Apprendimento per Rinforzo

Corso di Apprendimento Automatico

Laurea Magistrale in Informatica

Nicola Fanizzi

Dipartimento di Informatica
Università degli Studi di Bari

11 febbraio 2009

Sommario

- Apprendimento del Controllo
- Politiche di Controllo che scelgono azioni ottimali
- Q-learning
- Convergenza
- Estensioni

Apprendimento del Controllo I

Si consideri di imparare a scegliere *azioni* da intraprendere, es.,

- Robot che impara a parcheggiare sulla postazione di ricarica delle batterie
- Imparare a scegliere azioni in modo da ottimizzare la produzione di una fabbrica
- Imparare a giocare a giochi come scacchi, dama, backgammon, ...

Apprendimento del Controllo II

Osservazioni

Si notino diverse caratteristiche del problema rispetto alle altre forme di learning trattate:

- Ricompensa differita
- Opportunità di esplorazione attiva
- Possibilitïà di stati solo parzialmente osservabili
- Possibilità di dover apprendere compiti multipli tramite gli stessi sensori/effettuatori

Esempio: TD-Gammon

Obiettivo: Imparare a giocare a Backgammon

Ricompensa immediata

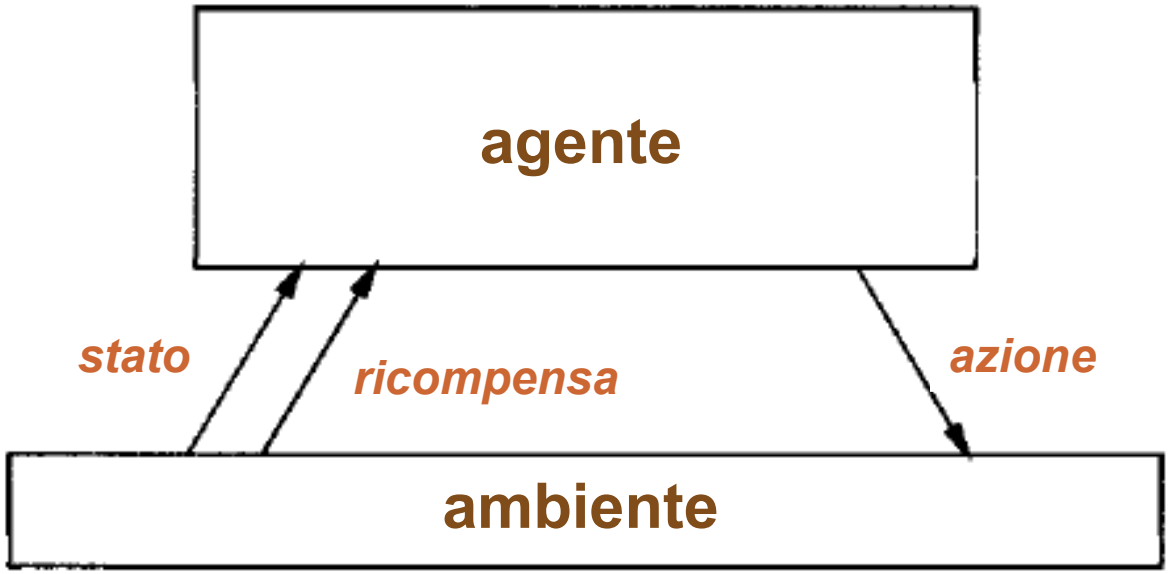
- $+100$ se si vince
- -100 se si perde
- 0 per tutti gli altri stati

Sistema addestrato giocando 1.5M partite contro se stesso

Alla fine il sistema risulta avere approssimativamente prestazioni paragonabili al *miglior* giocatore umano

[Tesauro, 1995]

Problema dell'Apprendimento per Rinforzo



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

***Obiettivo*: scegliere le azioni che massimizzino**

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \text{ , dove } 0 \leqslant \gamma < 1$$

Processi di decisione di Markov (MDP)

Si assuma:

- un insieme di **stati** S finito
- un insieme di **azioni** A
- ad ogni istante discreto
  - l'agente osserva lo stato $s_t \in S$ e sceglie l'azione $a_t \in A$
  - quindi riceve una ricompensa immediata r_t
  - e lo stato corrente diventa s_{t+1}
- Assunzione di **Markov**: $s_{t+1} = \delta(s_t, a_t)$ e $r_t = r(s_t, a_t)$
  - ossia, r_t e s_{t+1} dipendono solo dallo stato *corrente* e dall'azione intrapresa
  - le funzioni δ e r potrebbe non essere deterministica
  - le funzioni δ e r non sono necessariamente note all'agente

Compito di apprendimento per l'agente

Si eseguono le azioni nell'ambiente, si osservano i risultati e

- si impara una **politica** di azioni $\pi : S \rightarrow A$ che massimizzi

$$E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots]$$

  a partire da qualunque stato di partenza in S
- qui $0 \leq \gamma < 1$ è il *tasso di sconto* per ricompensa future

Novità

- Funzione obiettivo $\pi : S \rightarrow A$
- ma non ci sono esempi di training del tipo $\langle s, a \rangle$
  bensi gli esempi di training sono invece del tipo $\langle \langle s, a \rangle, r \rangle$

Funzione di valutazione

Per cominciare si considerino mondi deterministici ...

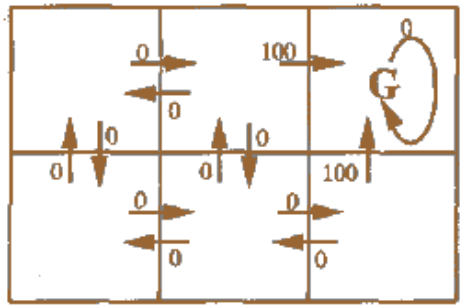
Per ogni possibile politica π che l'agente potrebbe adottare, si può definire una **funzione di valutazione** sull'insieme degli stati

$$\begin{aligned} V^{\pi}(s) &\equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \\ &\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i} \end{aligned}$$

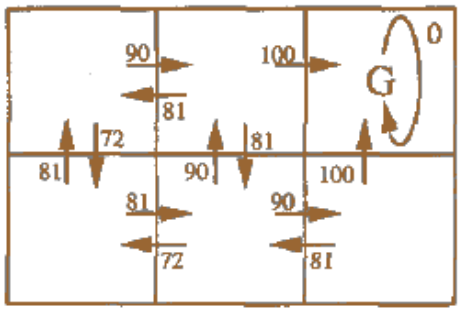
dove le $r_t, r_{t+1}, \ldots$ sono generate seguendo la politica π a partire dallo stato s

In altri termini,
il task consiste nell'apprendere la politica ottimale π^*

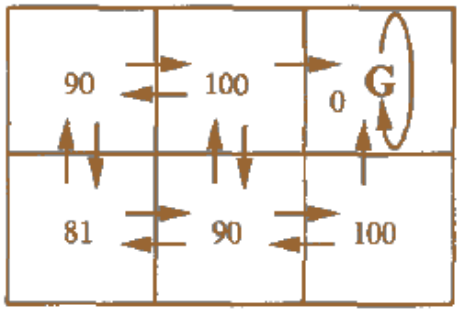
$$\pi^* \equiv \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(s) \qquad \forall s$$



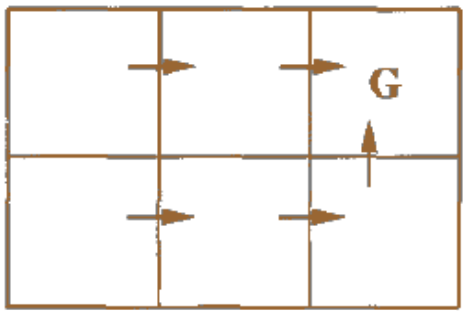
Valori di *r(s,a)* (ricompensa immediata)



Valori di *Q(s,a)*



Valori di *V*(s)*



Strategia ottimale

Cosa imparare

Si può provare a far imparare la funzione di valutazione V^{π^*} (che si denoter anche V^*)

Si può operare una ricerca in avanti (*lookahead*) per scegliere la migliore azione a partire da ogni stato s poiché

$$\pi^*(s) = \operatorname{argmax}_a [r(s,a) + \gamma V^*(\delta(s,a))]$$

Problema:

- Funziona bene se l'agente conosce le funzioni $\delta : S \times A \to S$, e $r : S \times A \to \Re$
- ma quando questo non accade
  non si possono scegliere le azioni in questo modo

Funzione Q

Si definisce una nuova funzione simile a V^*

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a))$$

Se l'agente impara la funzione Q,
si potrà scegliere l'azione ottimale anche senza conoscere δ

$$\pi^*(s) = \operatorname*{argmax}_a [r(s,a) + \gamma V^*(\delta(s,a))]$$

$$\pi^*(s) = \operatorname*{argmax}_a Q(s,a)$$

Q è la funzione di valutazione che l'agente dovrà imparare

Regola di training per imparare Q

Si noti che Q e V^* sono strettamente legate:

$$V^*(s) = \max_{a'} Q(s, a')$$

il che permette di riscrivere Q in modo ricorsivo:

$$\begin{aligned} Q(s_t, a_t) &= r(s_t, a_t) + \gamma V^*(\delta(s_t, a_t))) \\ &= r(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') \end{aligned}$$

Denotata con $\hat{Q}$ l'approssimazione corrente si Q, si consideri la regola di training:

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

dove s' è lo stato risultante dall'applicazione dell'azione a nello stato s

Q-Learning per mondi deterministici

Per ogni s, a inizializzare la cella della tabella: $\hat{Q}(s, a) \leftarrow 0$

Sia s lo stato corrente

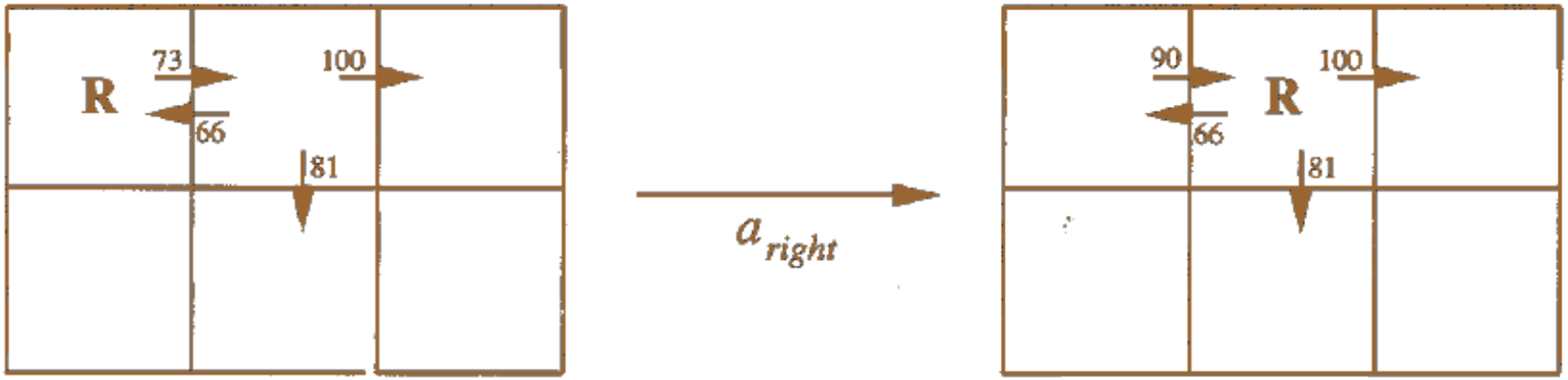
Ripeti:

- Selezionare un'azione a ed eseguirla
- Ricevere la ricompensa immediata r
- Sia s' il nuovo stato
- Aggiornare l'elemento in tabella $\hat{Q}(s, a)$ come segue:

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$

Aggiornamento di $\hat{Q}$ I



Stato iniziale s_1 Stato finale s_2

$$\hat{Q}(s_1, a_{right}) \leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a')$$
$$\leftarrow 0 + 0.9 \max\{66, 81, 100\}$$
$$\leftarrow 90$$

