Reinforcement Learning nel caso multiagente: NashQ-learning e FFQ-learning.

Giuseppe Bruno

Agosto 2022

Indice

Sommario

Dopo aver richiamato le nozioni fondamentali del Reinforcement Learning e dei giochi stocastici, viene introdotta una generalizzazione del Q-learning a giochi multiagente noncooperativi: il NashQ-learning. La dimostrazione della convergenza del procedimento iterativo richiede ipotesi restrittive, indebolite in una sua variante nota come FFQ-learning. Infine il comportamento degli algoritmi presentati è discusso attraverso due esempi.

1 Reinforcement Learning

1.1 Markov Decision Processes

Il Reinforcement Learning è un'area del machine learning con lo scopo di determinare come e quali azioni un agente deve compiere al fine di massimizzare le proprie ricompense in un ambiente, tramite le interazioni con esso.

Tale ambiente è formalizzato attraverso il concetto di *Markov Decision Process*:

Definizione 1.1. *Un Markov Decision Process (MDP) è il dato di di* $(\mathcal{S}, \mathcal{A}, r, p)$ *dove* $\mathcal{S}$ *è lo spazio discreto degli stati,* $\mathcal{A}$ *è lo spazio discreto delle azioni,* $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *è la funzione ricompensa dell'agente e* $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ *è la funzione di transizione.*

L'agente al tempo t si trova in uno stato $S_t \in \mathcal{S}$, compie un'azione $A_t \in \mathcal{A}$, al tempo successivo riceve una ricompensa R_{t+1} e si sposta nello stato S_{t+1} secondo la probabilità di transizione p. Dunque si hanno dei processi stocastici a valori nei rispettivi spazi che si susseguono nel seguente modo:

$$S_0, A_0, R_1, S_1, A_1, R_2, ...$$

In particolare il termine *Markov* si riferisce al fatto (implicato dalla definizione stessa della funzione di transizione) che la ricompensa R_{t+1} e lo stato S_{t+1} dipendono soltanto dall'azione e dallo stato al tempo t presente e non dal passato.

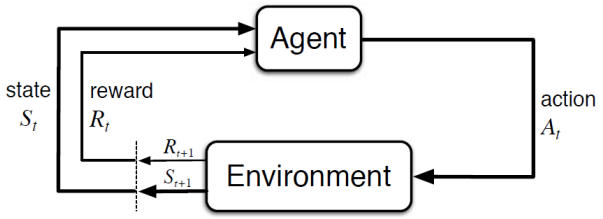


Figura 1: Interazione ambiente-agente in un MDP.

1.2 Value function e Q-function

L'obiettivo dell'agente è quello di trovare una strategia che gli consenta di massimizzare le ricompense a "lungo termine".

Per formalizzare ciò nel contesto dei MDP appena introdotti occorrono delle nuove definizioni:

Definizione 1.2. *Una policy (strategia) stazionaria è una funzione* $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ *che ad ogni stato assegna la probabilità che un'azione venga compiuta.*

Osservazione 1.3. *Si potrebbero considerare anche strategie non stazionarie ma ciò complicherebbe notevolmente la trattazione e non sarebbe necessario ai nostri fini grazie al teorema 2.3.*

Definizione 1.4. *La ricompensa cumulativa scontata di un fattore $\beta \in [0,1)$ è definita come $G_t := \sum_{k=0}^{\infty} \beta^k R_{t+k+1}$.*

Osservazione 1.5. *Il fattore β consente di quantificare il "lungo termine". Ad esempio se $\beta = 0$ l'agente è detto miope ed è interessato soltanto alle ricompense immediate.*

L'obiettivo diventa dunque trovare una *policy* π che massimizzi la speranza della ricompensa cumulativa G_t, condizionata al seguire tale strategia durante il gioco. Si definisce pertanto la *value function* $v(s,\pi)$ da massimizzare come:

$$v(s,\pi) = v_\pi(s) := \mathbb{E}[G_t | S_t = s, \pi] = \mathbb{E}\left[\sum_{k=0}^{\infty} \beta^k R_{t+k+1} \middle| S_t = s, \pi\right].$$

Allo stesso modo è possibile definire l'*action-value function* $Q_\pi(s,a)$ per lo stato s e l'azione a sotto la policy π come:

$$\begin{aligned} Q_\pi(s,a) &:= \mathbb{E}\left[\sum_{k=0}^{\infty} \beta^k R_{t+k+1} \middle| S_t = s, A_t = a, \pi\right] = \\ &= \mathbb{E}\left[R_{t+1} + \beta v_\pi(S_{t+1}) \middle| S_t = s, A_t = a, \pi\right]. \end{aligned} \tag{1}$$

