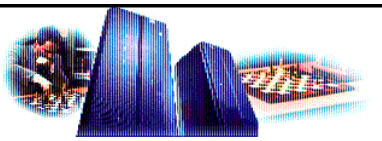
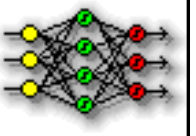




Sistemi Intelligenti
Reinforcement Learning:
Processi Markoviani e Value function

Alberto Borghese

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

Sommario

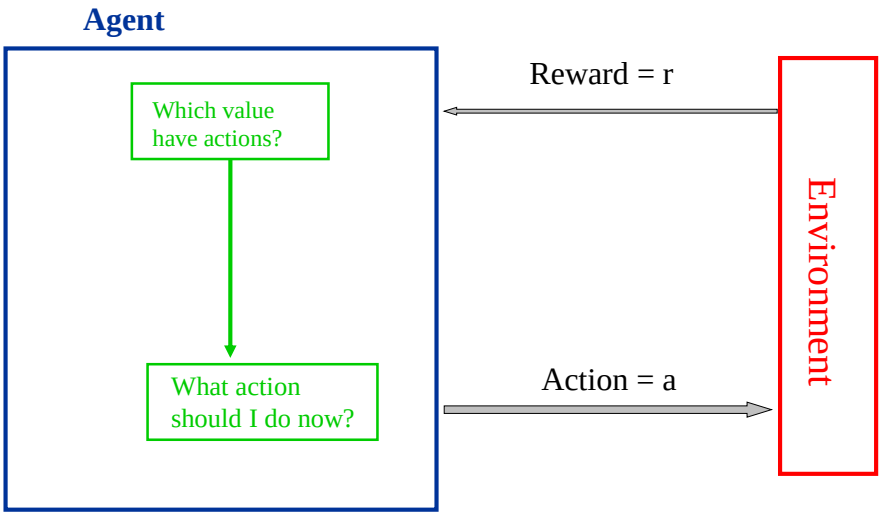
Il Reinforcement Learning in setting non associativi

Processi Markoviani.

