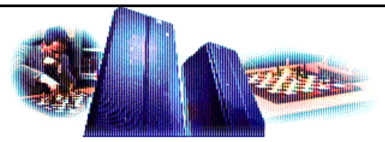
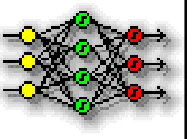




# Sistemi Intelligenti Reinforcement Learning: Iterative policy evaluation

Alberto Borghese

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



# Sommario

Le equazioni di Bellman

Stima iterativa della funzione valore

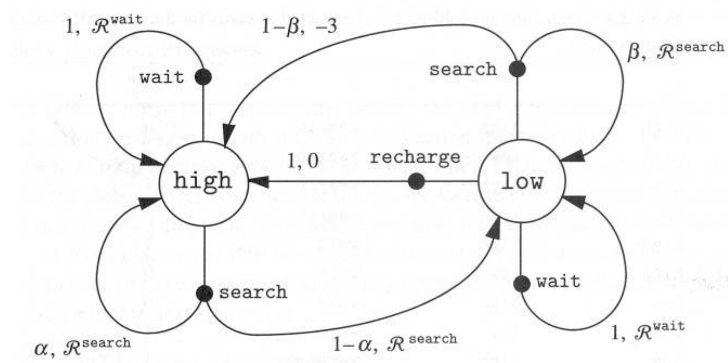
Policy iteration

Esempi

# Esempio di calcolo della Value function

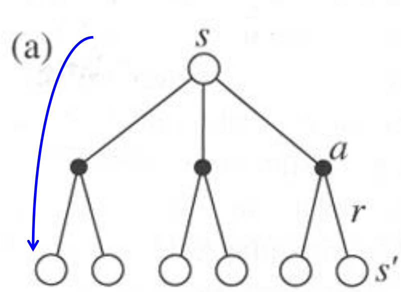
Value function
V(high) = ?
V(low) = ?

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






# Analisi ad un passo




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

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

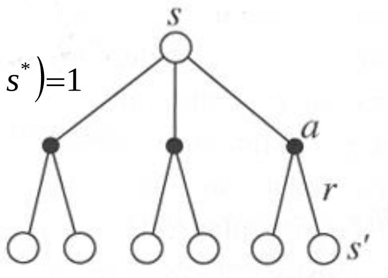
Archi multipli fuoriuscenti da un'azione sono associati alla probabilità di scegliere quel cammino (ambiente stocastico).

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

# Value function e modelli markoviani

Anche la policy può essere stocastica.

$$\sum_{j=1}^{N\_azioni} \Pr\left(a_j \mid s = s^*\right) = 1$$




L'azione scelta in s può essere scelta in modo stocastico (e.g. ε-greedy policy)

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

Per ogni stato devo valutare:
- Più azioni
- Per goni azione, più stati prossimi
- Reward stocastici.



# Il modello markoviano

Il comportamento dell'ambiente è definito dallo stato: S = $\{s_j\}$
Per ogni stato l'agente sceglie un'azione: a = a(s) A = $\{a_k\}$
**Policy di un agente: π(s, a) è quanto può definire (e ottimizzare) l'agente.**

L'ambiente ha una evoluzione stocastica rappresentata da un MDP:

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

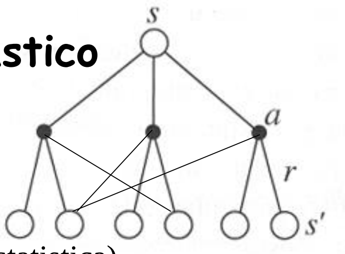
Inoltre, ad ogni istante fornisce un reward immediato associato alla transizione, stimato all'istante t come:

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

$$\forall \mathbf{s} \in \mathbf{S};\ \forall \mathbf{a} \in \mathbf{A}$$

# Reward stocastico

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




Reward stocastico (da una distribuzione statistica)
È un valore condizionato a s, a, s' e vale:

Pr(reward = r' | s, a, s') = Pr(reward = r' | s') * Pr(a | s) * Pr(s' | s,a)

Questa è la probabilità condizionata a: stato prossimo, azione e reward.

