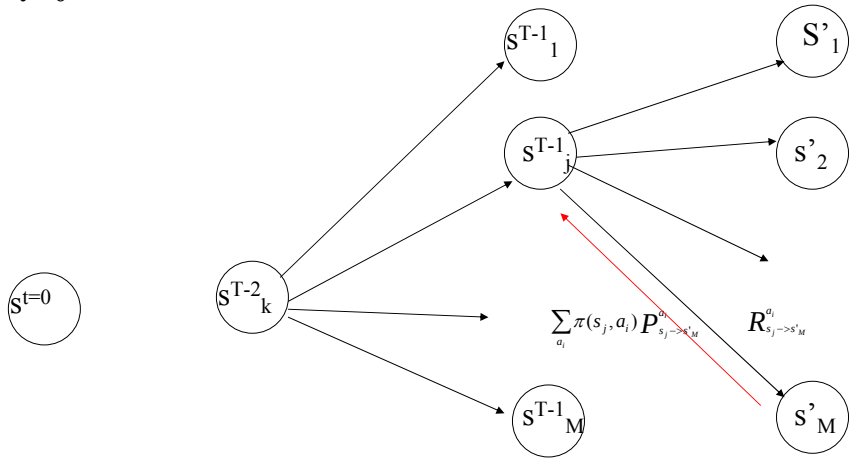
Framework: DP

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



Ottimizzazione su grafo.

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

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.

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

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

Value iteration:

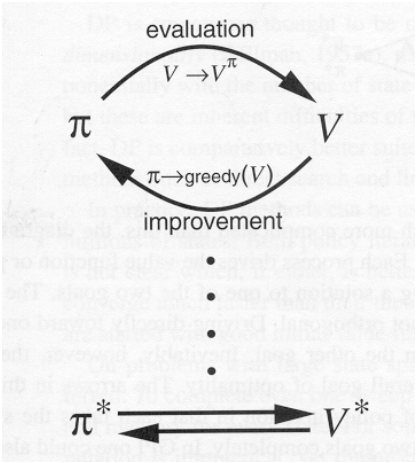
$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P_{s \to s'|a} \left[R_{s \to s'|a} + \gamma V_k(s')\right]$$

∀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

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 ricorsivo della Value function

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

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?

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 value iteration della DP:

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

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

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

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)

Quanto vale α?

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*" $\alpha = ?$

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

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

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

$\alpha = ?$

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