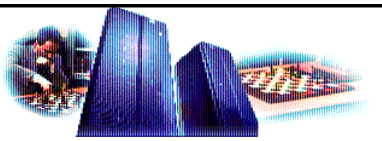
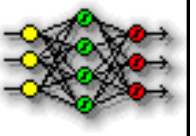




Sistemi Intelligenti
Reinforcement Learning:
Processi Markoviani e Value function

Alberto Borghese

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

Chapter 3 – Barto Sutton

Sommario

Agenti e sistemi dinamici

La value function: ricompensa a lungo termine.

Esempi di calcolo

Gli agenti

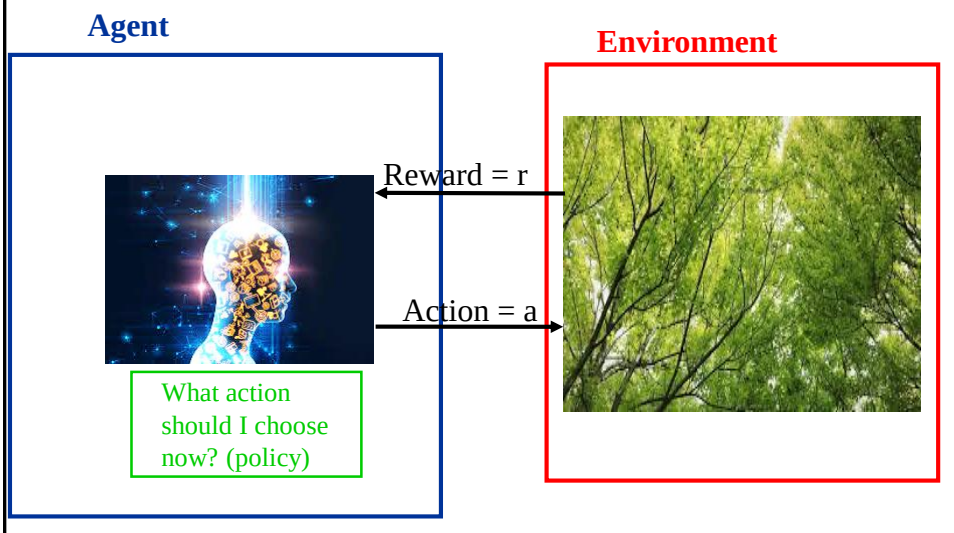
Agente (software): essere software che svolge servizi per conto di un altro programma, solitamente in modo automatico ed invisibile. Tali software vengono anche detti agenti intelligenti

"They are seen as a ***natural metaphor*** for conceptualising and building a wide range of complex computer systems (the world contains many passive objects, but it also contains very many *active* components as well);

They ***cut across a wide range of different technology and application areas***, including telecoms, human-computer interfaces, distributed systems, WEB and so on;

They are seen as a natural development in the search for ever-more powerful abstractions with which to build computer systems."

Schematic diagram of an agent



Dipende dalla situazione!

The same action, different outcomes

white wins

white looses

It depends on **the situation** (configuration of the pieces on the board)

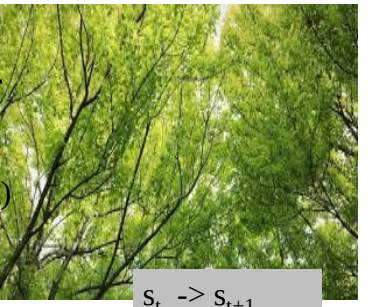
Evoluzione del sistema

Agent

Environment

Reward = r_t

Action $a_t = g(s_t)$



What action should I choose now? (policy)

$s_t \rightarrow s_{t+1}$

$s_{t+1} = f(s_t, a_t)$

L'azione dipende dalla situazione: $a_t = g(s_t)$!

La situazione dell'ambiente evolve nel tempo: **sistema dinamico**

$s_{t+1} = f(s_t, a_t)$

Lo stato

Lo stato è la situazione in cui si trova l'ambiente.

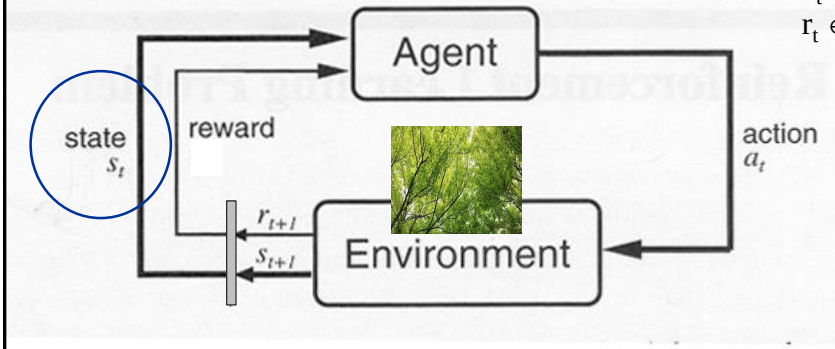
La situazione è il risultato dell'azione dell'agente sull'ambiente e dalla dinamica (**spesso non nota**) dell'ambiente stesso.

- La situazione è quindi il risultato della storia passata dell'interazione con l'ambiente.
- Lo stato deve essere:
  - efficiente per il raggiungimento del goal.
  - misurato dall'agente (osservabile).

Come rappresento lo stato, s?
- Memorizzo la sequenza temporale degli stimoli semplici di interesse. Utilizzo una variabile che riassume la situazione attuale (concetto di **stato**).

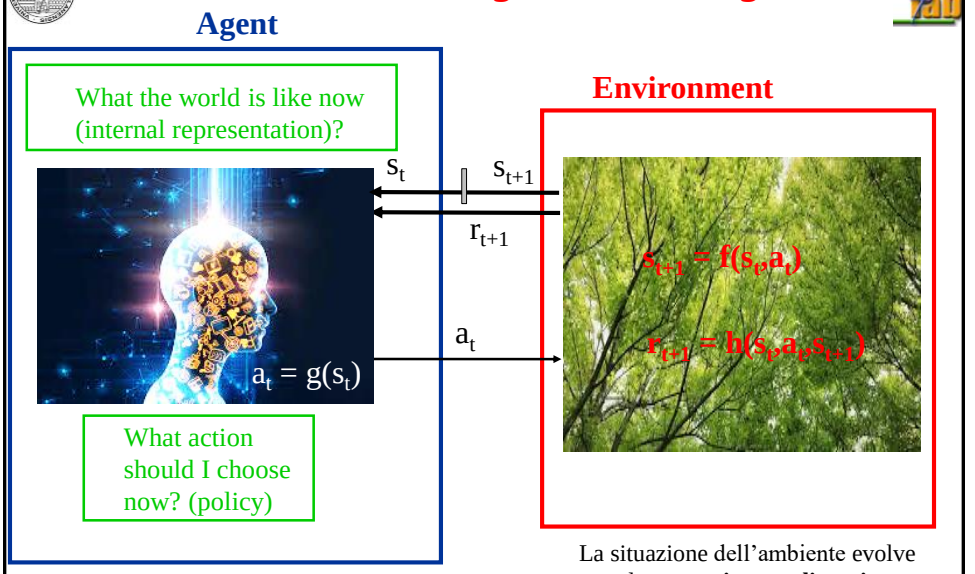
Interazione Agente-Ambiente

$s_t \in S$;
$a_t \in A$;
$r_t \in R$;



Il **reward** viene fornito dall'ambiente all'agente.
Il reward è un valore numerico, qualitativo, che viene fornito in certi istanti di tempo: $r_{t+1} = r(s_t, a_t, s_{t+1})$.