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

Esempi di calcolo

Schematic diagram of an agent

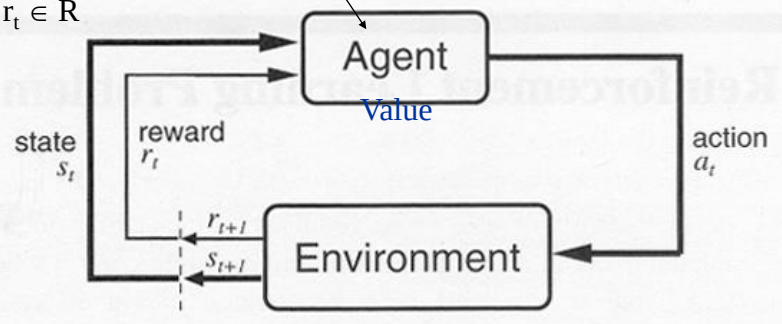


Interazione Agente-Ambiente

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

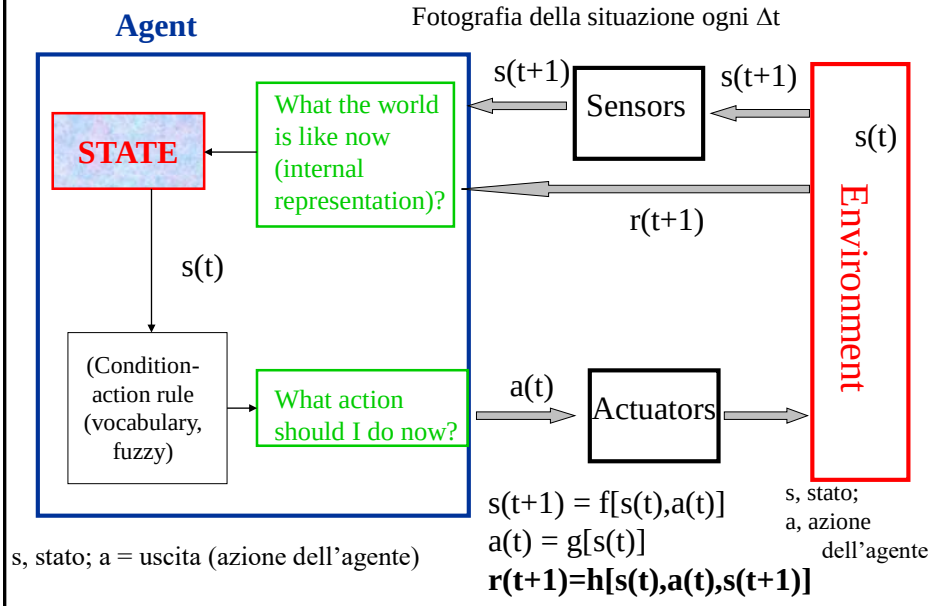
Goal

Policy stocastica dell'agente: $\pi(s, a)$



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

Enhanced schematic diagram of an agent



Gli elementi del RL

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 pe rappresentare il comportamento dell'agente (e.g. tabella, funzione continua parametrica…).

Reward function. 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 function. "Cost-to-go". 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.

Environment. Fornisce la reward function, fornisce l'input in base al quale l'agente aggiorna lo stato. Reagisce agli output dell'agente.

Meccanismo di apprendimento nel RL

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

1) Implemento una policy
2) Aggiorno la Value function
3) Aggiorno la policy.

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 = viene generato all'esterno dell'agente.
Value = viene stimata all'interno dell'agente.

Caratteristiche dello stato

Policy di un agente: $\pi(s, a)$.

Caratteristiche dello stato, s:
• Contiene gli stimoli pre-elaborati a partire dagli stimoli semplici misurati sull'ambiente.
• Gli stimoli pre-elaborati analizzano una sequenza temporale di stimoli semplici.
• Lo stato deve potere essere misurato dall'agente.
• Lo stato deve essere efficiente per il raggiungimento del goal.

Come rappresento s? Memorizzo la sequenza temporale degli stimoli semplici di interesse? Introduzione del concetto di stato.

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.

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: $\pi(s, a)$, la ricompensa sarà una funzione dei reward negli istanti di tempo successivi a t, ad esempio:

$$R^{\pi}{}_t = r_{t+1} + r_{t+2} + r_{t+3} + \ldots\ldots r_T$$

Terminal State

Quando è adeguata?

Problemi ad orizzonte finito (episodic tasks, a terminal state is defined).

Problemi stazionari.

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

Infinite horizon problems (continuing tasks)

Il concetto fondamentale è il "**discount**".

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".
Present value of future rewards.

$$R_t \to \frac{r}{1-\gamma} \quad if \quad r_t = r_{t+k} \quad \forall k$$

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

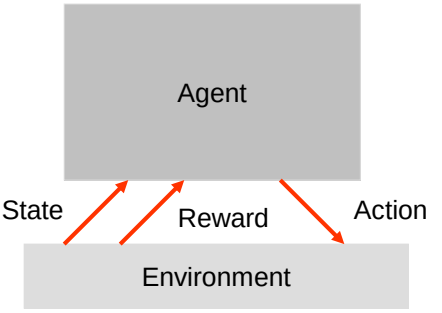
Cosa succede se $\gamma \to 0$ *e* $\gamma \to 1$?

Reinforcement Learning Problem

Given: Repeatedly…
- Executed action
- Observed state
- Observed reward

Learn action policy $\pi: S \rightarrow A$
- Maximizes life reward
  $r_0 + \gamma r_1 + \gamma^2 r_2 \ldots$
  from any start state.
- Discount: $0 < \gamma < 1$

Note:
- Unsupervised learning
- Delayed reward
- $R_t = R(s_t)$



$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} s_3$

Goal: Learn to choose actions that maximize life reward

$r_0 + \gamma r_1 + \gamma^2 r_2 \ldots$

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
Determinazione della Value Function e scelta della policy via via sempre più intrecciate tra loro.

Sommario

Il Reinforcement Learning.

Processi Markoviani.

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

Esempi di calcolo

Environment Markoviano

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.

Formalizziamo. Supponiamo s ed r variabili discrete appartenenti ad un insieme finito di valori.

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

Se lo stato è Markoviano:

$Pr\{s_{t+1} = s' \mid s_t, a_t\}$

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

Reinforcement 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
Azione che dall'esterno viene imposta all'ambiente

Approssimazione

L'ipotesi di ambiente e rinforzo Markoviani possono essere soddisfatte in modo approssimato.

Environment – cart + pole.

Stato – Posizione e velocità di cart e di pole, attuatore che trasforma la corrente per il motore in forza di traslazione.

Agente – Controllore del motore che pilata la traslazione (corrente).

Reward – Cade / non_cade - 0 /1.