Permette di quantificare la bontà di un'azione a in un certo stato s al tempo t, supponendo che dal tempo $t+1$ l'agente seguirà la policy π. In particolare tra le due funzioni definite finora intercorre la seguente relazione:

$$v_\pi(s) = \sum_a Q_\pi(s,a)\ \pi(a|s).$$

Sull'insieme delle *policy* è possibile definire un ordinamento parziale, affermando che $\pi \geq \pi'$ se e solo se $v_\pi(s) \geq v'_\pi(s)\ \forall s \in \mathcal{S}$. Da quanto seguirà si può ricavare che esiste sempre almeno una *policy* π_* ottimale. Si definiscono inoltre la *optimal state-value function* v_* come:

$$v_*(s) := \max_\pi v_\pi(s) \quad \forall s \in \mathcal{S},$$

e la *optimal action-value function* Q_*:

$$Q_*(s,a) := \max_\pi Q_\pi(s,a) \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}.$$

1.3 Equazione di Bellman

Il calcolo diretto delle funzioni appena introdotte è spesso difficile. L'approccio più conveniente e più utilizzato sfrutta invece la cosiddetta *equazione di Bellman* che permette di ricondursi ad un problema di punto fisso.

Dalla definizione di speranza condizionale si ottiene immediatamente che:

$$\begin{aligned}
v_\pi(s) &= \mathbb{E}\left[G_t | S_t = s, \pi\right] = \\
&= \mathbb{E}\left[R_{t+1} + \beta G_{t+1} | S_t = s, \pi\right] = \\
&= \sum_a \pi(a|s)\ \mathbb{E}\left[R_{t+1} + \beta G_{t+1} | S_t = s, A_t = a, \pi\right] = \\
&= \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)(r(s,a) + \beta \mathbb{E}[G_{t+1} | \underbrace{S_{t+1} = s', \pi}_{Markov}]) = \\
&= \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)\left(r(s,a) + \beta v_\pi(s')\right).
\end{aligned}$$

Ovvero:

$$v_\pi(s) = \sum_a \pi(a|s) \left(r(s,a) + \beta \sum_{s'} p(s'|s,a) v_\pi(s') \right). \tag{2}$$

Lemma 1.6. *Sia $B(\mathcal{S}) = \{v : \mathcal{S} \to \mathbb{R} : ||v||_\infty < +\infty\}$. Data una policy π si definisce l'operatore di Bellman $T^\pi : B(\mathcal{S}) \to B(\mathcal{S})$ nel seguente modo[1]:*

$$(T^\pi v)(s) = r(s, \pi(s)) + \beta \sum_{s'} p(s'|s, \pi(s)) v(s') \quad s \in \mathcal{S}.$$

Allora T^π è una β-contrazione con punto fisso dato da v_π.

Dimostrazione. Il fatto che v_π sia un punto fisso per T^π è una conseguenza immediata dell'equazione (2). Inoltre si tratta di una contrazione rispetto a $||\cdot||_\infty$ perché date $u, v \in B(\mathcal{S})$ vale:

$$\begin{aligned}
||T^\pi u - T^\pi v||_\infty &= \beta \max_s \left| \sum_{s'} p(s'|s, \pi(s))(u(s') - v(s')) \right| \\
&\leq \beta \max_s \sum_{s'} p(s'|s, \pi(s)) \left| u(s') - v(s') \right| \\
&\leq \beta \max_s \sum_{s'} p(s'|s, \pi(s)) ||u - v||_\infty \\
&\leq \beta ||u - v||_\infty,
\end{aligned}$$

ovvero la tesi. □

Osservazione 1.7. *Il precedente lemma ha importanti conseguenze dal punto di vista algoritmico. Grazie al teorema del punto fisso di Banach sullo spazio metrico completo $B(\mathcal{S})$, è possibile infatti valutare numericamente v_π tramite le iterazioni $v_0, T^\pi v_0, (T^\pi)^2 v_0, ...$, con v_0 inizializzata randomicamente. Tale procedura è nota anche come policy evaluation.*

[1] Per semplicità di notazione si considerano qui policy derteministiche, ma la dimostrazione è analoga per policy stazionarie qualsiasi.

Una volta risolto il problema del calcolo di v_π si può determinare con un approccio simile anche v_*.