Schematic diagram of an agent



La situazione dell'ambiente evolve nel tempo: **sistema dinamico**
$\mathbf{s_{t+1} = f(s_t,a_t)}$

s, stato; a = uscita (azione dell'agente)
Policy: a = g(s): s -> a

a_t, dipende dalla situazione!

Environment Markoviano

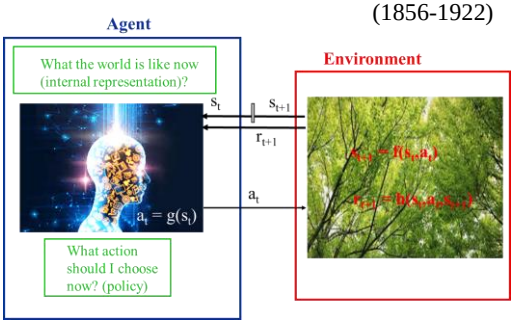
Andrej Markov (1856-1922)

Una variabile di stato (non funzione del tempo), che riassume le informazioni sulla storia del task, utili all'agente per agire, è detta variabile Markoviana.
Si definisce **processo stocastico markoviano**, un processo aleatorio in cui la **probabilità di transizione** che determina il passaggio a uno stato di sistema **dipende <u>solo</u> dallo stato del sistema immediatamente precedente** e non da come si è giunti a questo stato.

Formalizziamo. Supponiamo s_t e r_t variabili discrete appartenenti a un insieme finito di valori.

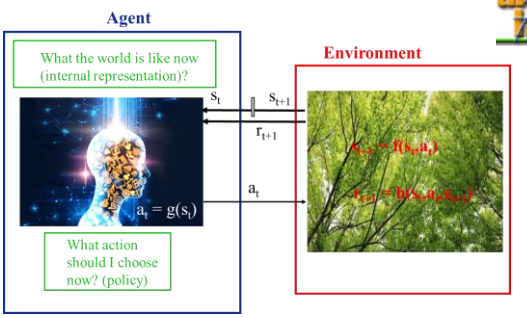
$\Pr\{s_{t+1}=s'|s_t,a_t; s_{t-1},a_{t-1}; \dots ; s_0,a_0\}$

Se lo stato è Markoviano:
$\Pr\{s_{t+1} = s' \mid s_t, a_t\}$



NB: Nella pratica $\Pr\{s_{t+1} = s' \mid s_t, a_t\}$ non è nota!

Rinforzo Markoviano



Reward stocastico.

$\Pr\{r_{t+1} = r' \mid s_t, a_t, s_{t+1}; s_{t-1}, a_{t-1}, r_t; \ldots\ldots; s_0, a_0, r_1\}$

Se lo stato è Markoviano:

Reward stocastico: $\Pr\{r_{t+1} = r' \mid s_t, a_t, s_{t+1}\}$
Reward deterministico: $r_{t+1} = r(s_t, a_t, s_{t+1})$.

L'ambiente ha completamente proprietà Markoviane.

I modelli Markoviani sono modelli molto generali!

Markov decision process

(Finite) Markov Decision Process.

$$P_{s->s'|a} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$$

Probabilità di transizione

$$R_{s->s'|a} = E\{r_{t+1} = r' \mid s_t = s, a_t = a, s_{t+1} = s'\}$$

Descrizione della dinamica dell'ambiente

Elemento chiave è lo stato (situazione)

Sommario

Agenti e sistemi dinamici

La value function: ricompensa a lungo termine

Esempi di calcolo

Il condizionamento classico

Il segnale di rinforzo è sempre lo stesso per ogni coppia input – output: Reward istantaneo, **non dipende dalla situazione**.

Esempio: risposta riflessa Pavloviana. Campanello (stimolo condizonante) prelude al cibo. Questo induce una risposta (salivazione). La risposta riflessa a uno stimolo viene evocate da uno stimolo condizionante.

Il rinforzo

"*Learning is an adaptive change of behaviour and what is indeed the reason of its existence in animals and man"* (K. Lorentz, 1977).

Rinforzo **puntuale istante per istante**, associato all'azione (condizionamento classico).

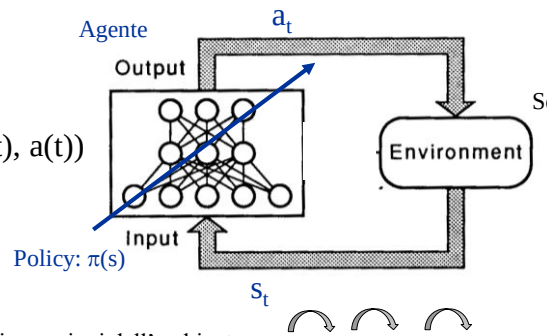
Rinforzo **puntuale istante per istante,** associato a un'azione in una **catena di azioni (comportamento).** (condizionamento operante).
Rinforzo non necessariamente a ogni istante),

Comportamento

Interessa un comportamento. Una sequenza di input / output che può essere modificata agendo sui parametri che definiscono il **comportamento dell'agente**: l'azione scelta in un certo stato (= situazione). Il comportamento dell'agente viene chiamato **policy**, $\pi(s)$.

L'agente deve scoprire quale azione (policy) fornisca la ricompensa massima provando le varie azioni (in stile trial-and-error) sull'ambiente con un criterio intelligente.