Value Function ?

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".

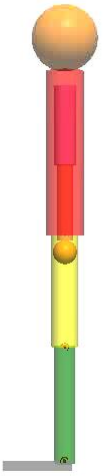
Nevertheless, we learn upright posture in the very first year of our life.

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



Sommario

Il Reinforcement Learning.

Processi Markoviani.

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:

A(s = high) = {Search, Wait}
A(s = low) = {Search, Wait, Recharge}

Goal: collezionare il maggior numero di lattine.

Funzionamento del Robot

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

Se il livello di energia è alto (s_t = alto):
se scelgo Wait – s_{t+1} = alto.
se scelgo Search, s_{t+1} avrà una certa probabilità di diventare low.

$$P_{high->low|Search} = \Pr\{s_{t+1} = low | s_t = high, a_t = Search\} = \alpha$$

Se il livello di energia è basso (s_t = basso):
se scelgo Wait – s_{t+1} = basso.
se scelgo Recharge - s_{t+1} = alto.
se scelgo Search, s_{t+1} avrà una certa probabilità di fermarsi.

$$P_{low->low|Search} = \Pr\{s_{t+1} = low | s_t = low, a_t = Search\} = \beta$$

Reward del Robot

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

Funzione Reward:

R^{search} reward se il robot sta cercando.

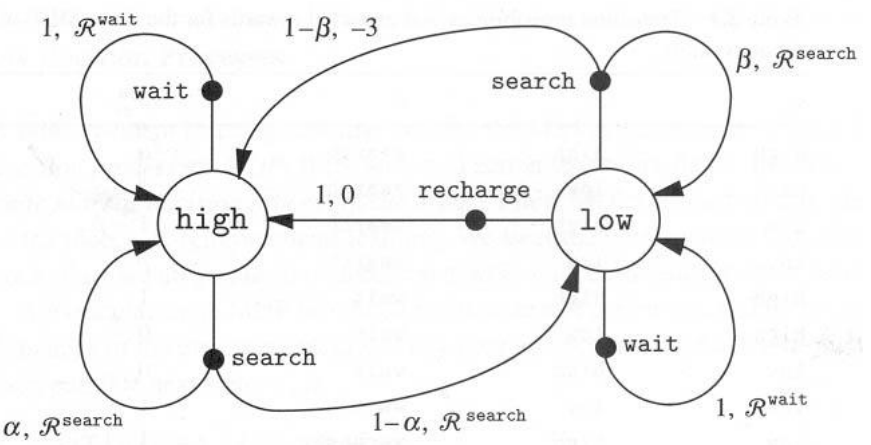
R^{wait} reward se il robot sta cercando.

-3 se occcorre portarlo a ricaricarsi.

0 se il robot va autonomamente a ricaricarsi.

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

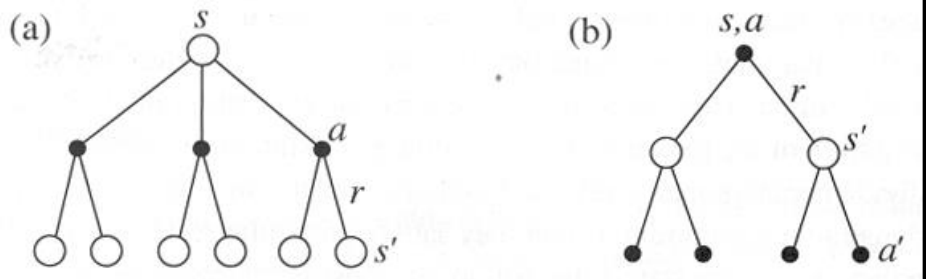
State Transition Graph



State Transition Table

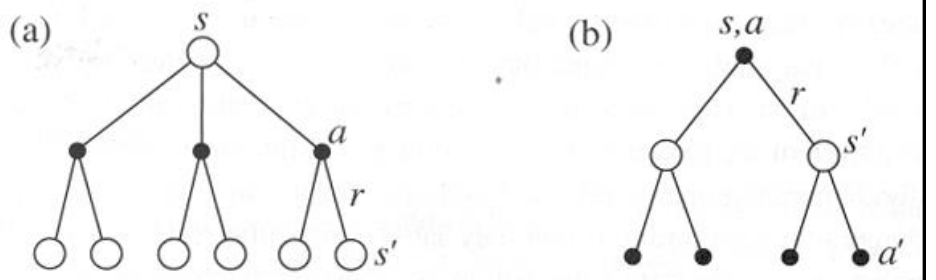
s	a	s'	$P_{s->s'\|a}$	$R_{a(s)->a'}$
alta	ricerca	alta	α	R^{search}
alta	ricerca	bassa	$1 - \alpha$	R^{search}
bassa	ricerca	alta	$1 - \beta$	-3
bassa	ricerca	bassa	β	R^{search}
alta	attesa	alta	1	R^{wait}
alta	attesa	bassa	0	R^{wait}
bassa	attesa	alta	0	R^{wait}
bassa	attesa	bassa	1	R^{wait}
bassa	ricarica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