Aggiornamento di $\hat{Q}$ II

Si noti che se le ricompense sono non negative, allora

$$(\forall s, a, n) \;\; \hat{Q}_{n+1}(s, a) \geq \hat{Q}_n(s, a)$$

e

$$(\forall s, a, n) \;\; 0 \leq \hat{Q}_n(s, a) \leq Q(s, a)$$

Convergenza I

Teorema $\hat{Q}$ converge a Q.
Si considera il caso di un mondo deterministico dove ogni $\langle s, a \rangle$ sia visitato infinite volte

Dim.:
Definire un intervallo pieno durante il quale $\langle s, a \rangle$ viene visitato.
Durante ogni intervallo pieno l'errore piu' grande nella tabella $\hat{Q}$ si riduce del fattore γ

Sia $\hat{Q}_n$ la tabella ottenuta dopo n aggiornamenti e
Δ_n l'errore massimo in $\hat{Q}_n$; ossia:

$$\Delta_n = \max_{s,a} |\hat{Q}_n(s, a) - Q(s, a)|$$

Convergenza II

Per ogni elemento della tabella $\hat{Q}_n(s,a)$ aggiornando all'iterazione $n+1$, l'errore nella nuova stima $\hat{Q}_{n+1}(s,a)$ sarà:

$$
\begin{aligned}
|\hat{Q}_{n+1}(s,a) - Q(s,a)| &= |(r + \gamma \max_{a'} \hat{Q}_n(s',a')) \\
&\quad -(r + \gamma \max_{a'} Q(s',a'))| \\
&= \gamma |\max_{a'} \hat{Q}_n(s',a') - \max_{a'} Q(s',a')| \\
&\leq \gamma \max_{a'} |\hat{Q}_n(s',a') - Q(s',a')| \\
&\leq \gamma \max_{s'',a'} |\hat{Q}_n(s'',a') - Q(s'',a')| \\
|\hat{Q}_{n+1}(s,a) - Q(s,a)| &\leq \gamma \Delta_n
\end{aligned}
$$

Convergenza III

Si noti che si ricorre alla proprietà seguente:

$$|\max_a f_1(a) - \max_a f_2(a)| \leq \max_a |f_1(a) - f_2(a)|$$

Caso non deterministico I

Che succede se la ricompensa e lo stato successivo non sono deterministici ?

Si ridefiniscono V e Q considerando i valori attesi

$$\begin{aligned} V^{\pi}(s) &\equiv E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots] \\ &\equiv E[\sum_{i=0}^{\infty} \gamma^i r_{t+i}] \end{aligned}$$

$$Q(s,a) \equiv E[r(s,a) + \gamma V^*(\delta(s,a))]$$

Caso non deterministico II

Il Q-learning si estende a mondi non deterministici

Si modifica la regola di training

$$\hat{Q}_n(s,a) \leftarrow (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \alpha_n[r + \max_{a'} \hat{Q}_{n-1}(s',a')]$$

dove

$$\alpha_n = \frac{1}{1 + visits_n(s,a)}$$

Si può comunque provare la convergenza di $\hat{Q}$ a Q

[Watkins & Dayan, 1992]

Temporal Difference Learning

Q-learning: ridurre la *discrepanza* tra stime successive di Q

Differenza di un passo:

$$Q^{(1)}(s_t, a_t) \equiv r_t + \gamma \max_a \hat{Q}(s_{t+1}, a)$$

Due passi:

$$Q^{(2)}(s_t, a_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 \max_a \hat{Q}(s_{t+2}, a)$$

Per n passi:

$$Q^{(n)}(s_t, a_t) \equiv r_t + \gamma r_{t+1} + \cdots + \gamma^{(n-1)} r_{t+n-1} + \gamma^n \max_a \hat{Q}(s_{t+n}, a)$$

Mettendo tutto insieme:

$$Q^{\lambda}(s_t, a_t) \equiv (1-\lambda) \left[Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t) + \cdots \right]$$

Particolarità e sviluppi possibili

- Cambiare la tabella $\hat{Q}$ con una rete neurale o altri sistemi di generalizzazione
- Trattare il caso di stati solo parzialmente osservabili
- Progettare strategie ottime di esplorazione
- Estendere al caso di azioni continue (stati continui)
- Imparare ad usare $\hat{\delta} : S \times A \rightarrow S$
- Relazione con la programmazione dinamica

Fonti

- T. M. Mitchell: *Machine Learning*, McGraw Hill