

Effettuare decisioni complesse

Alessandro Saetti

(saetti@ing.unibs.it)

Università degli Studi di Brescia



Intelligenza Artificiale

A.A. 2011/2012

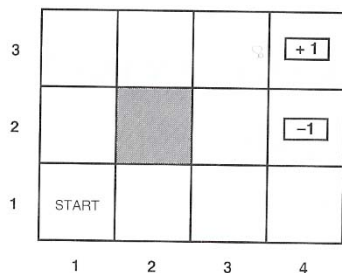
Planning vs Sequential Decisions

Differenze tra Planning e Sequential Decision Problem:

- **Planning:**
 - l'obiettivo è realizzare i goals del problema
 - l'ambiente è deterministico e completamente osservabile
 - ⇒ l'output è una sequenza (parzialmente ordinata) di azioni
- **Sequential Decisions:**
 - l'obiettivo è massimizzare una funzione di utilità
 - l'ambiente non è deterministico e completamente osservabile
 - ⇒ l'output è il comportamento dell'agente

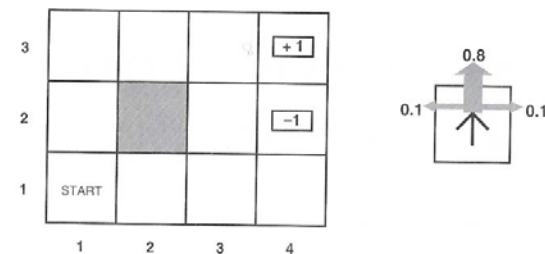
Introduzione

- Se l'ambiente fosse deterministico...



- Le possibili azioni sono: {Up, Down, Left, Right, Wait}
- Se in (1,1) esegue l'azione Down, l'agente rimbalza contro il muro e resta in (1,1)

Sequential Decision Problems



- Dalla posizione (1,1) l'azione Up,
 - con probabilità 0.8 sposta l'agente in (2,1)
 - con probabilità 0.1 sposta l'agente a destra in (1,2)
 - con prob. 0.1 rimbalza contro il muro e resta in (1,1)
- La sequenza di decisioni [Up, Up, Right, Right, Right] raggiunge (3,4) solo con probabilità 0.32768

Sequential Decision Problems

- Il problema consiste nell'eseguire (effettuare) una sequenza di azioni (decisioni)
- Ciascuna azione può comportare un cambiamento **non deterministico** sullo stato del mondo
 - Con osservabilità, conosco lo stato dopo aver eseguito l'azione
 - Senza osservabilità, non conosco lo stato neanche dopo aver eseguito l'azione (lo stato è una distribuzione di probabilità sui possibili stati del mondo)
- Il processo di cambiamento può essere visto come una sequenza di "snapshot"

Processo Stocastico a tempo discreto

- La situazione al tempo t è rappresentato da una variabile casuale S^t
- Il processo decisionale è interpretato come un **processo stocastico a tempo discreto**
- Un processo stocastico a tempo discreto è una successione di variabili casuali $X_t, t \in \mathcal{N}$
 - *Spazio degli Stati*: insieme di valori che le variabili casuali possono assumere
 - *Storia (traiettoria) di un processo stocastico*: realizzazione della successione di v.c.

Processo Stocastico a tempo discreto

- Il risultato del lancio di una moneta è testa (T) o croce (C)

Lo spazio degli stati è $\{T, C\}$

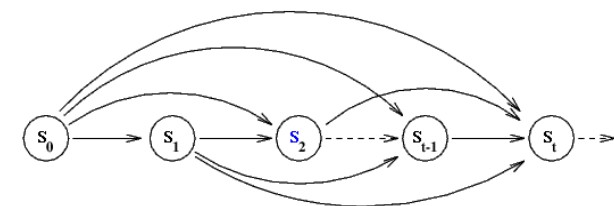
Una traiettoria possibile è T, C, C, T, C

TCCTCTCTTCT
CTCTCTCTCTC
TCTCTCTTCTC
12345....