$a(t) = \pi(s(t))$
$s(t+1) = (g(s(t), a(t))$



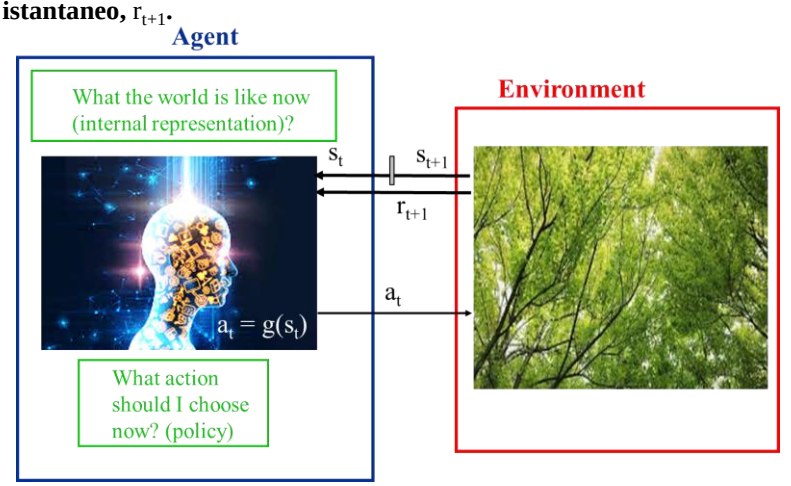
Comportamenti
=
Sequenza di azioni

Intreccio di azioni e reazioni dell'ambiente: $s_0, a_0; s_1, a_1; s_2, a_2 \ldots$

Apprendimento

L'agente vuole scegliere a in modo tale da massimizzare il **reward TOTALE** accumulato **nel tempo.** A partire dallo stato (situazione) in cui si trova deve scegliere il comportamento (le azioni) che massimizzano il **reward a lungo termine** a partire dal **reward istantaneo,** r_{t+1}**.**

.



Vogliamo costruire agenti lungimiranti che non siano greedy e miopi

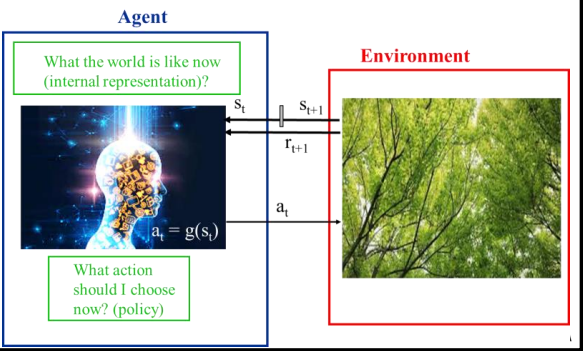
Ruolo dell'agente

Può scegliere un'azione sull'ambiente, tra un insieme continuo o discrete di azioni (policy)

L'azione dipende dalla situazione. La situazione è riassunta dallo stato del sistema.

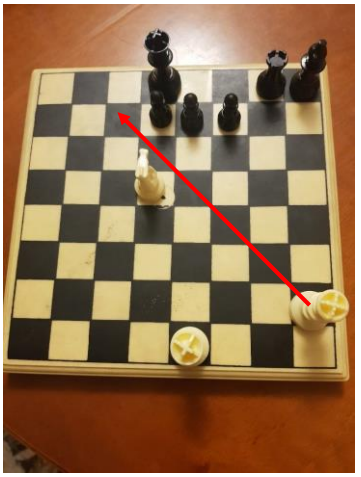
La scelta dell'azione è non banale non può scegliere semplicemente l'azione che massimizza il reward istantaneo, richiede un certo grado di "intelligenza".

L'agente ha un'intelligenza legata alla a memoria. Non può mantenere in memoria tutto il passato.



Esempio

white gains a knight at time t (r > 0)
white looses (bad total reward)



white gains nothing at time t (r = 0)
white wins (good total reward)

Deve guardare oltre la mossa greedy

Postural control is a complex problem

Complex system: multi-input – multi-output (each leg has 56 major muscle groups).

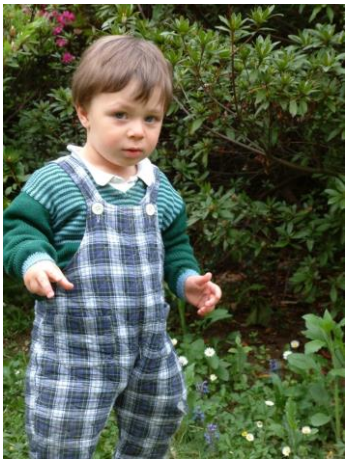
It is a non-linear system. High coupling between body segments (e.g. biarticular muscles).

Muscles bandwidth is limited.

The control system introduces delays, increasing from the periphery to the CNS.

Classical control theory is “difficult”.

Nevertheless, we learn upright posture in the very first year of our life.

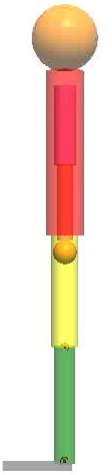


Results on simulations

Hat model of the human body
Muscles with maximum power and limited time constants.
Control of the joints of the leg.

Reinforcement learning with reinforcement signal when falling.

Video: APPR_tutto.m1v



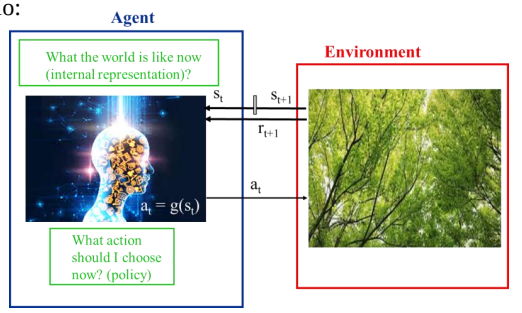
Value function

Cosa si intende per ricompensa a lungo termine?
Questa è rappresentata dalla **Value Function**; cosa rappresenta?

Al tempo t, **data una certa policy: π(s, a),** la ricompensa sarà una funzione dei reward negli istanti di tempo successivi a t, ad esempio:

$$R^{\pi}_{t}=r_{t+1}+r_{t+2}+r_{t+3}+r_{t+4}+\ldots..$$

$$R_t^{\pi} = \sum_{k=0}^{\infty} r_{t+k+1}$$



L'agente vuole massimizzare **il valore atteso del reward totale** (somma dei reward a partire dallo stato in cui si trova).

Infinite horizon problems (continuing tasks)

- We introduce a **discounted reward**: we give more weight to the rewards closer in the future and avoid that total reward diverges.

Discounted reward o discounted return:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+2} + \ldots\ldots = \sum_{k=0}^{+\infty} \gamma^k r_{t+(k+1)}$$

Dove $0 \leq \gamma \leq 1$ è il "discount rate".

R_t is the **value at time t**, of rewards that will be **collected in the future**.

$$R_t \leq \frac{z}{1-\gamma} \qquad \text{if} \quad r_{t+k} \leq z \qquad \forall k$$

Relazione con il caso non-stazionario nel setting non-associativo?

Cosa succede se $\gamma \to 0$ *e* $\gamma \to 1$?