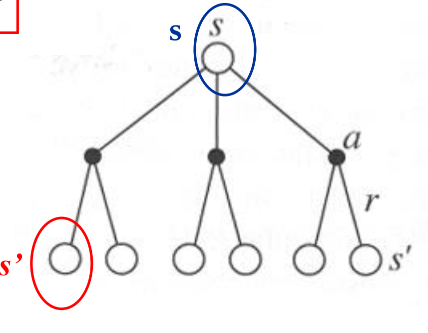


# Calcolo ricorsivo della Value function

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

$$\boxed{V^{\pi}(s') = E_{\pi}\{R_{t+1} \mid s_{t+1}\}}$$

Relazione tra $V^{\pi}(s)$ e $V^{\pi}(s')$?



## Calcolo ricorsivo della Value function

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

Isolo il reward ad un passo nella serie dei reward.

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

$$V^{\pi}(s_t) = E_{\pi}\left\{\left(r_{t+1} + \sum_{k=0}^{\infty}\gamma^{k+1} r_{t+k+2}\right)\middle| s_t = s\right\}$$

Io termine

IIo termine

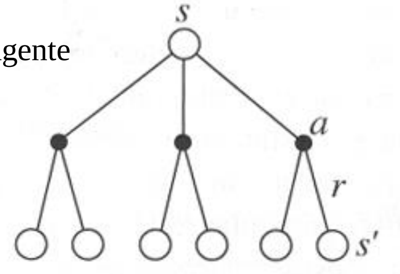


## $V^{\pi}(s)$ : primo termine

$$E_{\pi}\{r_{t+1} \mid s_t = s\} = \sum_{a_j}\pi(s,a)\left[\sum_{s'} P_{s\to s'|a_j} R_{s\to s'|a_j}\right]$$

Agente

Per ogni stato devo valutare:
- Più azioni
- Più stati prossimi
- Reward stocastici nella transizione ad un passo




**Visione Statistica**: Probabilità di ottenere il reward: condizionata all'arrivare nello stato s'.

$$R_{s\to s'|a_j}$$

# $V^{\pi}(s)$ : secondo termine

$$V^{\pi}(s_t)=E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k+1}r_{t+k+2}\middle| s_t=s\right\}=$$

$$\gamma E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+2}\middle| s_t=s\right\}=$$

$$\gamma\sum_{j}\Pr(a_j=a\mid s_t=s)$$

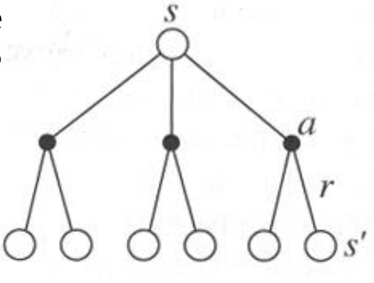
$$\sum_{s'}\left(\Pr(s_{t+1}=s'\mid s_t=s,a_t=a_j)E_{\pi}\left(\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+2}\mid s_{t+1}=s'\right)\right)=$$

$$=\gamma\sum_{j}\Pr(a_j=a\mid s_t=s)\sum_{s'}\left(\Pr(s_{t+1}=s'\mid s_t=s,a_t=a_j)V^{\pi}(s')\right)$$



# $V^{\pi}(s)$ : secondo termine

In (s) confluiranno i reward a lungo termine di tutti gli stati prossimi, s’, ciascuno pesato con la probabilità di passare da s a s’, ovverosia, in termini statistici, condizionati alla realizzazione della transizione di stato, s-> s’ e dai reward a lungo termine, ottenuti scegliendo in s l’azione a.




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

$$\sum_{j}\left(\Pr(a_t=a\mid s_t=s)\right)$$

$$\sum_{s'}\left(\Pr(s_{t+1}=s'\mid s_t=s,a_t=a)\right)\left(\Pr(r_{t+1}=r'\mid s_t=s,a_t=a,s_{t+1}=s')+\gamma V^{\pi}(s')\right)$$

# Calcolo ricorsivo della Value function

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

$$\boxed{V^{\pi}(s') = E_{\pi}\{R_{t+1} \mid s_{t+1} = s'\}}$$

Legame?