L'albero delle decisioni



Guardo la sequenza s -> a -> s' -> a' 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.

Confronto con il setting non associativo

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

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

Determinazione della Value Function e scelta della policy via via sempre più intrecciate tra loro.

Sommario

Il Reinforcement Learning.

Processi Markoviani.

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

Esempi di calcolo

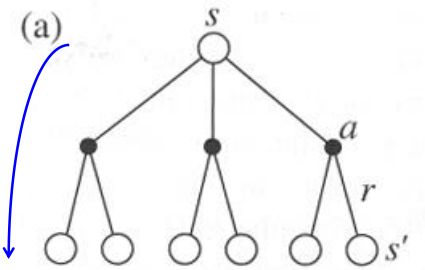
Value function & policy

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

Vogliamo costruire agenti lungimiranti.

State-Value function
(function of the policy):

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} =$$
$$E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s\right\}$$



Massimizzo la ricompensa a lungo termine, V(.). Dipende dalla policy π.

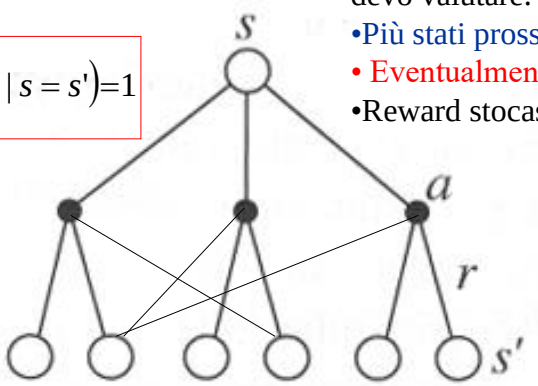
Value function e modelli markoviani

Anche la policy può essere stocastica.

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

Per ogni coppia stato-azione devo valutare:

- Più stati prossimi
- Eventualmente più azioni in s'
- Reward stocastici.

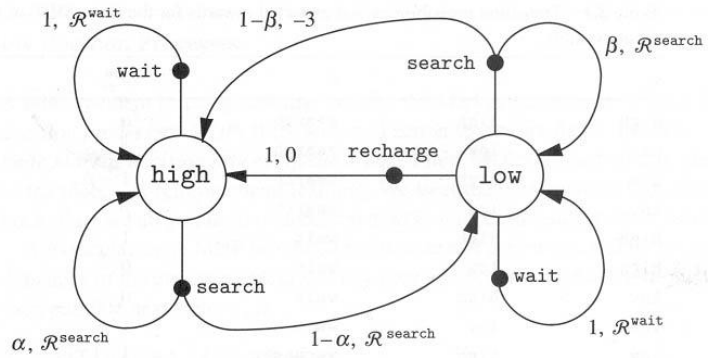


$$\sum_{k=1}^{N\_stati} \Pr\left(s_{t+1} = s_k \mid s_t = s'; a_t = a_j\right) = 1$$

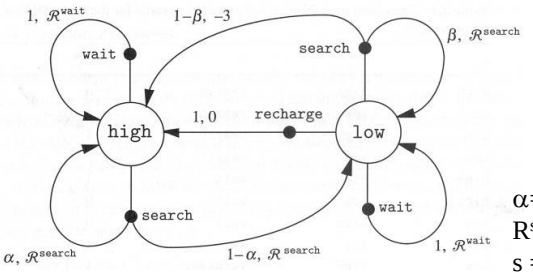
Esempio di calcolo della Value function

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

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



Policy deterministica - I



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

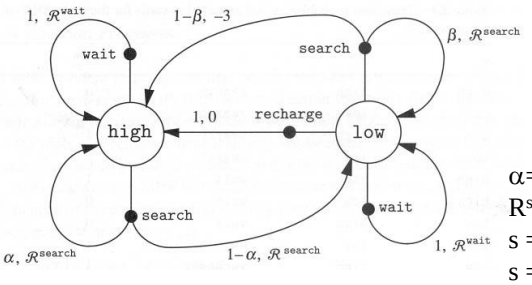
$V(h) = [1+0.8\ V(h)]$
Wait

$V(l) = 0.1x[3+0.8xV(l)]+0.9x[-3+0.8V(h)]$

Search -> Low *Search -> High*

Sistema lineare di 2 equazioni nelle 2 incognite: V(h) e V(l)
Come calcolo V(h) e V(l)?