Relazione con l'interazione statica

$$R_{N+1} = r_{N+1} + \gamma r_{N+2} + \gamma^2 r_{N+3} + \cdots + \gamma^M r_{N+1+M} = \sum_{i=0}^{M} \gamma^i r_{N+1+i} = Q_{N+1}(a_k)$$

Nel caso stazionario:

$$Q_{N+1} = \alpha\, r_{N+1} + (1-\alpha)\, Q_N$$

N+1 Peso di r decresce $\to \infty$

$$Q_{N+1} = (1-\alpha)^N Q_0 + \sum_{i=1}^{N} \alpha(1-\alpha)^{N-i} r_i =$$

$$= (1-\gamma) \sum_{i=0}^{N} \gamma^{N-i} r_i$$

$$= (1-\gamma) \sum_{i=0}^{N} \gamma^{i} r_{N-i}$$

$\gamma = 1 - \alpha$

$Q_0 = r_0$

$$Q_{N+1} = Q_N + \alpha\left[r_{N+1} - Q_N\right]$$

0 ← Peso di r decresce → N+1

Guardo indietro e stimo il reward a lungo termine a partire dal tempo attuale.
Guardo avanti e stimo il reward a lungo termine a partire dallo stato attuale.

Peso il reward via via piu' lontano (nel future o nel passato) con un'esponenziale decrescente.

Problemi a orizzonte temporale finito

Al tempo t, **data una certa policy: π(s, a),** la ricompensa sarà una funzione dei reward negli istanti di tempo successivi a t, ad esempio:

$$R_{N+1} = r_{N+1} + \gamma r_{N+2} + \gamma^2 r_{N+3} + \cdots + \gamma^{t-1} r_T$$

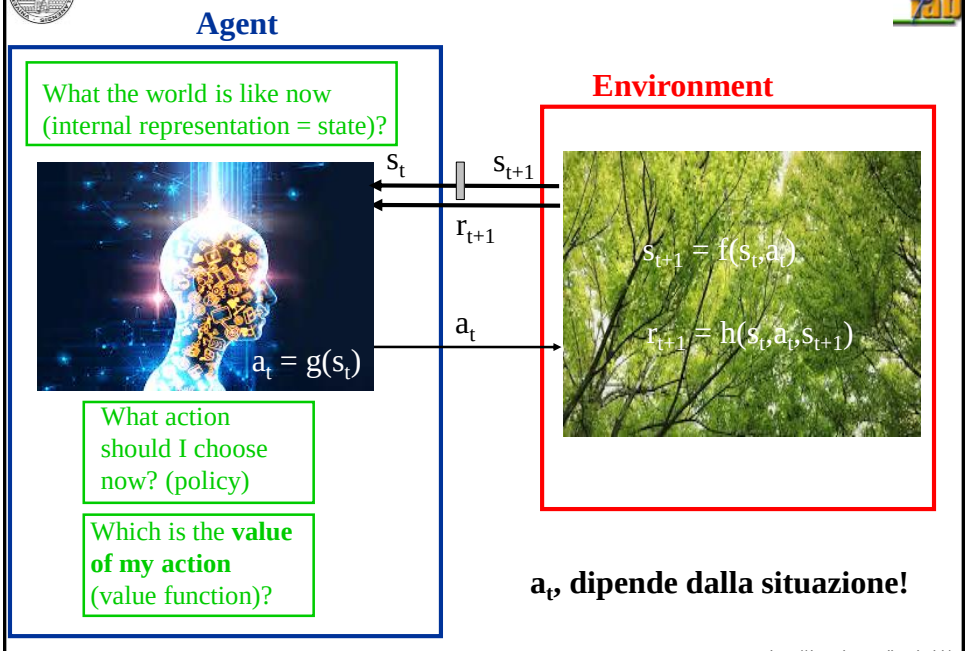
Terminal State

Quando è adeguata?

Problemi ad orizzonte finito (**episodic tasks, a terminal state is defined**).
Problemi stazionari.
Non ci sarebbe bisogno del discount, γ.

Obbiettivo migliorare la policy: π' : $R^{\pi'}{}_t > R^{\pi}{}_t$

The RL updated picture



a_t, dipende dalla situazione!

Gli elementi del RL

Environment. Fornisce il reward (istantaneo), fornisce lo stato dell'ambiente. Reagisce alle azioni (output) dell'agente.

Agente. Ragiona sullo stato fornito dall'ambiente e sul reward istantaneo per produrre un'azione adeguata sull'ambiente.

Goal. Obbiettivo che deve raggiungere l'agente. Si può aggiungere che l'agente deve raggiungere l'obbiettivo con una policy ottima.

Policy. Descrive l'azione scelta dall'agente: mapping tra stato (output dell'ambiente) e azioni dell'agente. Funzione di controllo. Le policy possono avere una componente stocastica. Viene utilizzata una modalità adeguata per rappresentare il comportamento dell'agente (e.g. tabella, funzione continua parametrica…).

Reward. Ricompensa **immediata**. Associata all'azione intrapresa in un certo stato. Può essere data al raggiungimento di un goal (esempio: successo / fallimento). E' uno scalare. Rinforzo primario, solitamente ***qualitativo***.

Value. Ricompensa a **lungo termine**. Somma dei reward: costi associati alle azioni scelte istante per istante + costo associato allo stato finale. ***Orizzonte temporale ampio.*** Rinforzo secondario. Vogliamo realizzare agenti che **ragionino in modo strategico**. (cf. "Cost-to-go" in ricerca operativa). Valore atteso della somma dei reward associati al comportamento.

Osservazioni

Formulazione generale che si adatta ad una grande quantità di problemi.

Agente = Controllore.

Tempo = tempo, ma anche stadio della decisione, del planning….

Azione = forza, voltaggio, decisioni…..

Stato = situazione = misura di grandezze fisiche, di grandezze interne, stato mentale,….

Pre-processing di misure fisiche. E' importante per un efficiente RL.

Policy = definisce quale azione in un certo stato (può essere stocastica, e.g. ε-greedy)

Ambiente = **tutto quanto non è modificabile direttamente dall'agente**. Può essere noto o meno.

Reward (rinforzo primario) = viene generato all'esterno dell'agente.

Value function (rinforzo secondario) = viene stimata all'interno dell'agente.

Reward e Obbiettivi

Il reward è "esterno" all'agente.

Massimizzare la ricompensa a **lungo termine**, Value, cumulando le ricompense istantanee: $r_t(a(t), s(t), s(t+1)) \in R$.

Defininendo una ricompensa che viene massimizzata solamente quando il goal viene raggiunto, possiamo ottenere che l'agente impari il task (raggiunga il goal).

Collegamento tra reward e goal.

Il reward consente di comunicare COSA si vuole ottenere; nulla è detto sul COME.

Proprietà del rinforzo

L'ambiente o l'interazione può essere complessa.

Il rinforzo può avvenire dopo una sequenza più o meno lunga di interazioni (azione-> cambiamento di stato e reward istantaneo): delayed reward.

E.g. Agente -- giocatore di scacchi
Ambiente – avversario

Problemi collegati:
temporal credit assignement (quando ho sbagliato la mossa?)
structural credit assignement (quale mossa era sbagliata?)

L'apprendimento non è per esempi, ma dall'osservazione dell'impatto sull'ambiente del comportamento dell'agente.

Meccanismo di apprendimento nel RL

Inizializzazione: se l'agente non agisce sull'ambiente non succede nulla. Occorre specifiare una policy iniziale.

