Funzione di valutazione

Risolvere un problema di RL significa **trovare una policy che permette di ottenere compensi complessivi alti**. Per gli MDP finiti è possibile costruire una simile politica facendo sì che l'agente impari, con l'esperienza, a *valutare correttamente gli stati*

Funzione di valutazione

$V^{\pi}(s)$: stima la bontà dello stato *s* quando si usa la policy π

$$V^{\pi}(s) = \underbrace{E\{R_t | s_t = s\}}_{\text{Aspettativa di compenso futura se s è lo stato corrente}} = \underbrace{\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\}}_{\text{Vedi compenso atteso}}$$

State-value function

Funzione di valutazione

Non esiste un modo solo per effettuare le valutazioni.
Vediamo subito un'alternativa.

Funzione di valutazione

$Q^{\pi}(s,a)$: stima la bontà dell'applicazione di *a* nello stato *s* quando si usa la policy π

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

Aspettativa di compenso futuro se s è lo stato corrente e in s si esegue a

Vedi compenso atteso

Action-value function

Equazione di Bellman

Proprietà fondamentale delle funzioni di valutazione è una relazione ricorsiva nota come **equazione di Bellman** (fondamentale anche per il dynamic programming:

$$V^{\pi}(s) = E\{R_t | s_t = s\} = \{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} = \ldots$$

Non vediamo i passaggi

$$\ldots = \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^{\pi}(s')]$$

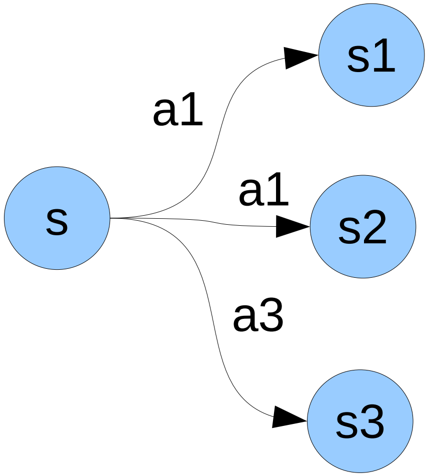
Equazione di Bellman

Proprietà fondamentale delle funzioni di valutazione è una relazione ricorsiva nota come equazione di Bellman (fondamentale anche per il dynamic programming:

$$V^{\pi}(s) = E\{R_t | s_t = s\} = \{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} = \ldots$$

Non vediamo i passaggi

$$\ldots = \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^{\pi}(s')]$$



Sia **s** lo stato di partenza. Siano **a1** e **a3** le azioni eseguibili in s e siano **s1**, **s2** e **s3** gli stati raggiungibili eseguendo in s tali azioni

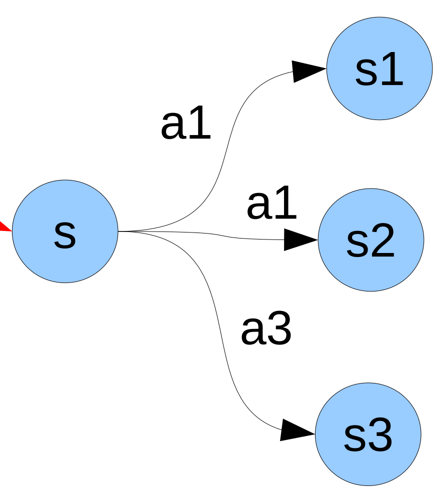
Equazione di Bellman

Proprietà fondamentale delle funzioni di valutazione è una relazione ricorsiva nota come equazione di Bellman (fondamentale anche per il dynamic programming:

$$V^{\pi}(s)=E\{R_t|s_t=s\}=\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t=s\}=\ldots$$

Non vediamo i passaggi

$$\ldots=\sum_{a}\pi(s,a)\sum_{s'}P^{a}_{ss'}[R^{a}_{ss'}+\gamma V^{\pi}(s')]$$



Sia **s** lo stato di partenza. Siano **a1** e **a3** le azioni eseguibili in s e siano **s1**, **s2** e **s3** gli stati raggiungibili eseguendo in s tali azioni