- Problemi in questo modello stocastico:
 - illimitato numero di variabili casuali
 - dipendenti da un illimitato numero di variabili casuali
 - dalle quali dipende un illimitato numero di variabili casuali

Ipotesi di "Forward Causality"

- **Ipotesi di "Forward Causality"**:
 S_t non dipende da v.c. S_k con $k > t$.
- Il sistema è descritto da $\forall t \geq 0, P(S_t | S_0, S_1, \dots, S_{t-1})$



Nodi: variabili casuali Archi: dipendenze statistiche

- I genitori sono solo i nodi precedenti
- Potrebbero ancora essere un numero illimitato

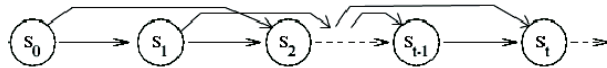
Ipotesi di "Markov"

- **Ipotesi di "Markov":**

S_t dipende da un numero finito di v.c S_i con $i < t$

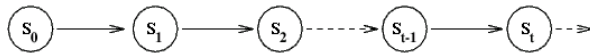
- **Processi di Markov del secondo ordine:**

$\forall t \geq 0, P(S_{t+1}|S_t, S_{t-1}, \dots, S_0) = P(S_{t+1}|S_t, S_{t-1})$



- **Processi di Markov del primo ordine:**

$\forall t \geq 0, P(S_{t+1}|S_t, S_{t-1}, \dots, S_0) = P(S_{t+1}|S_t)$



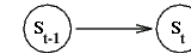
- Risolve il problema sul numero illimitato di genitori

Ipotesi di "Stazionarietà"

- **Ipotesi di stazionarietà:**

$\forall t > 0, P(S_t|S_{t-1}) = P(S_{t+1}|S_t)$

- Il sistema è descritto da $P(S_t|S_{t-1})$



- $P(S_t|S_{t-1})$ è chiamato **modello di transizione** per processi del primo ordine

- Risolve il problema sul numero illimitato di variabili casuali

Processo Markoviano

- Supponiamo che il mondo evolve in conseguenza di eventi esterni e che non sia completamente osservabile

- **Notazione:**

X_t : le variabili non osservabili nello stato S_t

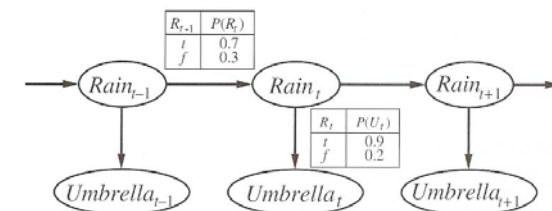
E_t le variabili osservabili nello stato S_t

- Un processo di Markov è descritto da:

- Modello di transizione: $\mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$
- Modello sensoriale: $\mathcal{O} \times \mathcal{S} \rightarrow \mathbb{R}$
- Distribuzione di probabilità sullo stato al tempo 0: $\mathcal{S} \rightarrow \mathbb{R}$

Esempio di Processo Markoviano

- Una guardia di sicurezza controlla una installazione sotterranea:
- Giornalmente il suo unico contatto con l'esterno è l'entrata e l'uscita del direttore
- Quando piove il direttore solitamente ha con sé l'ombrello
- La guardia si chiede se all'esterno piove



Esempio di Processo Markoviano

- Il modello di transizione è descritto da una matrice 2x2
- L'elemento x_{ij} della matrice indica $P(X_{t+1} = x_i | X_t = x_j)$
- Il modello sensoriale è descritto da una matrice 2x2
- L'elemento x_{ij} della matrice indica $P(E_t = e_i | X_t = x_j)$
- X_0 è un vettore di 2 colonne
- L'elemento x_i del vettore indica $P(X_0 = x_i)$

Inferenze su un Processo Markoviano

- Dato una completa descrizione di un processo markoviano è possibile compiere le seguenti inferenze:
 - Monitoraggio: $P(X_t | e_{1..t})$
 - Predizione: $P(X_{t+k} | e_{1..t})$ per $k > 0$
 - Retrospezione: $P(X_k | e_{1..t})$ per $k > 0$
 - Spiegazione più probabile: $\text{ARGMAX}_{x_{1..t}} P(x_{1..t} | e_{1..t})$