Bellman's equation

Policy

Next-state

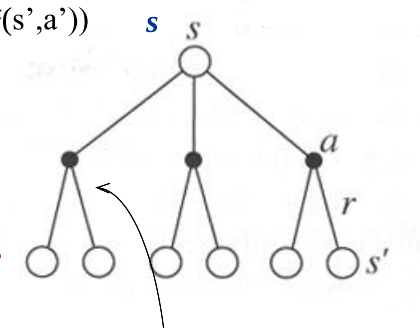
$$\boxed{V^{\pi}(s) = \sum_j \pi(a_j, s)\left\{\sum_{s_l'}\left\{P_{s \to s_l' | a_j}\left[R_{s \to s_l' | a_j} + \gamma V^{\pi}(s_l')\right]\right\}\right\}}$$



# Osservazioni

$V^{\pi}(s) = Q^{\pi}(s,a) = \text{funz}(Q^{\pi}(s',a'))$

***s***

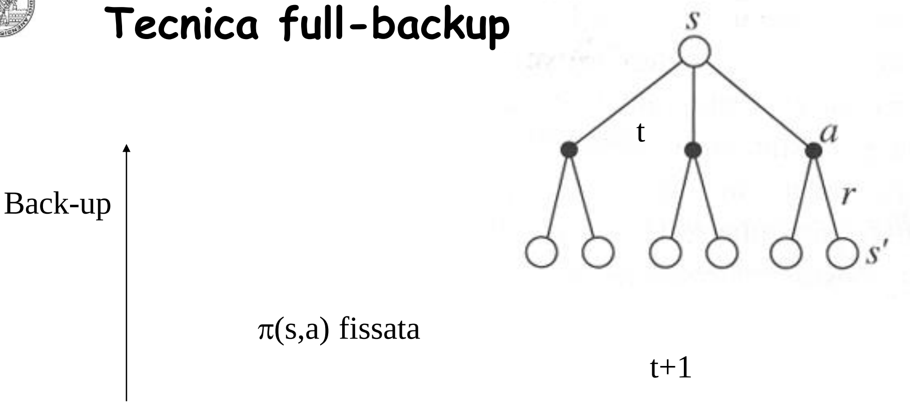



**Backwards in time**

***s'***

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

# Tecnica full-backup



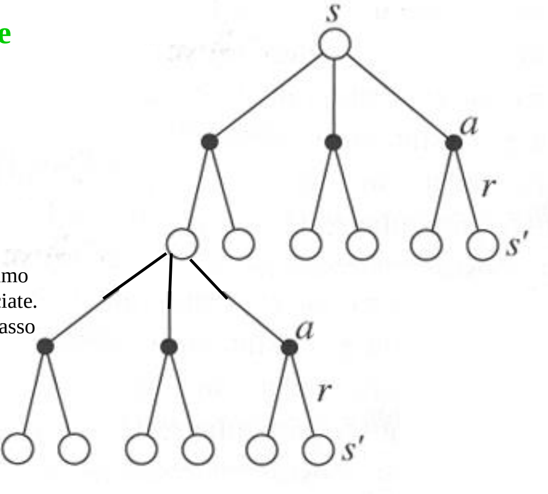

Conosciamo $V_k(s_t)$ $\forall s_t$, anche per $s'_{t+1}$ quindi:
Analizziamo la transizione da $s_t,a_t$ -> $(s'_{t+1})$
Calcoliamo un nuovo valore di V per s: $V_{k+1}(s_t)$ congruente con:
$V_k(s_t)$ ed $r_{t+1}$
*Full backup* se esaminiamo tutti gli s' (cf. DP).
Da s' mi guardo indietro ed aggiorno V(s).