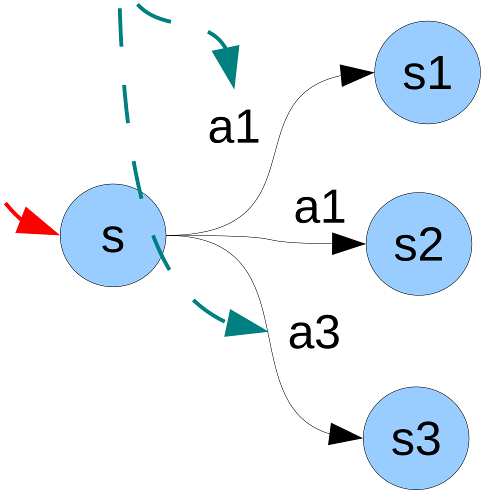
Equazione di Bellman

Proprietà fondamentale delle funzioni di valutazione è una relazione ricorsiva nota come equazione di Bellman (fondamentale anche per il dynamic programming):

$$V^{\pi}(s) = E\{R_t | s_t = s\} = \{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} = \ldots$$

Non vediamo i passaggi

$$\ldots = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'} [R^{a}_{ss'} + \gamma V^{\pi}(s')]$$



La prima sommatoria spazia sulle azioni eseguibili. Ricordiamoci che (s, a) è la *probabilità di eseguire l'azione* a essendo in s. La somma pesata dalle probabilità ci dà una media dei valori composti

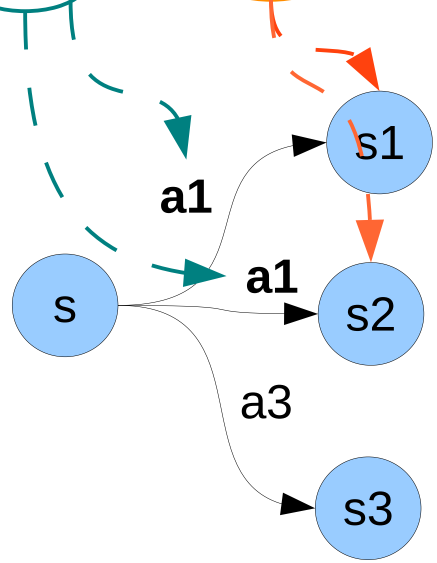
Equazione di Bellman

Proprietà fondamentale delle funzioni di valutazione è una relazione ricorsiva nota come equazione di Bellman (fondamentale anche per il dynamic programming):

$$V^{\pi}(s)=E\{R_t|s_t=s\}=\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t=s\}=\ldots$$

Non vediamo i passaggi

$$\ldots=\sum_{a}\pi(s,a)\sum_{s'}P_{ss'}^{a}[R_{ss'}^{a}+\gamma V^{\pi}(s')]$$



La seconda sommatoria produce un valore per ogni azione considerata dalla prima. Ogni azione può portare in più stati diversi (es. a1). Abbiamo quindi un contributo per ogni possibile stato successore, pesato dalla probabilità di entrare in quello stato

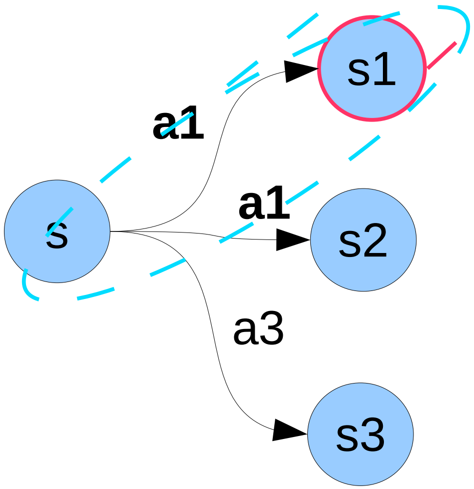
Equazione di Bellman

Proprietà fondamentale delle funzioni di valutazione è una relazione ricorsiva nota come equazione di Bellman (fondamentale anche per il dynamic programming):

$$V^{\pi}(s) = E\{R_t | s_t = s\} = \{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} = \ldots$$

Non vediamo i passaggi

$$\ldots = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'} [R^{a}_{ss'} + \gamma V^{\pi}(s')]$$



Per ogni possibile stato successore, si calcola un contributo sommando il rinforzo immediato ottenuto eseguendo l'azione che ha portato in quello stato e la valutazione della bontà dello stato raggiunto (di qui la ricorsione)