Policy deterministica - I

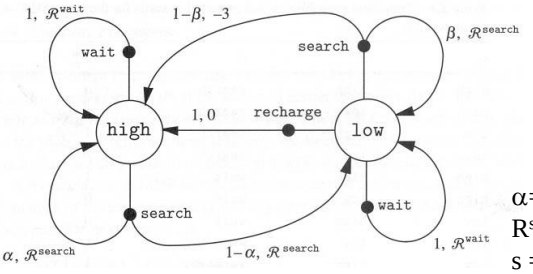


α=0.4, β=0.1, γ=0.8,
R^{search}=3, R^{wait}=1 $R^{recharge}$ = -3, R^{auto} = 0
s = High - a = Wait;
s = Low - a = Search;

V(h) = [1+0.8 V(h)] → V(h) = 5
V(l) = 0.1x[3+0.8xV(l)]+0.9x[-3+0.8 V(h)]

V(l) = 0.3 + 0.08 V(l) -2.7 +0.72 V(h) →
V(l) (1-0.08) = 0.3-2.7+3.6 → V(l) = 1,304

Policy deterministica - II



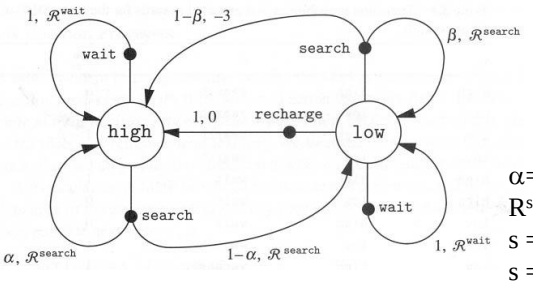
α=0.4, β=0.1, γ=0.8,
R^{search}=3, R^{wait}=1 $R^{recharge}$ = -3, R^{auto} = 0
s = High **- a = Search;**
s = Low - a = Search;

V(h) = 0.4x [3+0.8 V(h)] + 0.6 x[3+0.8 x [V(l)]

V(l) = 0.1x[3+0.8xV(l)]+0.9x[-3+0.8 V(h)]

Sistema lineare di 2 equazioni nelle 2 incognite: V(h) e V(l)

Policy deterministica - II



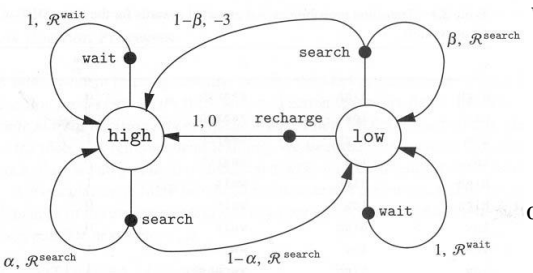
α=0.4, β=0.1, γ=0.8,
R^{search}=3, R^{wait}=1 $R^{recharge}$ = -3, R^{auto} = 0
s = High - **a = Search;**
s = Low - a = Search;

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

V(h)[0.68] - 0.48 V(l) = 3. -> V(h) = 5.7429
V(l)[0.92] -0.72 V(h) = -2.4 -> V(l) = 1.8857

La policy è migliore per entrambi gli stati.

Policy stocastica



Wait

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

V(h) = Pr(a=W) x 1 x [1+0.8V(h)] + Pr(a=S) x {0.4 x [3+0.8 x V(h)] +
0.6 x [3+0.8 x V(l)]

V(l) = Pr(a=W) x 1 x [1+0.8 V(l)] + Pr(a=S) x 0.1 x [3+0.8xV(l)] +
Pr(a=S) x 0.9 x [-3+0.8 x V(h)] + Pr(a=R) x 1 x [0+0.8xV(h)]

Come calcolo Vh e Vl?

Sommario

Il Reinforcement Learning.

Processi Markoviani.

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

Esempi di calcolo