Esempio di Monitoraggio

- Supponiamo di osservare l'ombrello per due giorni di fila e che inizialmente si ha completa incertezza sul meteo esterno

$$P(R_{t+1} | R_t) = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}$$

$$P(u_t | R_t) = \begin{bmatrix} 0.9 & 0 \\ 0 & 0.1 \end{bmatrix}$$

$$P(\bar{u}_t | R_t) = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.8 \end{bmatrix}$$

$$P(R_0) = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

Esempio di Monitoraggio

- $P(R_1) = P(R_1 | R_0) P(R_0)$
$$P(R_1) = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$
- $P(u_1) = P(u_1 | r_1) P(r_1) + P(u_1 | \bar{r}_1) P(\bar{r}_1)$
$$P(u_1) = 0.9 \cdot 0.5 + 0.2 \cdot 0.5 = 0.55$$
- $P(R_1 | u_1) = \frac{P(u_1 | R_1) P(R_1)}{P(u_1)}$

$$P(R_1 | u_1) = \frac{\begin{bmatrix} 0.9 & 0 \\ 0 & 0.2 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}}{0.55} = \begin{bmatrix} 0.818 \\ 0.182 \end{bmatrix}$$

Esempio di Monitoraggio

- $P(R_2|u_1) = P(R_2|R_1)P(R_1|u_1)$

$$P(R_2|u_1) = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix} \begin{bmatrix} 0.818 \\ 0.182 \end{bmatrix} = \begin{bmatrix} 0.627 \\ 0.373 \end{bmatrix}$$

- $P(u_2) = P(u_2|r_2)P(r_2) + P(u_2|\bar{r}_2)P(\bar{r}_2)$

$$P(u_2) = 0.9 \cdot 0.627 + 0.2 \cdot 0.373 = 0.639$$

- $P(R_2|u_1, u_2) = \frac{P(u_2|R_2)P(R_2|u_1)}{P(u_2)}$

$$P(R_2|u_1, u_2) = \frac{\begin{bmatrix} 0.9 & 0 \\ 0 & 0.2 \end{bmatrix} \begin{bmatrix} 0.627 \\ 0.373 \end{bmatrix}}{0.639} = \begin{bmatrix} 0.883 \\ 0.117 \end{bmatrix}$$

Processo Decisionale di Markov (MDP)

- È un sequential decision problem in cui
 - l'ambiente è completamente osservabile
 - il modello di transizione è markoviano del primo ordine e stazionario
 - i rewards sono additivi
- Un MDP è descritto da:
 - Un modello di transizione: $T : S \times S \times \mathcal{A} \rightarrow \mathcal{R}$
 - Una funzione di reward: $R : S \rightarrow \mathcal{R}$
 - Uno stato iniziale: s_0

Politica

- Una sequenza di decisioni non risolve il problema
- La soluzione dovrebbe specificare cosa l'agente deve fare in qualsiasi stato
- **Politica:** $\pi : S \rightarrow \mathcal{A}$
- Se la politica è completa l'agente sa sempre come comportarsi

```

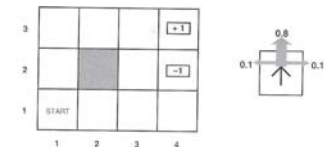
Execute-Policy( $\pi$ )
  Loop
    sense the current state  $s$ 
     $a \leftarrow \pi(s)$ 
    execute action  $a$ 
  end
    
```

- Rappresenta il comportamento dell'agente in modo esplicito (un "riflesso" dell'agente)

Esempio di Politica

- MDP bilancia costi e benefici

- Esempio:



$$\pi_1 = \{((1,1), \text{Up}), ((2,1), \text{Up}), ((3,1), \text{Right}), ((3,2), \text{Right}), ((3,3), \text{Right}), ((1,2), \text{Right}), ((1,3), \text{Up}), ((2,3), \text{Up}), ((1,4), \text{Left})\}$$

$$\pi_2 = \{((1,1), \text{Up}), ((2,1), \text{Up}), ((3,1), \text{Right}), ((3,2), \text{Right}), ((3,3), \text{Right}), ((1,2), \text{Left}), ((1,3), \text{Left}), ((2,3), \text{Left}), ((1,4), \text{Down})\}$$

- π_1 è più veloce di π_2 ma più rischiosa
- Dato che il costo $0.04 \ll$ del costo di finire in -1 , π_2 dovrebbe essere preferibile a π_1