Problemi di RL e metodi per risolverli

- Problemi:
  - Problema di predizione: consiste nel costruire la valutazione di una policy
  - Problema di controllo: consiste nel costruire una policy
- La letteratura è molto ampia, fra gli approcci di base:
  - Metodi Monte Carlo (per problemi di natura episodica; e se un episodio non termina? Se impiega tanto tempo prima di terminare?)
  - Temporal Difference (usa state-value function)
  - Q-learning (usa action-value function)

Temporal Difference - TD(0)

- Nei metodi noti come **temporal difference (TD)** l'apprendimento avviene tramite **l'esperienza**
- Non occorre attendere il termine di un episodio per imparare perché l'apprendimento avviene tramite una forma di **bootstrap**
- Per questo sono adatti anche a problemi che **non hanno** natura episodica
- Il metodo che vediamo è il tipo più semplice di TD, detto **TD(0)**

Temporal Difference - TD(0)

Nel metodo Monte Carlo la valutazione di uno stato viene aggiornata quando l'episodio termina:

$$V(s_t) := (1-\alpha) V(s_t) + \alpha R_t, 0 \leq \alpha \leq 1$$

Una forma equivalente è:

$$V(s_t) := V(s_t) + \alpha [R_t - V(s_t)]$$

ma ...

Temporal Difference - TD(0)

Nel metodo Monte Carlo la valutazione di uno stato viene aggiornata quando l'episodio termina:

$$V(s_t) := (1-\alpha) V(s_t) + \alpha R_t, 0 \leq \alpha \leq 1$$

Una forma equivalente è:

$$V(s_t) := V(s_t) + \alpha [R_t - V(s_t)]$$

ma ...

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

Temporal Difference - TD(0)

$$V(s_t) := V(s_t) + \alpha[R_t - V(s_t)]$$

$$R_t = r_{t+1} + \gamma V(s_{t+1})$$

Possiamo quindi riscrivere la regola di aggiornamento nel seguente modo:

$$V(s_t) := V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

La valutazione di uno stato ...

... viene raffinata

usando il rinforzo immediato ...

... e la differenza fra la stima corrente della valutazione dello stato e la stima della valutazione dello stato successivo

http://incompleteideas.net/sutton/book/ebook/node61.html

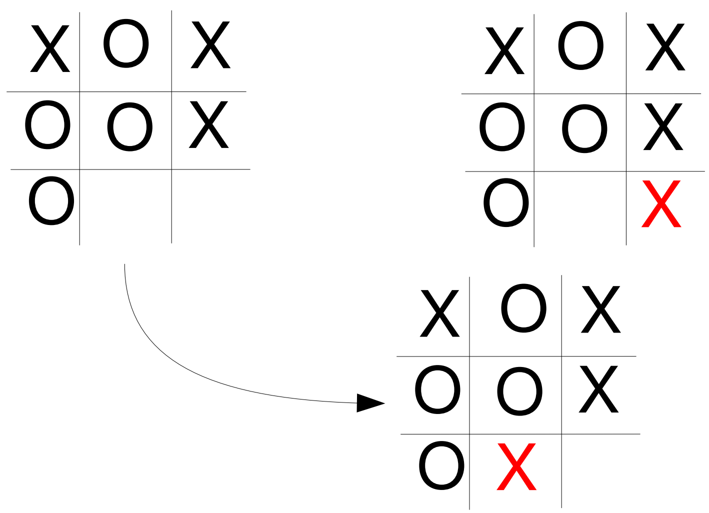
TD(0): algoritmo

Problema di predizione

Siano:
π: la policy da valutare
V: una funzione di valutazione arbitraria

Repeat per ogni episodio:
    Inizializza s
    Repeat per ogni passo s dell'episodio:
      (1) a := azione restituita da **π** per s
      (2) esegui a
      (3) osserva il rinforzo immediato r e lo stato successivo s'
      (4) $V(s) := V(s) + \alpha[\, r + \gamma\, V(s') - V(s)]$
      (5) s := s'
    end
end

Agenti greedy ed esplorazione



Tocca all'agente X, che ha due possibili azioni fra cui scegliere: una porta a Vincere, l'altra a pareggiare

Rinforzo vittoria: +1
Rinforzo pareggio: -1

Se nel primo episodio, X sceglie a caso e pareggia, il penultimo stato avrà una cattiva valutazione.

Un agente puramente greedy non è esplorativo e non riesce a individuare le politiche migliori

Soluzione: tutti gli algoritmi di tanto in tanto abbandonano la politica greedy e scelgono a caso l'azione da eseguire, anche se non è promettente secondo la valutazione corrente.