$\boxed{\pi \text{ fissata}}$



# Analogia con la programmazione dinamica

**Backwards in time**




Per ogni stato calcolo il costo minimo
per arrivare in s e il percorso associate.
Il calcolo e' a 1 passo: costo di 1 passo
+ costo totale dallo stato di arrivo

## Sommario

Le equazioni di Bellman

Stima iterativa della funzione valore

Policy iteration

Esempi



## Calcolo iterativo della Value Function

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

Equazione di Bellman per "iterative policy evaluation":

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

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

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

## Iterative policy evaluation

Evoluzione del sistema da s(t=0) a {s'(t = T)} utilizzando la policy $\pi$(s,a), prefissata.

Quanto valgono gli stati?

Parto da $V(s(t=0))_{k=0}$ arbitraria, otterrò una value function per ogni stato-azione che sarà funzione di V(s(t=0)).

Devo migliorare, come?

Utilizziamo l'informazione sul **passato**, tenendo conto che gli stati sono in numero finito e vengono ri-visitati.

$\{V\}^0, \{V\}^1, \{V\}^2, \{V\}^3, \{V\}^4, \{V\}^5, \ldots\ldots. \{V\}^\infty$

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



## Fondamenti del metodo

- Supponiamo di essere all'istante t. In questo istante t, si può passare ad un certo insieme di stati: $\{s'_{t+1}\}$.

- Analizziamo un solo passo: cosa succede nella transizione da t a t+1.

- Migliorare la stima della nostra Value Function ad ogni iterazione.

# Algoritmo per "iterative policy evaluation", versione batch

Partiamo da una politica π(s,a) data.
Definiamo una soglia di convergenza τ
Inizializziamo V(s) = 0 ∀s, compreso gli stati finali.

Forwards pass

```
Repeat
{    Δ = 0;
     for s = 1 : NS                          // ∀s, ≠ TS
    {    Temp_V(s) = 0;
         for a = 1: NA                       // ∀a that can be chosen in s
         {   Pr_a = policy(s,a);
             for snext = 1 : NS
            {      PrSnext = NextState(s,a);
                   reward = ComputeReward(s,a,snext);
                   Temp_V(s) = Pr_a*PrSNext*(reward + γV(snext));
     }     }  }
     for s=1:NS;
     {      if ( |  Temp_V(s) – V(s) | > Δ )
            {    Δ =    |  Temp_V(s) – V(s) | ;
                 V(s) = Temp_V(s);
            }
    }
}  Until (Δ < τ);
```

Backwards pass

# Interpretazione dell'update (batch o trial)

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

Al termine dell'aggiornamento dei V(s) per tutti gli stati, V(s) = $V_{new}$(s). **Aggiornamento batch.**

Utilizzerò in parte già il nuovo valore di V(s) all'interno dell'equazione di aggiornamento. **Aggiornamento per trial.**

Entrambe le modalità di aggiornamento convergono.

## Algoritmo per "iterative policy evaluation", versione per trial

Partiamo da una politica $\pi(s,a)$ data.
Definiamo una soglia **relativa** di convergenza $\tau$
Inizializziamo $V(s) = 0$ $\forall s$, compreso gli stati finali.
Repeat

```
{       Δ = 0;
        for s = 1 : N                    // ∀s, ≠ TS
        {       Value = V(s);
```

$$V_{k+1}(s) = \sum_{a_j} \pi(s,a_j) \sum_{s'} P_{s->s'}^{a_j} \left[ R_{s->s'}^{a_j} + \gamma V(s') \right]$$

$$\Delta = \max(\Delta, (| Value - V_{k+1}(s) |) / | Value |)$$

```
        }
} Until (Δ < τ);
```



## Problematiche legate al calcolo di V(s): problema di policy evaluation

3 assunzioni:

1) Conoscenza della dinamica dell'ambiente: P(s->s'| a)
2) Conoscenza della policy (eventualmente stocastica), $\pi(s, a)$
3) Potenza di calcolo sufficiente
4) Proprietà Markoviane dell'ambiente (definizione di uno stato).

Le equazioni contengono dei termini statistici (valori attesi).

Soluzione di un sistema lineare in N incognite (numero di stati).

Come mai posso determinare la Value function per la policy $\pi(.)$, se questa si basa sul reward che riceverò negli istanti futuri?

C'e' poca interazione con l'ambiente e molta simulazione (cf. metodi Montecarlo).