Storie

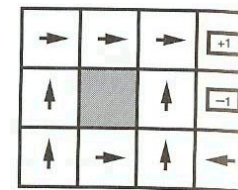
- *Storia*: sequenza (infinita) di stati. $h = \langle s^0, s^1, s^2, \dots, s^i, \dots \rangle$
- L'esecuzione di una politica π corrisponde ad una storia
- Se il mondo contiene uno stato finale, l'esecuzione di una politica può generare una sequenza finita di stati

- Data una politica π , alcune storie sono più probabili di altre

$$P(h|\pi) = \prod_{i \geq 0} T(s^i, \pi(s^i), s^{i+1})$$

- Data una politica, differenti stati iniziali generano altre storie

Esempi di Storie



$h_0 = \langle (1,1), (1,2), (1,3), (2,3), (2,3), (3,3), (3,4) \rangle$

$h_1 = \langle (1,1), (2,1), (3,1), (3,2), (3,3), (3,4) \rangle$

$h_2 = \langle (1,1), (1,2), (1,1), (2,1), (3,1), (3,2), (3,3), (3,4) \rangle$

$h_3 = \langle (1,1), (1,1), (1,1), \dots \rangle$

$P(h_0|\pi_1) = 0.1 \cdot 0.8 \cdot 0.8 \cdot 0.1 \cdot 0.8 \cdot 0.8 = 0.004$

$P(h_1|\pi_1) = 0.8 \cdot 0.8 \cdot 0.8 \cdot 0.8 \cdot 0.8 = 0.3276$

$P(h_2|\pi_1) = 0.1 \cdot 0 \cdot 0.8 \cdot 0.8 \cdot 0.8 \cdot 0.8 \cdot 0.8 = 0$

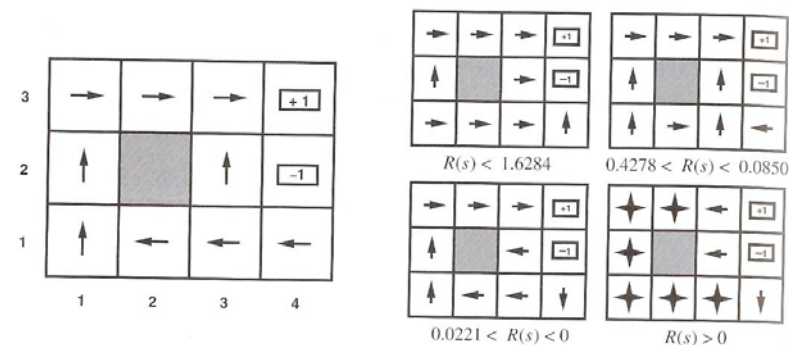
$P(h_3|\pi_1) = 0.1 \cdot 0.1 \cdot \dots = 0$

Politica Ottima

- Ogni storia ha una propria utilità
- Fondamentalmente l'utilità di una storia è la somma dei rewards degli stati "raccontati"
- La qualità di una politica è misurata in termini dell'utilità *attesa* tra le possibili storie
- **Politica ottima**: politica con la più alta utilità attesa

Esempio di Politica Ottima

Il bilanciamento tra rischi e reward dipende da $R(s)$:



Orizzonte

- **Politiche ad orizzonte infinito:** non si ha un limite sul numero di “mosse”

Le politiche sono stazionarie: $\pi : \mathcal{S} \rightarrow \mathcal{A}$

- **Politiche ad orizzonte finito:**

$$\exists N \in \mathbb{N}, \forall h \in \mathcal{H} \cdot U(\langle s^0, s^1, \dots, s^{N+k} \rangle) = U(\langle s^0, s^1, \dots, s^N \rangle)$$

Le politiche sono non stazionarie: $\pi : \mathcal{S} \times \mathbb{N} \rightarrow \mathcal{A}$

- Calcolare politiche ad orizzonte infinito è più semplice

Utilità di Storie

- **Ipotesi di Stazionarietà:** $\langle s_0, s_1, s_2, \dots \rangle, \langle s'_0, s'_1, s'_2, \dots \rangle$. Se $s_0 = s'_0$ allora le due storie sono ordinate in base a $\langle s_1, s_2, \dots \rangle, \langle s'_1, s'_2, \dots \rangle$.