Applicando infatti l'equazione (1) a v_* si ottiene per monotonia della speranza condizionale:

$$Q_*(s,a) = \mathbb{E}\left[R_{t+1} + \beta v_*(S_{t+1}) \middle| S_t = s, A_t = a, \pi_*\right]. \tag{3}$$

Dunque:

$$v_*(s) = \max_a Q_*(s,a) = \max_a \mathbb{E}\left[R_{t+1} + \beta v_*(S_{t+1}) \middle| S_t = s, A_t = a, \pi_*\right],$$

da cui si ricava l'*equazione di ottimalità di Bellman* per v_*:

$$v_*(s) = \max_a \left\{ r(s,a) + \beta \sum_{s'} p(s'|s,a) v_*(s') \right\}, \tag{4}$$

e per Q_*:

$$Q_*(s,a) = r(s,a) + \beta \sum_{s'} p(s'|s,a) \max_{a'} Q_*(s',a'). \tag{5}$$

In modo analogo a quanto dimostrato nel lemma 1.6 si ha che anche l'*optimality operator di Bellman* T^*, definito come:

$$(T^* v)(s) = \max_a \left\{ r(s,a) + \beta \sum_{s'} p(s'|s,a) v(s') \right\}, \quad s \in \mathcal{S},$$

è una β-contrazione su $B(\mathcal{S})$.

L'importanza della valutazione di v_* o Q_* è riassunta nel teorema che segue. La conoscenza di tali funzioni equivale infatti a risolvere il problema introdotto finora, dato che una strategia ottimale si può ottenere semplicemente scegliendo una strategia *greedy* rispetto ai valori calcolati.

Teorema 1.8. *Sia v punto fisso dell'operatore T^* e sia π una policy greedy rispetto a v (ovvero $T^\pi v = T^* v$). Allora $v = v^*$ e π è una policy ottimale.*

Dimostrazione. Sia π policy qualsiasi. Allora in generale si osserva che $T^\pi \leq T^*$ da cui:

$$v^\pi = T^\pi v^\pi \leq T^* v^\pi,$$

e, dato che $T^* u \leq T^* v$ se $u \leq v$, applicando n volte T^* alla disuguaglianza appena ottenuta si ottiene:

$$v^\pi \leq (T^*)^n v^\pi.$$

Grazie alla proprietà di contrazione di T^* il membro di destra converge all'unico punto fisso v. Ciò prova che $v^\pi \leq v$ per qualsiasi policy π.

Se adesso π è una policy greedy per v, ossia se $T^\pi v = T^* v$, allora si può concludere che $v = T^* v = T^\pi v$, ma per unicità del punto fisso di T^π deve valere $v = v^\pi$. Pertanto $v = v^*$ e π è una policy ottimale. □

Osservazione 1.9. *Il teorema appena dimostrato fornisce un algoritmo per ottenere un'optimal policy per il problema considerato. Ancora una volta grazie al teorema del punto fisso di Banach, è infatti possibile ricavare v^* tramite le iterazioni $v_0, T^*v_0, (T^*)^2v_0, \dots$, con v_0 inizializzata randomicamente. Infine si considera una policy greedy π^* rispetto a v^*. Tale procedura è nota anche come value-iteration.*

1.4 Q-Learning

Sebbene quanto presentato finora sia una soluzione semplice ed efficace per la determinazione di una *policy* ottimale, essa richiede una conoscenza perfetta dell'*ambiente* in cui si trova l'agente, in particolare delle ricompense e delle dinamiche dell'ambiente stesso. Negli ultimi decenni gli studi relativi al reinforcement learning si sono concentrati sui casi in cui tale conoscenza viene meno ed è quindi necessario sfruttare risultati probabilistici per permettere all'agente di apprendere tramite l'esperienza.

Numerosi sono i risultati ottenuti e gli approcci possibili: metodi Monte Carlo, metodi TD, ricerca diretta... Tuttavia per i nostri scopi ci si soffermerà su uno degli algoritmi più studiati, il *Q-learning*, introdotto da C. Watkins nel 1992.

L'agente inizia con valori arbitrari per $Q(s,a)\ \forall s \in \mathcal{S},\ \forall a \in \mathcal{A}$ e durante l'esplorazione dell'ambiente aggiorna i valori come segue:

$$Q_{t+1}(s_t,a_t) = (1-\alpha_t)Q_t(s_t,a_t) + \alpha_t[r_t + \beta \max_a Q_t(s_{t+1},a)], \tag{6}$$

dove $\alpha_t \in [0,1)$ è la successione dei learning rates. La regola di aggiornamento sorge spontanea dall'equazione (5) ed è resa rigorosa dal seguente teorema (dimostrato da Watkins stesso in [10]):

Teorema 1.10. *Se le ricompense r_t sono limitate e la successione dei learning rates verifica:*

- $\alpha_t \in [0,1)$,
- $\sum_{t=0}^{\infty} \alpha_t = \infty$,
- $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$,

allora $Q_t(s,a)$ converge a $Q^*(s,a)\ \forall s \in \mathcal{S}, a \in \mathcal{A}$ quasi certamente.