## Riassunto

Posso determinare la Value function in modo ricorsivo. Per ogni stato, sarà funzione dell'output dell'ambiente in quell'istante (attraverso la funzione stato prossimo ed il reward istantaneo) e della policy scelta in quell'istante e dei reward a lungo termine attesi negli stati in cui l'ambiente mi porta.

Per scegliere la policy devo esaminare il reward a lungo termine che mi si prospetta nello stato in cui mi trovo e scegliere l'azione che lo massimizza.



## Problematiche legate al calcolo di V*(s)

Soluzione vicina alla ricerca esaustiva. Devo valutare per ogni stato tutte le possibili azioni (devo trovare il massimo).

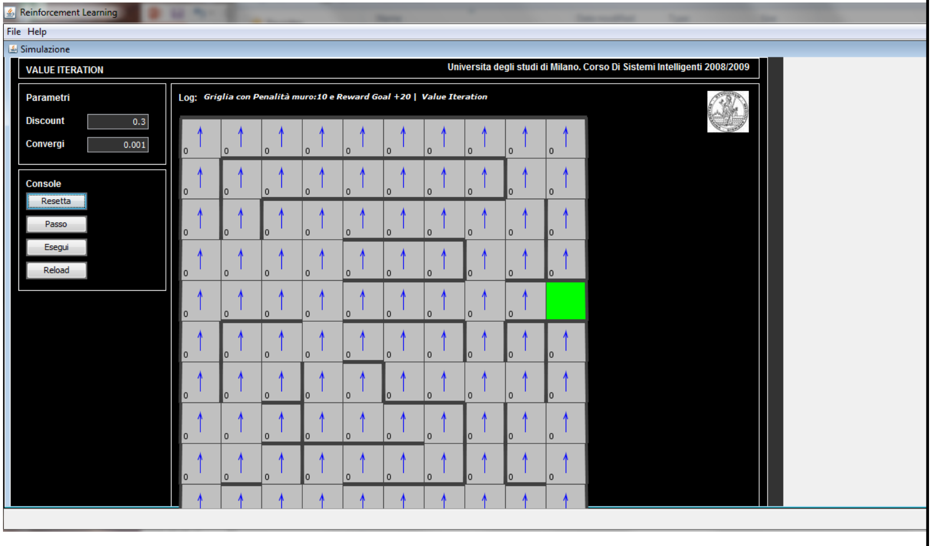
Per tutte le possibili azioni devo calcolare la probabilità di transizione allo stato successivo e di ottenere una certa reward.

3 assunzioni:

1) Conoscenza della dinamica dell'ambiente: $P(s\text{->}s' | a_j)$
2) Potenza di calcolo sufficiente
3) Proprietà Markoviane dell'ambiente (definizione di uno stato).

Soluzioni approssimate.

# Iterative policy evaluation




DeRosaRocco_ValueIteration



# Sommario

Le equazioni di Bellman

Stima iterativa della funzione valore

Policy iteration

Esempi

# 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 $s_m$? $a_{new} \neq \pi(s_m)$.
Cosa viene influenzato?

Scelgo $a_{new}$ in $s_m$, visiterò una certa sequenza di stati, per questi stati seguirò la policy precedente per $s \neq s_m$. 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 $s_m$, $\{s_{t+k}\}$, ed il reward a lungo termine:

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

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

V(s) = value function sullo stato

?

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

Se il reward fosse migliore con $a_{new}$, sceglierò sempre $a_{new}$ in $s_m$.

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

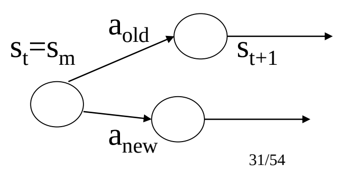
# Enunciato del teorema del miglioramento della policy

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

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

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

**Tesi:** $\pi$' è meglio di $\pi$. Cioè: $V^{\pi'}(s) >= V^{\pi}(s)$ **∀s.**
$Q^{\pi'}(s,a_{new}) >= Q^{\pi}(s,a_{old})$






# Dimostrazione del teorema del miglioramento della policy

**Analizziamo la seguente condizione:**

$\pi$'= $\pi$ ∀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 $a_{new} = \pi'(s)$.