- **Premi additivi:**

$$U(\langle s_0, s_1, s_2, \dots \rangle) = R(s_0) + R(s_1) + R(s_2) + \dots$$

- **Proper policy:** una politica che assicura di raggiungere uno stato terminale

Con “proper policy” è possibile usare questa strategia

- Se il mondo non contiene uno stato terminale, l'utilità di tutte le storie sarà $\pm\infty$

Utilità di Storie

- **Premi scontati:** Sia $0 < \gamma \leq 1$,

$$U(\langle s_0, s_1, s_2, \dots \rangle) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots$$

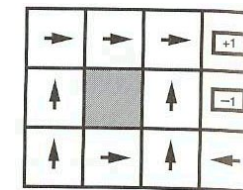
$$- U(\langle s_0, s_1, s_2, \dots \rangle) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{R_{max}}{1-\gamma}$$

- In caso l'ambiente ammetta politiche **non-proper** è necessario usare questa strategia

- Se $\gamma = 1$ coincide con la strategia a premi additivi

- Se γ è vicino a 0 i reward futuri sono insignificanti

Esempi di Utilità di Storie



$$h_0 = \langle (1, 1), (1, 2), (1, 3), (2, 3), (2, 3), (3, 3), (4, 3) \rangle$$

$$h_4 = \langle (1, 1), (2, 1), (3, 1), (3, 2), (4, 2) \rangle$$

Supponiamo $\gamma = 1$, $R(s) = -0.04$,

$$U(h_0) = (-0.04) + (-0.04) + (-0.04) + (-0.04) + (-0.04) + (-0.04) + 1 = 0.76$$

$$U(h_4) = (-0.04) + (-0.04) + (-0.04) + (-0.04) - 1 = -1.16$$

Utilità di una Politica

- È la media pesata tra tutte le utilità delle possibili storie

- $U(\pi) = \sum_{h \in \mathcal{H}} P(h|\pi)U(h)$

- Ad esempio, con $\gamma = 1$, $R(s) = -0.04$,

$$h_0 = \langle (1, 1), (1, 2), (1, 3), (2, 3), (2, 3), (3, 3), (4, 3) \rangle, P(h_0|\pi_1) = 0.004, U(h_0) = 0.76$$

$$h_1 = \langle (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3) \rangle, P(h_1|\pi_1) = 0.3276, U(h_1) = 0.8$$

$$h_2 = \langle (1, 1), (1, 2), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3) \rangle, P(h_2|\pi_1) = 0, U(h_2) = 0.72$$

$$h_3 = \langle (1, 1), (1, 1), (1, 1), \dots \rangle, P(h_3|\pi_1) = 0, U(h_3) = -\infty$$

$$U(\pi_1) = 0.004 \cdot 0.76 + 0.3276 \cdot 0.8 + 0 \cdot 0.72 + \dots$$

- La politica ottima è quella che massimizza l'utilità attesa

Utilità di uno Stato

- L'utilità di uno stato s è definito in termini di utilità delle possibili storie che iniziano da s

$$U_\pi(s) = \sum_{h \in \mathcal{H}} p(h|\pi)U(h)|_{s_0=s}$$

3	0.812	0.868	0.918	+1
2	0.762		0.660	-1
1	0.705	0.655	0.611	0.388
	1	2	3	4

- **Principi della massima utilità attesa:** La politica ottima è quella che massimizza, per ogni stato, l'utilità attesa dai possibili successori

$$\pi^*(s) = \text{ARGMAX}_a [\sum_{s'} T(s, a, s')U(s')]$$

Equazione di Bellman

- $U(s)$ dipende da

- $R(s)$ tiene conto del premio a “breve termine” in s
- $U(s')$ tiene conto del premio a lungo termine da s' in poi

- **Equazione di Bellman** (Bellman'57):

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s')U(s')$$

L'utilità su uno stato è la somma tra il reward dello stato e l'utilità attesa dello stato successore generato dalla azione scelta dalla politica