L'algoritmo è riassunto in 1.

2 NashQ-Learning

L'algoritmo Q-learning così presentato è inadatto ad affrontare il problema del reinforcement learning in un contesto multiagente, in cui l'ambiente non è più stazionario. Tale non stazionarietà infatti deriva non soltanto dalla componente stocastica, ma anche dalla razionalità degli altri agenti che si adattano alle

Algorithm 1 Q-learning

Fissa $\alpha \in (0,1]$, $\varepsilon > 0$.
Inizializza $Q(s,a)$ $\forall s \in \mathcal{S}, a \in \mathcal{A}$. Inizializza s.
while s non è terminale **do**
  Scegli a per lo stato s tramite la policy ε-greedy derivata da Q
  Compi l'azione a ed osserva r,s'
  $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \beta \max_a Q(s',a) - Q(s,a)]$
  $s \leftarrow s'$
end while

strategie scelte. Nel calcolare la ricompensa attesa occorre prendere atto di tale razionalità e supporre che gli altri agenti attuino le migliori strategie a loro disposizione, dove il termine "migliori" è reso formalmente dal concetto di equilibrio di Nash nei giochi stocastici.

2.1 Giochi stocastici

Per modellizzare il caso di sistemi multi-agenti noncooperativi e a tempi discreti si utilizzerà la nozione di *giochi stocastici*, introdotta da L. Shapley nel 1953:

Definizione 2.1. *Un gioco stocastico con n giocatori è il dato di:*

$$(\mathcal{S}, A^1, ..., A^n, r^1, ..., r^n, p),$$

dove $\mathcal{S}$ è lo spazio degli stati, A^i è l'insieme delle azioni per il giocatore i, $r^i : \mathcal{S} \times A^1 \times ... \times A^n \to \mathbb{R}$ è la funzione di utilità per il giocatore i, $p : \mathcal{S} \times A^1 \times ... \times A^n \to \Delta(\mathcal{S})$ è la probabilità di transizione, con $\Delta(\mathcal{S})$ l'insieme delle distribuzioni di probabilità su $\mathcal{S}$.

Dato uno stato s, gli agenti scelgono indipendentemente le azioni $a^1, ..., a^n$ e ricevono le ricompense $r^i(s, a^1, ..., a^n)$ per $i = 1, ..., n$. Infine il nuovo stato diventa s' secondo la probabilità di transizione fissata.

In un *discounted stochastic game* l'obiettivo di ogni giocatore è massimizzare la speranza condizionale della propria ricompensa cumulativa scontata. Ovvero se π^i è la strategia del giocatore i ed s è lo stato iniziale, allora il giocatore i proverà a massimizzare:

$$v^i(s, \pi^1, .., \pi^n) := \sum_{t=0}^{\infty} \beta^t \mathbb{E}[r_t^i | \pi^1, ..., \pi^n, s_0 = s].$$

Si richiama qui anche la definizione di *equilibrio di Nash* nel contesto dei giochi stocastici:

Definizione 2.2. *In un gioco stocastico Γ, un equilibrio di Nash è il dato di n strategie $(\pi_*^1, ..., \pi_*^n)$ tale che $\forall s \in S$ e $i = 1, .., n$,*

$$v^i(s, \pi_*^1, ..., \pi_*^n) \geq v^i(s, \pi_*^1, ..., \pi_*^{i-1}, \pi^i, \pi_*^{i+1}, ..., \pi_*^n) \quad \forall \pi^i \in \Pi^i,$$

dove Π^i è l'insieme delle strategie possibili per l'agente i.

Di fondamentale importanza per semplificare la trattazione, come già annunciato, è il seguente teorema di A. Fink [3]:

Teorema 2.3. *Ogni gioco stocastico a $n-$giocatori possiede almeno un equilibrio di Nash a strategie stazionarie.*

2.2 NashQ-Values

Per adattare il Q-learning al caso di un sistema di $n-$agenti occorre estendere la definizione di *Q-function*, definendo la *NashQ-value* Q^i_* come la speranza condizionale della ricompensa cumulativa sapendo che gli agenti seguiranno le strategie di uno specificato equilibrio di Nash. Più precisamente:

Definizione 2.4. *La NashQ-function dell'agente i è definita come:*

$$Q^i_*(s, a^1, ..., a^n) = r^i(s, a^1, ..., a^n) + \beta \sum_{s'} p(s'|s, a^1, ..., a^n) v^i(s', \pi^1_*, ..., \pi^n_*),$$

dove $(\pi^1_*, ..., \pi^n_*)$ è un equilibrio di Nash, $v^i(s', \pi^1_*, ..., \pi^n_*)$ è la ricompensa totale scontata a partire dallo stato s' sapendo che gli agenti seguiranno le strategie dell'equilibrio.