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

**Tesi:** $\pi$' è meglio di $\pi$. Cioè: $V^{\pi'}(s) >= V^{\pi}(s)$ ∀s (ed in particolare per gli altri stati 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 $Q^{\pi *}(.)$

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

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



## Osservazioni

$s = s_m$ $$Q^\pi(s_m, \pi'(s)) \geq Q^\pi(s_m, \pi(s))$$

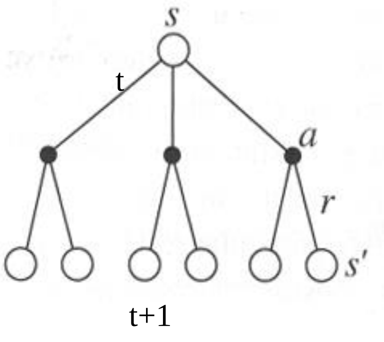
$s \neq s_m$ $$Q^\pi(s,a) = E_{\pi'}\{r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s\}$$

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

Se $s_{t+k}$ = sm miglioro la Q(s,a).

Se nessun $s_{t+k} = s_m$. Non varia la Q(s,a).

# Visione grafica del miglioramento




Ogni volta che sono in uno stato, s , scelgo un'azione che migliora il reward a lungo termine ottenuto da quell'istante/stato in poi.

Per gli altri stati, il reward a lungo termine non viene modificato ogni volta che l'albero uscente da s' passa per s.



# Ottimizzazione policy

Per ogni stato scelgo le azioni secondo la policy: $\pi(s,a)$.

Posso ordinare la Value function Q(s,a) in ordine decrescente, in funzione delle azioni scelte in s (policy).

Si definisce una policy, $\pi_1$, migliore di un'altra, $\pi_2$, se e solo se:
$Q^{\pi 1}(s,a(s)) > =Q^{\pi 2}(s,a(s)) \; \forall s$.

In particolare si definisce una politica ottima, $\pi^*$, se e solo se:
$Q^*(s,a(s)) > =V^{\pi}(s,a(s)) \; \forall s$

$Q^*(s,a(s)) > =Q^{\pi}(s,a(s)) \; \forall [s,a]$

## Calcolo ricorsivo della Value function ottima::confronti

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

Q*(s,a) di uno stato-azione, quando viene scelta la policy ottima, deve essere uguale al valore atteso del reward per l'azione migliore per lo stato s.

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

Politica greedy: scelgo l'azione ottimale.
Ha senso per il robot raccogli-lattine?

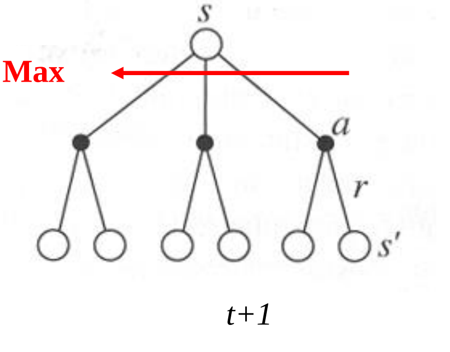


## V*(s) – Osservazioni

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

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
(cf. Software Sommaruga)



## Algoritmo

**Inizialization**

V(s) = 0;
$\pi$(s,a) = random (e.g. equiprobabile);

Repeat
    point 2.
    point 3.
until policy_stable

## Algoritmo – point2

**Policy evaluation – versione per trial**

Repeat
th = 0; // small value;
for s = 1:N
V_temp = $\sum_{a_j} \pi(s, a_j) \sum_{s'} \Pr_{s->s'|a_j} \left[ R_{s->s'|a_j} + \gamma V(s') \right]$

$\Delta$V = |V(s) – V_temp)|
V(s) = V_temp;
th = max(th, $\Delta$V)
end;
until th < th_max;



## Algoritmo – point3

**Policy improvement**

policy_stable = true;
for s = 1:N // in alternativa, scelgo uno stato
a_old = $\pi$(s);
a_new = $\arg\max_a \left( \sum_{s'} \Pr_{s->s'|a} \left[ R_{s->s'|a} + \gamma V(s') \right] \right)$ ;

if (a_new $\neq$ a_old)
policy_stable = false;
end;