Esempio su Equazione di Bellman

L'utilità dello stato (1,1) è la soluzione della equazione di Bellman:

$$U((1, 1)) = -0.04 + \gamma \max \left\{ \begin{array}{ll} 0.8U((2, 1)) + 0.1U((1, 2)) + 0.1U((1, 1)), & (Up) \\ 0.9U((1, 1)) + 0.1U((2, 1)), & (Left) \\ 0.9U((1, 1)) + 0.1U((1, 2)), & (Down) \\ 0.8U((1, 2)) + 0.1U((2, 1)) + 0.1U((1, 1)) \} & (Right) \end{array} \right.$$

$$U((1, 1)) = -0.04 + 1 \cdot \max \left\{ \begin{array}{ll} 0.8 \cdot 0.762 + 0.1 \cdot 0.655 + 0.1 \cdot 0.705, & (Up) \\ 0.9 \cdot 0.705 + 0.1 \cdot 0.762, & (Left) \\ 0.9 \cdot 0.705 + 0.1 \cdot 0.655, & (Down) \\ 0.8 \cdot 0.655 + 0.1 \cdot 0.762 + 0.1 \cdot 0.705 \} & (Right) \end{array} \right.$$

$$U((1, 1)) = -0.04 + 1 \cdot \max\{0.745, 0.710, 0.699, 0.670\} = -0.04 + 0.745 = 0.705$$

In base alle utilità attese negli stati (1,1), (1,2) e (2,1), Up è la migliore azione

L'algoritmo Value Iteration

- Calcola le utilità attese di ciascuno stato
- È basato sulla equazione di Bellman
- Con n stati, ho n equazione di Bellman con n incognite
- Il sistema di equazione è non-lineare
- Approccio *iterativo*: $U_{i+1} \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U_i(s')$
 - Si inizia da un arbitrario assegnamento alle utilità attese
 - Si aggiornano le utilità attese in base alle utilità dei vicini
 - Si itera finché non si raggiunge l'equilibrio (a meno di un ϵ)

L'algoritmo Value Iteration

```

function VALUE-ITERATION( $mdp, \epsilon$ ) returns a utility function
  inputs:  $mdp$ , an MDP with states  $S$ , transition model  $T$ , reward function  $R$ , discount  $\gamma$ 
            $\epsilon$ , the maximum error allowed in the utility of any state
  local variables:  $U, U'$ , vectors of utilities for states in  $S$ , initially zero
                     $\delta$ , the maximum change in the utility of any state in an iteration

  repeat
     $U \leftarrow U'; \delta \leftarrow 0$ 
    for each state  $s$  in  $S$  do
       $U'[s] \leftarrow R[s] + \gamma \max_a \sum_{s'} T(s, a, s') U[s']$ 
      if  $|U'[s] - U[s]| > \delta$  then  $\delta \leftarrow |U'[s] - U[s]|$ 
    until  $\delta < \epsilon(1 - \gamma)/\gamma$ 
  return  $U$ 
  
```

Figure 17.4 The value iteration algorithm for calculating utilities of states. The termination condition is from Equation (17.8).

Convergenza di Value Iteration

- **Contrazione:** funzione che se applicata su due valori produce due nuovi valori tra loro più vicini degli originali
 - una contrazione ha solo un fixed point
 - quando la funzione è applicata ad un argomento, il nuovo valore deve essere più vicino al fixed point rispetto al precedente

- “Aggiornamento di Bellman”: $U_{i+1} \leftarrow B(U_i)$
- La distanza tra due vettori è misurata tramite *max norm*:

$$\|U\| = \max_s |U(s)|$$
- L'aggiornamento di Bellman è una contrazione di un fattore γ :

$$\|B(U_i) - B(U'_i)\| \leq \gamma \|U_i - U'_i\|$$

Convergenza di Value Iteration

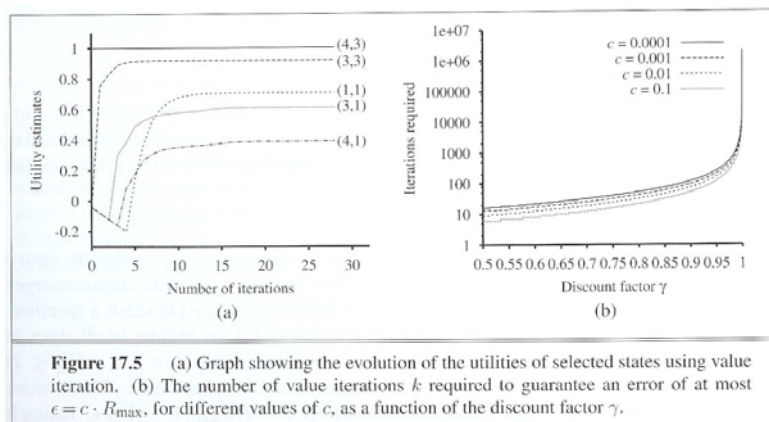
- Dato che $B(U) = U$, $\|B(U_i) - B(U)\| = \|U_{i+1} - U\| \leq \gamma \|U_i - U\|$