La differenza principale con quanto discusso finora sta nel fatto che l'agente non può più limitarsi ad aggiornare la propria *Q-function* massimizzando la propria utilità, ma deve considerare le utilità in un futuro equilibrio di Nash.

Un gioco stocastico può essere formato da più episodi, ognuno dei quali è detto *stage game* (ad esempio ogni turno di un MDP):

Definizione 2.5. *Uno stage game a $n-$giocatori è definito come $(M^1, ..., M^n)$ dove M^k è la funzione utilità per il giocatore k sullo spazio prodotto delle azioni.*

2.3 Algoritmo

Il giocatore i di riferimento apprenderà nel seguente modo: inizializza arbitrariamente $Q^i_0(s, a^1, ..., a^n)$ $\forall s \in S, a^1 \in A^1, ..., a^n \in A^n$. Al tempo t l'agente i nello stato corrente compie la sua azione ed osserva le azioni prese dagli altri agenti, le ricompense ed il nuovo stato s'. Dopo di ciò calcola un equilibrio di Nash $\pi^1(s'), ..., \pi^n(s')$ per lo stage game $(Q^1_t(s'), ..., Q^n_t(s'))$ ed aggiorna la sua *Q-function* nel seguente modo:

$$Q^i_{t+1}(s, a^1, ..., a^n) = (1 - \alpha_t) Q^i_t(s, a^1, ..., a^n) + \alpha_t [r^i_t + \beta \; NashQ^i_t(s')], \qquad (7)$$

dove[2]

$$NashQ^i_t(s') = Q^i_t(s', \pi^1(s'), ..., \pi^n(s')).$$

Per poter calcolare l'equilibrio di Nash l'agente i dovrebbe conoscere le *Q-function* $Q^1_t(s'), ..., Q^n_t(s')$ che però non sono note, pertanto sarà necessario apprendere anche queste tramite la stessa regola di aggiornamento.

[2]Qui e successivamente per semplicità di notazione si utilizzerà la notazione deterministica $Q^i(s, \pi^1(s), ..., \pi^n(s))$ sottintendendo la speranza: $\sum_{a^1,...,a^n} \pi^1(a^1|s), ..., \pi^n(a^n|s) Q^i(s, a^1, ..., a^n)$

Algorithm 2 NashQ-learning

Fissa $\alpha \in (0, 1]$.
Per ogni $s \in S, a^j \in A^j$ con $j = 1, ..., n$, fissa $Q_t^j(s, a^1, ..., a^n) = 0$
while s non è terminale **do**
  Scegli a_t^i per lo stato s
  Osserva $r_t^1, ..., r_t^n,\ a_t^1, ..., a_t^n$ e $s_{t+1} = s'$
  Aggiorna Q_t^j per $j = 1, ..., n$ tramite l'eq. 7
end while

Osservazione 2.6. *Il costo computazionale dell'algoritmo è dominato dal calcolo degli equilibri di Nash, che nei casi peggiori può avere complessità esponenziale.*

2.4 Convergenza

Le ipotesi necessarie affinchè la successione $(Q_t^1, ..., Q_t^n)$ converga alle *NashQ-functions* $(Q_*^1, ..., Q_*^n)$ sono tuttora oggetto di ricerca. Tre assunzioni sufficienti, seppur restrittive, vennero introdotte da Hu e Wellman nel loro articolo originale [5] e poi corrette da M. Bowling in [1], [4].

Due di tali ipotesi sono simili a quelle per il classico single-agent Q-learning:

1. Ogni stato $s \in \mathcal{S}$ e azione $a^k \in A^k$ per $k = 1, ..., n$ è visitato infinite volte;
2. il learning rate α_t soddisfa le seguenti proprietà:
   - $\alpha_t \in [0, 1)$,
   - $\sum_{t=0}^{\infty} \alpha_t = \infty$,
   - $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$.