Ciclo dell'agente (le tre fasi sono sequenziali):

1) Implemento una policy ($\pi(s,a)$)
2) Stimo la Value function ($Q^{\pi}(s,a)$) per tutte le coppie stato-azione
3) Miglioro la policy.

Framework analizzati:

1. Stocastico completo. L'agente conosce la statistica della dinamica dell'ambiente e dei reward, scrive le equazioni che legano le value function in stati diversi e calcola i valori di Q(.).
2. Stocastico con aggiornamento. L'agente conosce la statistica della dinamica dell'ambiente e dei reward. Procede da uno stato iniziale e da qui esplora in parallelo tutte le possibilità e aggiorna i valori delle Q(.).
3. Stocastico con interazione singola con la scelta di una singola azione. L'agente **non** conosce la statistica della dinamica dell'ambiente e dei reward. Procede da uno stato iniziale e da qui esplora una sola azione e un solo stato prossimo. Aggiorna i valori di Q(.).

Sommario

Il Reinforcement Learning.

La value function: ricompensa a lungo termine: formulazione ricorsiva.

Esempi di calcolo

Esempio: AIBO search

Azioni:
1) Rimanere fermo e aspettare che qualcuno getti nel cestino una lattina vuota.
2) Muoversi attivamente in cerca di lattine.
3) Tornare alla sua base (recharge station) e ricaricarsi.

Stato:
1) Alto livello di energia.
2) Basso livello di energia.

Azioni ammissibili (policy):
a(s = high) = {Search, Wait}
a(s = low) = {Search, Wait, Recharge}

Goal: collezionare il maggior numero di lattine.

Funzionamento del Robot

$$P_{s \to s'|a} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$$

Funzione Stato prossimo se il livello di energia è alto (s_t = alto):

1) se scelgo a = Wait – s_{t+1} = alto.

$$P_{high \to high|wait} = Pr\{s_{t+1} = high | s_t = high, a_t = wait\} = 1$$

2) se scelgo a = Search, s_{t+1} avrà una certa probabilità α di rimanere nello stato high.

$$P_{high \to high|search} = Pr\{s_{t+1} = high | s_t = high, a = search\} = \alpha = 0.4$$

$$P_{high \to low|search} = Pr\{s_{t+1} = low | s_t = high, a = search\} = 1 - \alpha = 0.6$$

Funzionamento del Robot

$$P_{s \to s'|a} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$$

Funzione Stato prossimo se il livello di energia è basso (s_t = basso):

1) se scelgo a = Wait – s_{t+1} = basso.

$$P_{low \to low|wait} = Pr\{s_{t+1} = low | s_t = low, a_t = wait\} = 1$$

2) se scelgo a = Search, s_{t+1} avrà una certa probabilità β di rimanere low.

$$P_{low \to low|searh} = Pr\{s_{t+1} = low | s_t = low, a_t = search\} = \beta$$

$$P_{low \to high|searh} = Pr\{s_{t+1} = high | s_t = low, a_t = search\} = 1-\beta$$

Il robot "muore" e viene portato a ricarcarsi.

3) se scelgo a = Recharge, s_{t+1} sarà carico: high.

$$P_{low \to high|recharge} = Pr\{s_{t+1} = high | s_t = low, a_t = recharge\} = 1$$

Reward del Robot

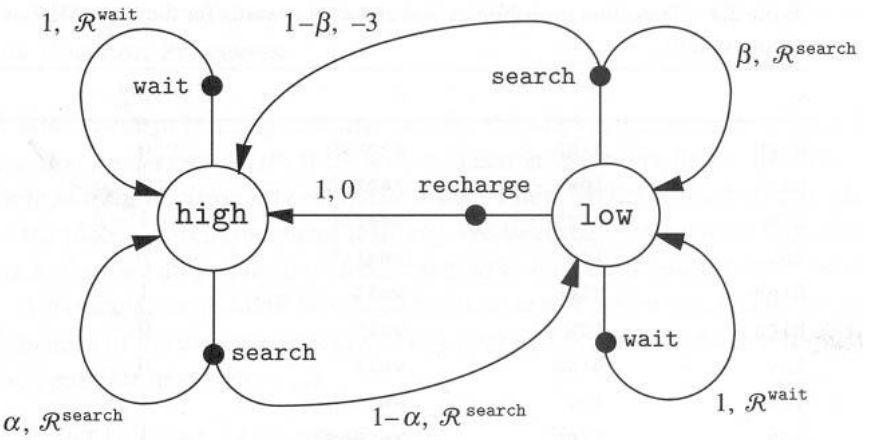
$$R_{s \to s'|a} = Pr\{r_{t+1} = r | s_t = s, a_t = a, s_{t+1} = s'\}$$

Funzione Reward:

R^{search} reward se il robot sta cercando	g(low) = search; g(high) = search:
R^{wait} reward se il robot sta cercando.	g(low) = wait; g(high) = wait;
-3 se occcorre portarlo a ricaricarsi.	g(low) = search; and s_{t+1} = high;
0 se il robot va autonomamente a ricaricarsi.	g(low) = recharge;

$R^{search} > R^{wait}$.

State Transition Graph



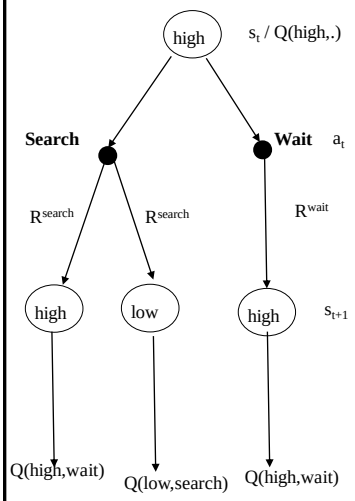
Graphical model

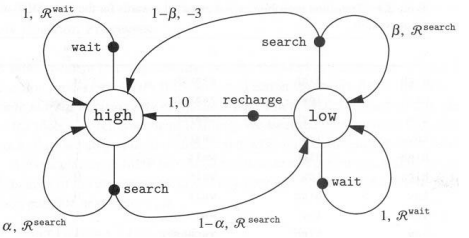
State Transition Table

s	a	s'	$P_{s->s'|a}$	$R_{a(s)->a'}$
alta	ricerca	alta	α	R^{search}
alta	ricerca	bassa	1 - α	R^{search}
bassa	ricerca	alta	1 - β	-3
bassa	ricerca	bassa	β	R^{search}
alta	attesa	alta	1	R^{wait}
alta	attesa	bassa	0	R^{wait}
bassa	attesa	alta	0	R^{wait}
bassa	attesa	bassa	1	R^{wait}
bassa	ricarica	alta	1	0
bassa	ricarica	bassa	0	0
alta	ricarica	Non esiste	X	X