- L'errore $\|U_i - U\|$ è ridotto ad ogni iterazione di un fattore γ

⇒ Value iteration converge in modo esponenziale

- $\|U_0 - U\| \leq 2R_{max}/(1 - \gamma)$
- $\gamma^N \cdot 2R_{max}/(1 - \gamma) \leq \epsilon \Rightarrow N = \lceil \log(2R_{max}/(\epsilon(1 - \gamma)))/\log(1/\gamma) \rceil$
- if $\|U_{i+1} - U_i\| < \epsilon \cdot \frac{1-\gamma}{\gamma}$ then $\|U_{i+1} - U\| < \epsilon$

Convergenza di Value Iteration



Algoritmo Value Iteration

- Aspetti positivi:
 - L'algoritmo converge in modo esponenziale
 - Il numero N di iterazioni non dipende molto da ϵ
 - Il numero N di iterazioni non dipende molto neppure da $R_{\max}$
 - Aspetto negativo: N cresce rapidamente se γ si avvicina ad 1
- Se γ è piccolo l'agente è miope (non riesce ad intuire gli effetti a lungo termine delle proprie azioni)

Algoritmo Policy Iteration

- Si inizia da una politica arbitraria π_0
- Calcola le utilità attese U_i di ciascuno stato in base a π_i
- Calcola una nuova politica π_{i+1} usando una strategia one-step look-ahead basata sui valori di U_i

$$\pi^*(s) = \text{ARGMAX}_a \left[\sum_{s'} T(s, a, s') U(s') \right]$$

- L'algoritmo termina quando l'iterazione i non cambia la politica

Algoritmo Policy Iteration

```

function POLICY-ITERATION( $mdp$ ) returns a policy
  inputs:  $mdp$ , an MDP with states  $S$ , transition model  $T$ 
  local variables:  $U, U'$ , vectors of utilities for states in  $S$ , initially zero
                  $\pi$ , a policy vector indexed by state, initially random

  repeat
     $U \leftarrow \text{POLICY-EVALUATION}(\pi, U, mdp)$ 
    unchanged?  $\leftarrow$  true
    for each state  $s$  in  $S$  do
      if  $\max_a \sum_{s'} T(s, a, s') U[s'] > \sum_{s'} T(s, \pi[s], s') U[s']$  then
         $\pi[s] \leftarrow \text{argmax}_a \sum_{s'} T(s, a, s') U[s']$ 
    unchanged?  $\leftarrow$  false
  until unchanged?
  return  $P$ 
    
```

Figure 17.7 The policy iteration algorithm for calculating an optimal policy.

Algoritmo Policy Iteration

- L'algoritmo termina
 - il numero di possibili politiche è finito
 - ad ogni iterazione la politica è migliorata
- Come ricavare le utilità attese data una politica?
 - Alla iterazione i , $\pi_i(s)$ specifica già l'azione da svolgere in s

$$U_i(s) = R(s) + \gamma \sum_{s'} T(s, \pi_i(s), s') U_i(s')$$

- Un sistema lineare di n equazioni in n incognite si risolve in $O(n^3)$

Algoritmo Misto

- Quando n è piccolo policy iteration converge più velocemente
- Quando n è grande, $O(n^3)$ può essere proibitivo
- Algoritmo misto: il calcolo delle utilità è un'approssimazione
Policy-Evaluation è implementato eseguendo alcune iterazione semplificate (la politica è fissata) di Value-Iteration

$$U_{i+1}(s) = R(s) + \gamma \sum_{s'} T(s, \pi_i(s), s') U_i(s')$$

- L'algoritmo misto è spesso più efficiente sia di Value-Iteration che Policy-Iteration