## Algoritmo - II

**Policy evaluation – versione per epoch**

Repeat
  Th = 0; // small value;
  for s = 1:N
    V_temp(s,a) = $\sum_{a_j} \pi(s, a_j) \sum_{s'} \Pr_{s->s'|a_j} \left[ R_{s->s'|a_j} + \gamma V(s') \right]$
      $\Delta$V = |V(s) – V_temp(s))|
      th = max(th, $\Delta$V)
    end:
  end;
  for s = 1:N
    V(s) = V_temp(s);
   end; end;
until th < th_max;



## Max or soft max

**Policy improvement**

policy_stable = true;
for s = 1:N  // in alternativa, scelgo uno stato
  a_old = $\pi$(s);

  a_new = $\arg\max_a \left\{ \sum_a \pi(s,a) \sum_{s'} \Pr_{s->s'|a} \left[ R_{s->s'|a} + \gamma V(s') \right] \right\}$

  if (a_new $\neq$ a_old)
    policy_stable = false;
end;

Max con policy $\varepsilon$-greedy, soft-max, ...

## Iterative policy evaluation sulla value function V(s)

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

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



## Value iteration

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

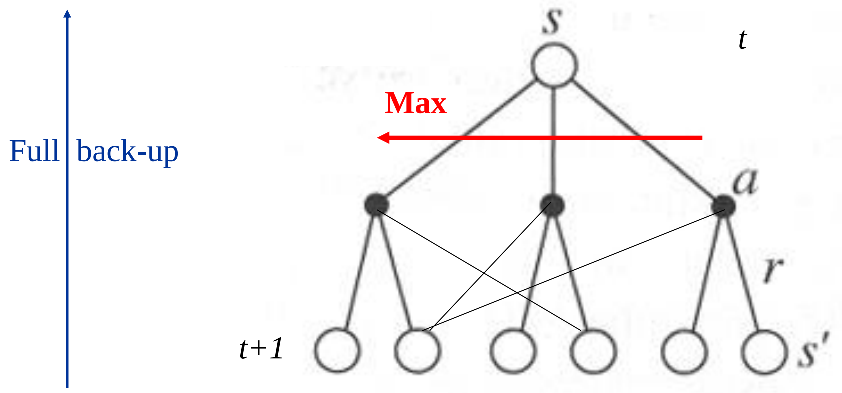
Invece di considerare una policy stocastica, consideriamo l'azione migliore:

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

$\forall s$

## Visualizzazione grafica

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

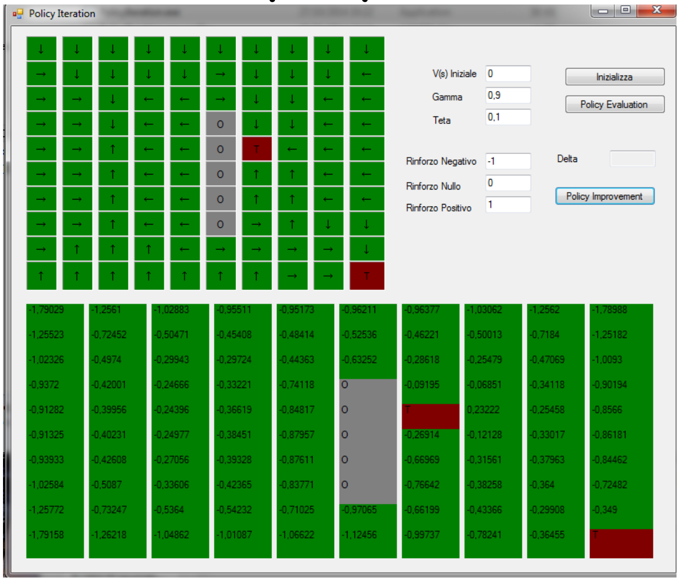





## Sommario

- Le equazioni di Bellman
- Stima iterativa della funzione valore
- Policy iteration
- Esempi