L'albero delle decisioni

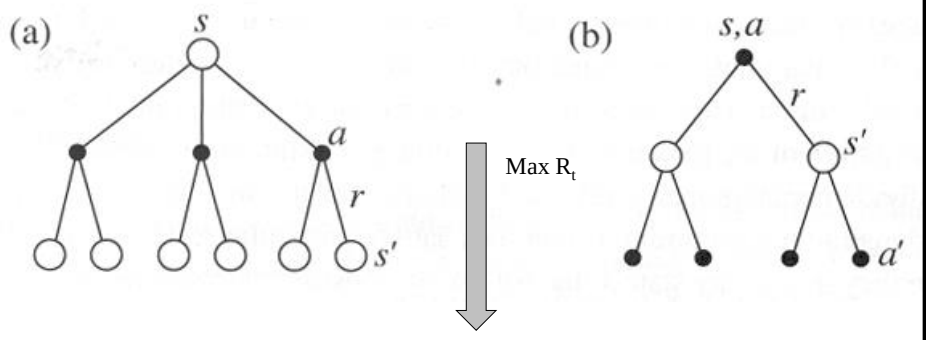
Apro il grafo





Guardo la sequenza s -> a -> s' in 1 passo di interazione

Policy



La policy deve essere ancora determinata. Come fa l'agente a determinare la policy ottimale?

Archi multipli fuoriuscenti da un'azione sono associati alla probabilità di scegliere quel cammino (ambiente stocastico).

Archi multipli fuoriuscenti da uno stato, sono associati alla policy.

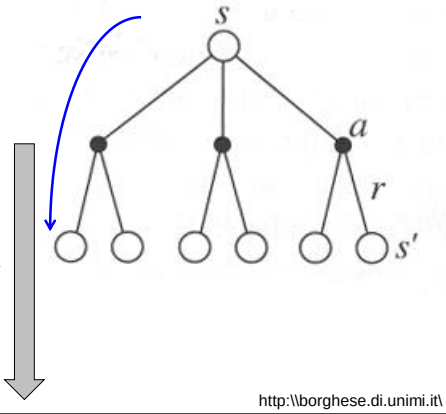
Value function & policy

Nulla è detto sulla policy: dato uno stato, in quale nodo azione mi sposto?

Vogliamo costruire agenti lungimiranti.

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

Value function on state-action



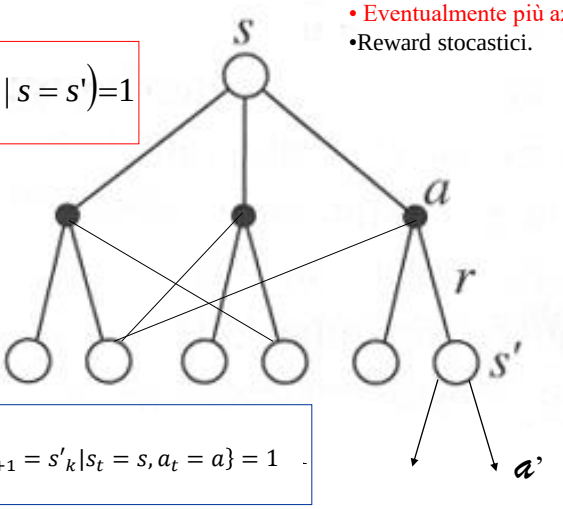
Massimizzo la ricompensa a lungo termine, $Q^{\pi}(.)$. Dipende dalla policy π.

Value function e modelli markoviani

Anche la policy può essere stocastica.

Per ogni coppia stato-azione devo valutare:
- Più stati prossimi
- Eventualmente più azioni in s'
- Reward stocastici.

$$\sum_{j=1}^{N\_azioni} \Pr\left(a'_j \,|\, s = s'\right) = 1$$



$$\sum_{k=1}^{N\_stati} Pr\{s_{t+1} = s'_k | s_t = s, a_t = a\} = 1$$

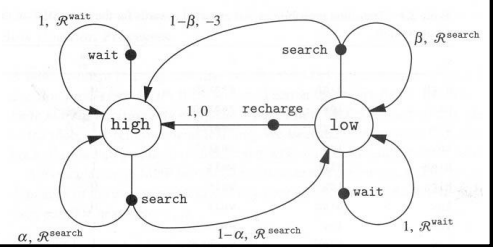
Il nostro filo logico

La Value function ci serve per decidere l'azione migliore.
Per calcolare la Value function devo collezionare reward futuro.

Come se ne esce?

Determinazione algebrica della Value Function
Determinazione "esplorativa" della Value Function

Doteremo l'agente di un'intelligenza adatta a effettuare l'esplorazione.

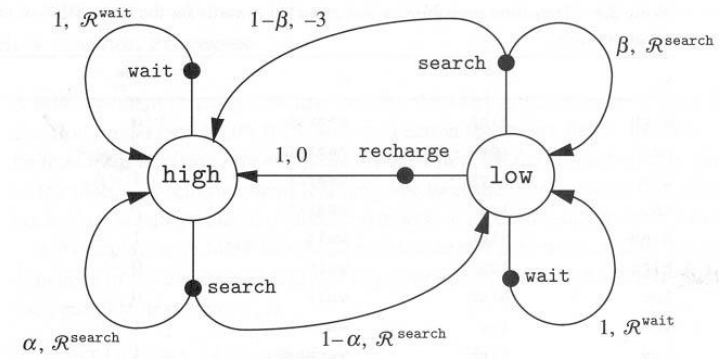


Confronto con il setting non associativo

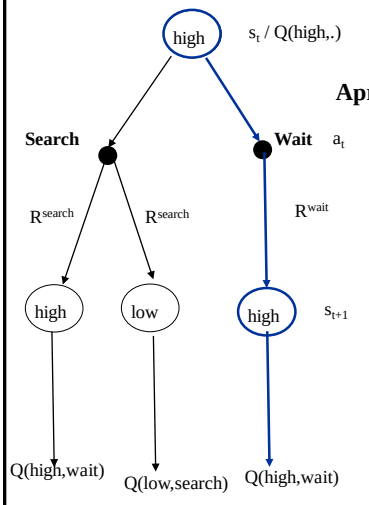
	Setting non associativo	Setting associativo
Task	Azioni	Comportamenti (catena di azioni)
Reward	Reward istantaneo	Reward istantaneo (= 0 per certi passi di iterazione)
Max	Reward atteso sulla singola azione	Reward atteso sulla singola azione, dato un certo comportamento. Somma (scontata) dei reward collezionati lungo il task.
Orizzonte temporale del task	Finito (1 azione)	Finito / infinito per il singolo task
Policy	Stocastica	Stocastica
Stato	Non definito	Markoviano

Esempio di calcolo della Value function in s = high

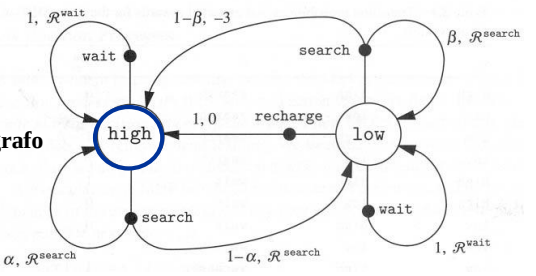
Value function
Q(high, wait) = ?
Q(high, search) = ?