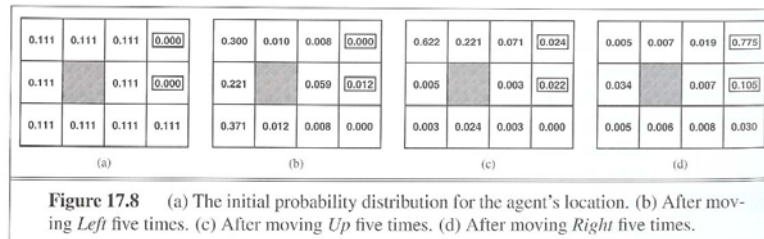
Ambiente parzialmente osservabili

- L'agente non sa in quale stato del mondo si trova
- L'agente non può eseguire l'azione $\pi(s)$ raccomandata per un certo stato s
- L'agente può migliorare la sua percezione del mondo esterno compiendo delle *osservazioni*
- Le osservazioni forniscono informazioni sulle variabili di stato osservabili

Belief State

- È l'insieme di stati del mondo in cui l'agente potrebbe essere
- È la distribuzione di probabilità condizionata sugli stati del mondo data la sequenza di azioni ed osservazioni effettuate
- Se $|\mathcal{S}| = 11$, il belief state è un vettore di 11 numeri reali:
 $\langle \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, 0, 0 \rangle$
- $b(s)$: prob. associata allo stato del mondo s dal belief state b
 $\forall s \in \mathcal{S}, 0 \leq b(s) \leq 1$, e $\sum_{s \in \mathcal{S}} b(s) = 1$.
- Lo spazio dei belief state è continuo

Esempio di Belief State



La sequenza (Left,Left,Left,Left,Left, Up, Up,Up,Up,Up, Right,Right, Right,Right,Right) raggiunge (3,4) con probabilità 77.5% ma scarso reward

Processo Decisionale di Markov Parzialmente Osservabili (POMDP)

- È un sequential decision problem in cui
 - l'ambiente è parzialmente osservabile
 - il modello di transizione è markoviano di ordine 1 e stazionario
 - i rewards sono additivi
- Un POMDP è descritto da:
 - Uno stato iniziale: s_0
 - Un modello di transizione: $T(S_t, A, S_{t+1})$
 - Un modello di osservazione: $O(S, O)$
 - Una funzione di reward: $R(S)$

Politica per POMDP

- L'azione *non* dipende dallo stato effettivo del mondo in cui si trova l'agente
- L'azione dipende solo dal belief state
- **Politica:** $\pi : \mathcal{B} \rightarrow \mathcal{A}$

```

Execute-policy( $\pi$ )
  Loop
     $a \leftarrow \pi(b)$ 
    Execute action  $a$ 
    Receive observation  $o$ 
    Compute the next belief-state Forward( $b, a, o$ )
    
```

- Forward(b, a, o): $b'(S') = \frac{1}{P(o)} O(S', o) T(S, a, S') b(S)$

POMDP to MDP

- Un POMDP può essere convertito in un MDP:
 - Uno stato iniziale: b_0
 - Un modello di transizione: $T(B_t, A, B_{t+1})$
 - Una funzione di reward: $R(B)$
- $T(b, a, b') = P(b'|a, b) = \sum_o P(b', o|a, b) = \sum_o P(b'|o, a, b) P(o|a, b)$
 - $P(o|a, b) = \sum_{s'} P(o|a, s', b) P(s'|a, b) = \sum_{s'} O(s', o) P(s'|a, b)$
 $= \sum_{s'} O(s', o) \sum_s T(s, a, s') b(s)$
 - $P(b'|o, a, b) = 1$ se $b' = \text{Forward}(b, a, o)$ e 0 altrimenti

POMDP to MDP

- $T(b, a, b') = \sum_o P(b'|o, a, b) \cdot \sum_{s'} O(s', o) \cdot \sum_s T(s, a, s') b(s)$

- $R(b) = \sum_s b(s) R(s)$

⇒ Risolvere un POMDP sugli stati del mondo corrisponde a risolvere un MDP sui belief state

- Si possono sviluppare versioni di policy-iteration e value-iteration per POMDP
- L'insieme dei belief state è rappresentato come un insieme di regioni di belief state
- La politica $\pi(b)$ associa un'azione ad una regione di belief state