La terza ipotesi richiede invece l'introduzione di alcune definizioni:

Definizione 2.7. *Le strategie $(\sigma^1, ..., \sigma^n)$ sono un punto ottimale globale per lo stage game $(M^1, ..., M^n)$ se $\forall k$*

$$M^k(\sigma) \geq M^k(\hat{\sigma}) \quad \forall \hat{\sigma} \in \sigma(A).$$

Osservazione 2.8. *Ogni punto ottimale globale è un equilibrio di Nash e tutti tali punti hanno la stessa utilità.*

Definizione 2.9. *Le strategie $(\sigma^1, ..., \sigma^n)$ sono un punto di sella per lo stage game $(M^1, ..., M^n)$ se sono un equilibrio di Nash e se $\forall k$:*

$$M^k(\sigma^k \sigma^{-k}) \geq M^k(\hat{\sigma}^k \sigma^{-k}) \quad \forall \hat{\sigma}^k \in \sigma(A^k),$$
$$M^k(\sigma^k \sigma^{-k}) \leq M^k(\sigma^k \hat{\sigma}^{-k}) \quad \forall \hat{\sigma}^{-k} \in \sigma(A^{-k}).$$

Ovvero ogni agente riceve un'utilità maggiore quando almeno uno degli altri agenti cambia strategia.

Osservazione 2.10. *Tutti i punti di sella di uno stage game hanno stessa utilità.*

Adesso è possibile enunciare l'ultima delle ipotesi sfruttate nella dimostrazione della convergenza:

3. Vale una delle seguenti due condizioni durante l'apprendimento:

   - **Condizione A.** Ogni stage game $(Q_*^1(s), ..., Q_*^n(s))$ e $(Q_t^1(s), ..., Q_t^n(s))$, per ogni t ed s, ha un punto di ottimo globale e l'aggiornamento delle Q-functions degli agenti avviene tramite questo equilibrio.
   - **Condizione B.** Ogni stage game $(Q_*^1(s), ..., Q_*^n(s))$ e $(Q_t^1(s), ..., Q_t^n(s))$, per ogni t ed s, ha un punto di sella e l'aggiornamento delle Q-functions degli agenti avviene tramite questo equilibrio.

La dimostrazione si fonda sul seguente risultato di Littman e Szepesvári [7] che stabilisce la convergenza di procedimenti di aggiornamento generali nel Q-learning, attraverso pseudo-contrazioni:

Teorema 2.11. *Sia $\mathbb{Q}$ lo spazio delle Q-functions. Si supponga che α_t soddisfi l'ipotesi 2 e che la mappa $P_t : \mathbb{Q} \to \mathbb{Q}$ soddisfi la seguente condizione: esiste un numero $0 < \gamma < 1$ ed una successione $\lambda_t \geq 0$ convergente q.c. a 0 tale che $Q_* = \mathbb{E}[P_t Q_*]$ e*

$$||P_t Q - P_t Q_*|| \leq \gamma ||Q - Q_*|| + \lambda_t \quad \forall Q \in \mathbb{Q},$$

allora l'iterazione definita da:

$$Q_{t+1} = (1 - \alpha_t) Q_t + \alpha_t [P_t Q_t]$$

converge q.c. a Q_*.

Nel caso in esame l'operatore P_t è definito come segue:

Definizione 2.12. *Sia $Q = (Q^1, ..., Q^n)$ e $\mathbb{Q} = \mathbb{Q}^1 \times ... \times \mathbb{Q}^n$. La mappa $P_t : \mathbb{Q} \to \mathbb{Q}$ tra spazi metrici completi è definita come $P_t Q := (P_t Q^1, ..., P_t Q^n)$, dove:*

$$P_t Q^k(s, a^1, ..., a^n) = r_t^k(s, a^1, ..., a^n) + \beta \; Q^k(s', \pi^1(s'), ..., \pi^n(s')),$$

dove s' è lo stato al tempo $t+1$ e $(\pi^1(s'), ..., \pi^n(s'))$ è un equilibrio di Nash per lo stage game $(Q^1(s'), ..., Q^n(s'))$.

Nel seguito si mostrerà che l'operatore P_t soddisfa le ipotesi del teorema 2.11. Occorre premettere un ulteriore risultato (Filar e Vrieze [2]) che collega gli equilibri di Nash per l'intero gioco stocastico con gli equilibri per gli stage games:

Lemma 2.13. *Le seguenti affermazioni sono equivalenti:*

1. $(\pi_*^1, ..., \pi_*^n)$ *è un equilibrio di Nash in un gioco stocastico scontato con utilità all'equilibrio* $(v^1(\pi_*^1, ..., \pi_*^n), ..., v^n(\pi_*^1, ..., \pi_*^n))$