Agent waits in high state



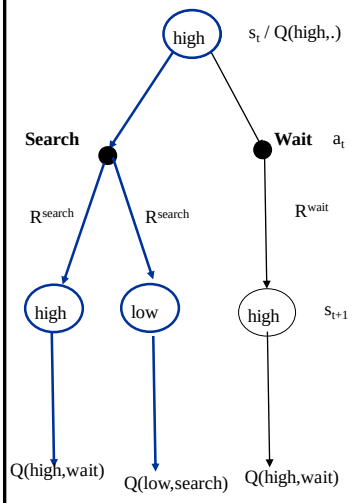
Apro il grafo



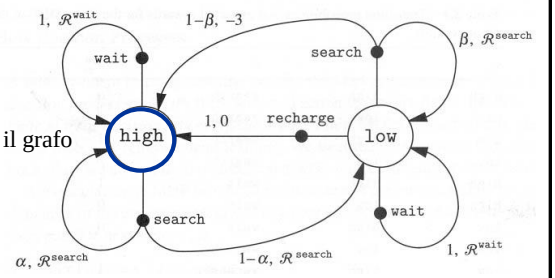
α=0.4, β=0.1, γ=0.8,
R^{search}=3, R^{wait}=1 R^{dead} = -3, R^{auto} = 0

$Q(high,wait) = [R^{wait} + \gamma\, Q(high,wait)] =$
$[1 + 0.8\, Q(high,wait)] = 5$

Agent searches in high state



Apro il grafo



α=0.4, β=0.1, γ=0.8,
R^{search}=3, R^{wait}=1 R^{dead} = -3, R^{auto} = 0

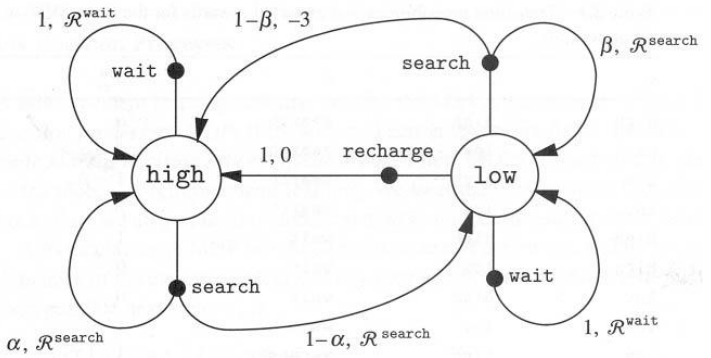
Q(high,search) = α [R^{search} + γ Q(high,search)] +
(1-α) (R^{search} + γ Q(low,search)] =

Q(high,search) = 0.4 [3 + 0.8 Q(high,search)] +
0.6 [3 + 0.8 Q(low,search)]

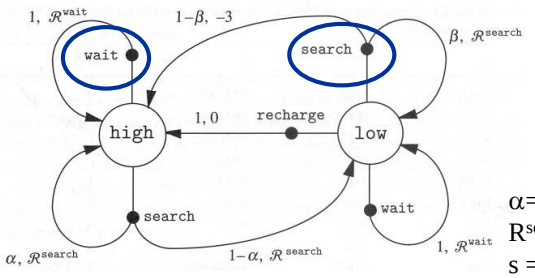
Esempio di calcolo della Value function in s = high

Policy deterministica
a(high) = wait
a(low) = search

Value function
Q(high, wait) = ?
Q(low, search) = ?



Policy deterministica



α=0.4, β=0.1, γ=0.8,
R^{search}=3, R^{wait}=1 R^{dead} = -3, R^{auto} = 0
s = High - a = Wait;
s = Low - a = Search;

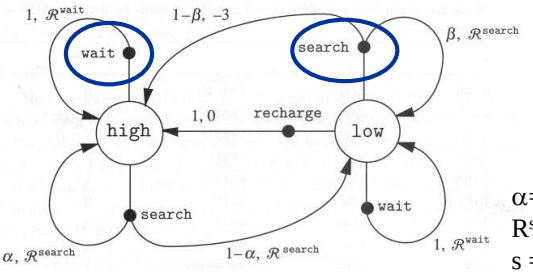
$Q(h,w) = [R^{wait}+\gamma\ Q(h,w)] = [1+0.8\ Q(h,w)]$
Wait

$Q(l,s) = \beta x[R^{search}+\gamma\ Q(l,s)]+(1-\beta)x[R^{dead}+\gamma\ Q(h,w)] = 0.1[3+0.8Q(l,s)]+0.9[-3+0.8Q(h,w)]$

Search -> Low *Search -> High*

Sistema lineare di 2 equazioni nelle 2 incognite: Q(h,w) e Q(l,s)
Come calcolo Q(h,w) e Q(l,s)?

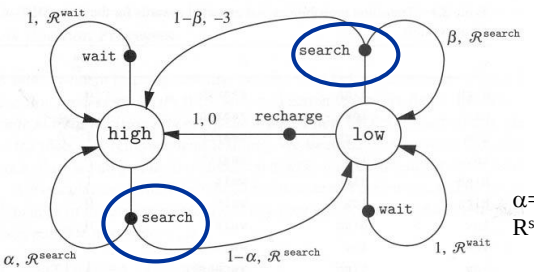
Policy deterministica - I



α=0.4, β=0.1, γ=0.8,
R^{search}=3, R^{wait}=1 R^{dead} = -3, R^{auto} = 0
s = High - a = Wait;
s = Low - a = Search;

$Q(h,w) = [1+0.8\ Q(h,w)]$ → $Q(h,w) = 5$
$Q(l,s) = 0.1x[3+0.8xQ(l,s)]+0.9x[-3+0.8\ Q(h,w)]$
↓
$Q(l,s) = 0.3 + 0.08\ Q(l,s) -2.7 +0.72\ Q(h,w)$ →
$Q(l,s)\ (1-0.08) = 0.3-2.7+3.6$ → $Q(l,s) = 1,304$

Policy deterministica - II



α=0.4, β=0.1, γ=0.8,
R^{search}=3, R^{wait}=1 R^{dead} = -3, R^{auto} = 0

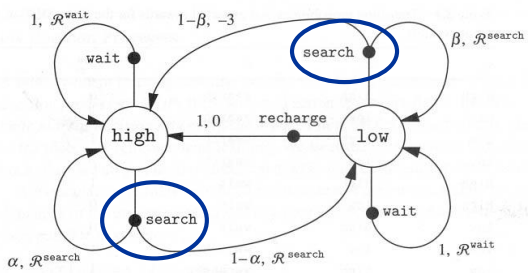
s = High - **a = Search;**
s = Low - a = Search;