# Iterative policy evaluation

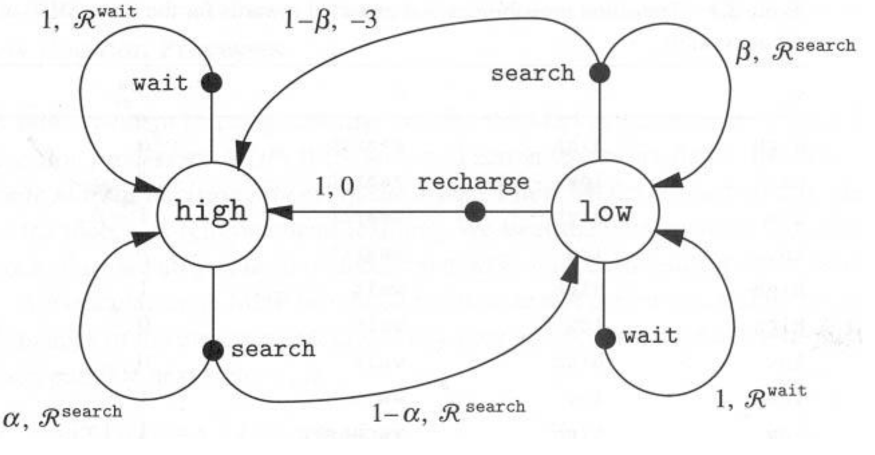





Forlivesi_PolicyIteration_Labirinto



# Robot cerca-lattine



## Esempio: robot – Policy deterministica

Q(h,search) = Pr(h->l, search) x [R(h->h, <u>search</u>) + $\gamma$ x Q(h,search)]
Pr(h->h, search) x [R(h->l, <u>search</u>) + $\gamma$ Q(l,wait)]
Q(h,search) = 0.4x[3+0.8xQ(h,search)]+0.6x[3+0.8Q(l,wait)]

Q(l,wait) = Pr(l->l, wait) x [R(l->l, wait)+0.8 x Q(l,wait)]
Q(l,wait) = 1 x [1+0.8 Q(l,wait)]

**Policy iniziale deterministica:**

**STATO: Q(h,search) ➔**
Q(h,s) ≅ 4,4 + 0.7 Q(l,w) ≅ 7.95
**STATO: Q(l, wait) ➔**
Q(l,wait) = 5

Posso migliorare la policy?



## Esempio: robot – miglioramento policy

**Miglioro la policy, modificando l'azione associata a s = low:**

**STATO: high**
a = search ➔ Q(h,search) ≅ 4,4 + 0.7 Q(l,recharge) = ??? ≠7.95
**STATO: low**
a = recharge ➔ Q(l,recharge) = 0 + 0.8 Q(h,search) = ???

Ho stimato correttamente Q(h,search)? No
Applico un passo di iterative policy evaluation, in modalità trial.
**STATO: Vl**
a = recharge → Q(l,r) = 0.8 Q(h,s) = 0.8 x 7.95 = 6.36
**STATO: high**
a = search ➔ Q(h,s) ≅ 4.4 + 0.7 Q(l,r) ≅ 4.4 + 0.7 x 6.36 = 8.85
Ho stimato correttamente Q(s,a)? No. Devo iterare la policy evaluation.

# Esempio: robot - IV

**<u>Asintoticamente</u> calcolo il valore vero delle coppie stato-azione:**

**STATO: high**
a = search ➔ $Q(h,s) \cong 4.4+0.7\ Q_1(l,r) = 4.4 + 0.7x6.36 = 8.85$
**STATO: low**
a = recharge → $Q(l,r) = 0.8\ Q(h,s) = 0.8x8.85 = 7.08$

Potrei ottenere gli stessi valori ottenuti asintoticamente, risolvendo il sistema lineare:
$Q(h,s) = 4.4 + 0.7\ Q(l,r) = 10$
$Q(l,r) = 0.8\ Q(h,s) = 8$

A questi valori si arriva solo asintoticamente



# Sommario

- Le equazioni di Bellman
- Stima iterativa della funzione valore
- Policy iteration
- Esempi