2. *per ogni* $s \in S$, $(\pi_*^1(s), ..., \pi_*^n(s))$ *è un equilibrio di Nash per lo stage game* $(Q_*^1(s), ..., Q_*^n(s))$ *con utilità all'equilibrio* $(v^1(s, \pi_*^1, ..., \pi_*^n), ..., v^n(s, \pi_*^1, ..., \pi_*^n))$, *dove:*

$$Q_*^k(s, a^1, ..., a^n) = r^k(s, a^1, ..., a^n) + \beta \sum_{s'} p(s'|s, a^1, ..., a^n) v^k(s', \pi_*^1, ..., \pi_*^n) \tag{8}$$

In particolare segue che $v_*^k(s) = Q_*^k(s, \pi_*^1(s), ..., \pi_*^n(s))$. Ciò porta al seguente lemma:

Lemma 2.14. *Per un gioco stocastico a* $n-$*giocatori,* $\mathbb{E}[P_t Q_*] = Q_*$, *dove* $Q_* = (Q_*^1, ..., Q_*^n)$.

Dimostrazione. Dall'equazione (8) e dall'osservazione precedente segue che:

$$\begin{aligned} Q_*^k(s, a^1, ..., a^n) &= r^k(s, a^1, ..., a^n) + \beta \sum_{s'} p(s'|s, a^1, ..., a^n) Q_*^k(s', \pi_*^1(s'), ..., \pi_*^n(s')) \\ &= \sum_{s'} p(s'|s, a^1, ..., a^n)(r^k(s, a^1, ..., a^n) + \beta Q_*^k(s', \pi_*^1(s'), ..., \pi_*^n(s'))) \\ &= \mathbb{E}[P_t^k Q_*^k(s, a^1, ..., a^n)], \end{aligned}$$

dove l'ultima uguaglianza segue dalle osservazioni 2.8 e 2.10. □

Per poter applicare il lemma 2.11 resta da mostrare che P_t è una pseudo-contrazione. Sotto l'ipotesi 3 in realtà si tratta di una vera contrazione.

Lemma 2.15. P_t *è una contrazione rispetto alla norma uniforme, ovvero:*

$$||P_t Q - P_t \hat{Q}|| \leq \beta ||Q - \hat{Q}|| \quad \forall Q, \hat{Q} \in \mathbb{Q}.$$

Dimostrazione.

$$\begin{aligned} ||P_t Q - P_t \hat{Q}|| &= \max_j ||P_t Q^j - P_t \hat{Q}^j|| \\ &= \max_j \max_s \beta \ |Q^j(s, \pi^1(s), ..., \pi^n(s)) - \hat{Q}^j(s, \hat{\pi}^1(s), ..., \hat{\pi}^n(s)|. \end{aligned}$$

Dunque è sufficiente provare che:

$$|Q^j(s, \pi^1(s), ..., \pi^n(s)) - \hat{Q}^j(s, \hat{\pi}^1(s), ..., \hat{\pi}^n(s))| \leq ||Q^j(s) - \hat{Q}^j(s)||.$$

Per semplicità di notazione si pone $\sigma^j = \pi^j(s)$. La disuguaglianza da provare diventa pertanto:

$$|Q^j(s, \sigma^j, \sigma^{-j}) - \hat{Q}^j(s, \hat{\sigma}^j, \hat{\sigma}^{-j})| \leq ||Q^j(s) - \hat{Q}^j(s)||.$$

- **Caso A**: se entrambi $(\sigma^1, ..., \sigma^n)$ e $(\hat{\sigma}^1, ..., \hat{\sigma}^n)$ sono punti ottimali globali allora, supponendo senza perdita di generalità che $Q^j(s, \sigma^j, \sigma^{-j}) \geq \hat{Q}^j(s, \hat{\sigma}^j, \hat{\sigma}^{-j})$, si ottiene:

$$\begin{aligned} Q^j(s, \sigma^j, \sigma^{-j}) - \hat{Q}^j(s, \hat{\sigma}^j, \hat{\sigma}^{-j}) &\leq Q^j(s, \sigma^j, \sigma^{-j}) - \hat{Q}^j(s, \sigma^j, \sigma^{-j}) \\ &= \sum_{a^1,...,a^n} \sigma^1(a^1)...\sigma^n(a^n)(Q^j(s, a^1, ..., a^n) - \hat{Q}^j(s, a^1, ..., a^n)) \\ &\leq \sum_{a^1,...,a^n} \sigma^1(a^1)...\sigma^n(a^n)||Q^j(s) - \hat{Q}^j(s)|| \\ &= ||Q^j(s) - \hat{Q}^j(s)||. \end{aligned}$$