Q(h,s) = 0.4x [3+0.8 Q(h,s)] + 0.6 x[3+0.8 x [Q(l,s)]

Q(l,s) = 0.1x[3+0.8xQ(l,s)]+0.9x[-3+0.8 Q(h,s)]

Sistema lineare di 2 equazioni nelle 2 incognite: Q(h,s) e Q(l,s)

Policy deterministica - II



α=0.4, β=0.1, γ=0.8,
R^{search}=3, R^{wait}=1 R^{dead} = -3, R^{auto} = 0

s = High - **a = Search;**
s = Low - a = Search;

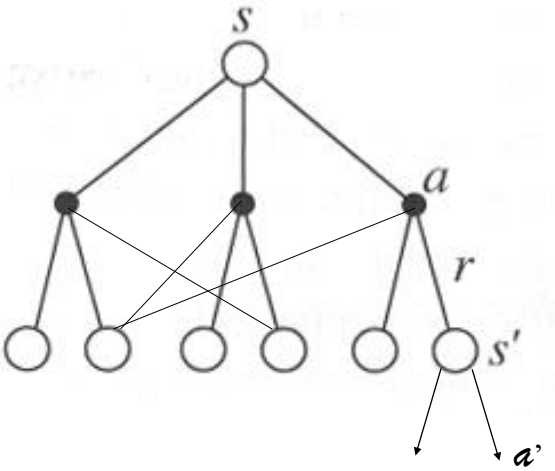
Q(h,s)[1-0.32] = 1.2 + 1.8 + 0.48 x Q(l,s)
Q(l,s)[1-0.08] = 0.3-2.7+ 0.72 Q(h,s)]

Q(h,s) [0.68] - 0.48 Q(l,s) = 3. → Q(h,s) = 5.7429
Q(l,s) [0.92] -0.72 Q(h,s) = -2.4 → Q(l,s) = 1.8857

La policy è migliore per entrambe le coppie stato-azione.

Valutazione policy stocastica

Nel valutare Q(s,a) dobbiamo valutare tutti i cammini che partono da ogni s'.



Osservazioni

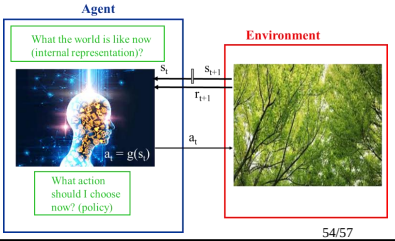
Posso migliorare l'azione in uno stato (e.g. nello stato High):

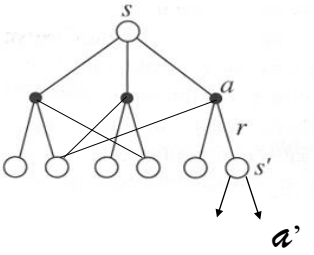
s = High - a = Wait; Q(h,w) = 5 a = Search Q(h,s) = 5.7429
s = Low - a = Search; Q(l,s) = 1,304

Questo ha impatto anche sugli altri stati: Q(l,s) = 1.8857

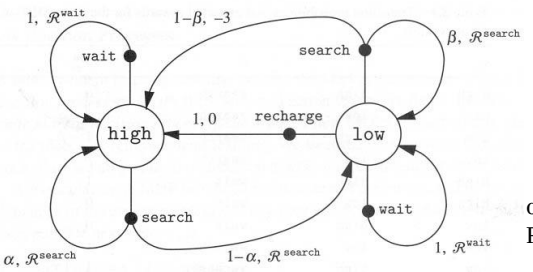
E' un sistema interconnesso in cui l'azione in uno stato si ripercuote a valle, anche sugli stati futuri (e le azioni scelte da lì in poi).

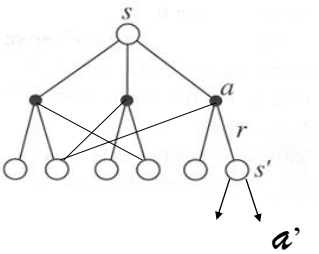
La scelta di un'azione o di un'altra, cambia la funzione Q per le coppie stato-azione future.





Policy stocastica





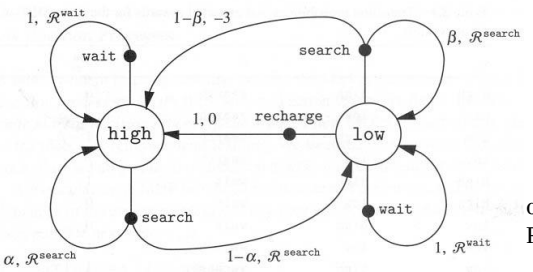
α=0.4, β=0.1, γ=0.8,
R^{search}=3, R^{wait}=1 R^{dead} = -3, R^{auto} = 0

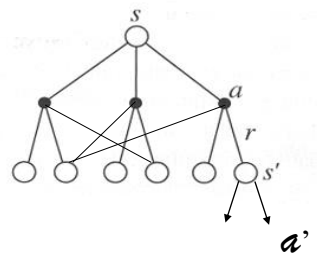
Q(high,wait) = 1 x {R^{wait}+γ [Pr(a'=search) **Q(high,search)**+ Pr(a'=wait) **Q(high,wait)**]} =
1x {1+0.8[Pr(a=search) **Q(high,search)** + Pr(a=wait) **Q(high,wait**)]}

Q(high,search) = α x {R^{search}+γ [(Pr(a'=search) **Q(high,search)**+ Pr(a'=wait) **Q(high,wait)**]} +
(1- α) x {R^{search}+γ [Pr(a'=search) **Q(low,search)**+ Pr(a'=wait) **Q(low,wait)** +
Pr(a' = recharge) **Q(low,recharge)]}**

Come calcolo Q(high,*)?
Quale azione, a scelgo, in s = high?

Policy stocastica





α=0.4, β=0.1, γ=0.8,
R^{search}=3, R^{wait}=1 R^{dead} = -3, R^{auto} = 0

Q(low,wait) = 1 x {R^{wait}+γ [Pr(a'=search) **Q(low,search)**+ Pr(a'=wait) **Q(low,wait)** +
Pr(a'=recharge) **Q(low,reharge)**]}

Q(low,search) = β x {R^{search}+γ [(Pr(a'=search) **Q(high,search)**+ Pr(a'=wait) **Q(high,wait)+**
Pr(a'=recharge) **Q(low,reharge)**]} +
(1-β) x {$R^{recharge}$+γ [Pr(a'=search) **Q(high,search)**+ Pr(a'=wait) **Q(high,wait)]}**

Q(low,recharge) = 1 x {R^{auto}+γ [(Pr(a'=search) **Q(high,search)**+ Pr(a'=wait) **Q(high,wait)**]}
33

Sommario

Il Reinforcement Learning.

Processi Markoviani.

Esempi di calcolo