- **Caso B**: se entrambi $(\sigma^1, ..., \sigma^n)$ e $(\hat{\sigma}^1, ..., \hat{\sigma}^n)$ sono punti di sella allora, supponendo senza perdita di generalità che $Q^j(s, \sigma^j, \sigma^{-j}) \geq \hat{Q}^j(s, \hat{\sigma}^j, \hat{\sigma}^{-j})$, si ottiene:

$$\begin{aligned} Q^j(s, \sigma^j, \sigma^{-j}) - \hat{Q}^j(s, \hat{\sigma}^j, \hat{\sigma}^{-j}) &\leq Q^j(s, \sigma^j, \sigma^{-j}) - \hat{Q}^j(s, \sigma^j, \hat{\sigma}^{-j}) \\ &\leq Q^j(s, \sigma^j, \hat{\sigma}^{-j}) - \hat{Q}^j(s, \sigma^j, \hat{\sigma}^{-j}) \\ &\leq ||Q^j(s) - \hat{Q}^j(s)||. \end{aligned}$$

Ciò conclude la dimostrazione. □

Per concludere si enuncia adesso il teorema nella sua versione finale, immediata conseguenza dei lemmi appena dimostrati.

Teorema 2.16. *Sotto le ipotesi 1-3, la successione $Q_t = (Q_t^1, ..., Q_t^n)$ aggiornata tramite:*

$$Q_{t+1}^k(s, a^1, ..., a^n) = (1 - \alpha_t)Q_t^k(s, a^1, ..., a^n) + \alpha_t(r_t^k + \beta Q_t^k(s', \pi^1(s'), ..., \pi^n(s'))$$

dove $(\pi^1(s'), ..., \pi^n(s'))$ è un equilibrio di Nash del tipo richiesto per lo stage game $(Q_t^1(s'), ..., Q_t^n(s'))$, converge alla Nash Q-value $Q_* = (Q_*^1, ..., Q_*^n)$.

In particolare il lemma 2.13 assicura che un equilibrio di Nash per il limite delle Q-functions corrisponde ad un equilibrio di Nash per l'intero gioco stocastico.

3 Friend or Foe Q-learning

Le ipotesi per la convergenza del NashQ-learning sono estremamente restrittive e poco realistiche. Un passo avanti è stato proposto da M. Littman [6] attraverso un nuovo algoritmo (FFQ) che si dimostra convergere sempre e, nel caso di giochi con equilibri di tipo *coordination* e *adversarial*, il limite coincide con la *NashQ-value function*. Per far ciò è necessario dichiarare a priori quali degli altri agenti sono amici (*Friend*) o nemici (*Foe*). Si richiamano qui due definizioni:

Definizione 3.1. *Un adversarial equilibrium (punto di sella) è un equilibrio di Nash π tale che nessun agente è danneggiato da un cambiamento degli altri agenti, ovvero:*

$$R_i(\pi_1, ..., \pi_n) \leq R_i(\pi'_1, ..., \pi'_{i-1}, \pi_i, \pi'_{i+1}, ..., \pi'_n).$$

Definizione 3.2. *Un coordination equilibrium (global optimum) è un equilibrio di Nash π tale che tutti gli agenti ottengono la loro massima utilità possibile, ovvero:*

$$R_i(\pi_1, ..., \pi_n) = \max_{a_1 \in A_1, ..., a_n \in A_n} R_i(a_1, ..., a_n)$$

Osservazione 3.3. *In un gioco di tipo zero-sum a due giocatori ($R_1 = -R_2$) tutti gli equilibri sono di tipo adversarial.*

In un gioco fully cooperative ($R_1 = R_2 = ... = R_n$) c'è almeno un equilibrio di tipo coordination.

Conseguenze immediate delle definizioni sono le seguenti importanti proprietà:

Lemma 3.4. *Tutti gli equilibri di tipo adversarial hanno stessa payoff.*

Tutti gli equilibri di tipo coordination hanno stessa payoff.

3.1 Algoritmo

L'idea fondamentale di Littman consiste nel sostituire l'operatore $NashQ^i$ dell'equazione (7) per l'aggiornamento nel NashQ-learning:

$$Q^i_{t+1}(s, a^1, ..., a^n) = (1 - \alpha_t) Q^i_t(s, a^1, ..., a^n) + \alpha_t [r^i_t + \beta \; NashQ^i_t(s')],$$

considerando le informazioni aggiuntive sugli altri agenti. Ovvero in un gioco con n giocatori siano $X_1, ..., X_k$ le azioni possibili dei k amici del giocatore i e $Y_1, ..., Y_l$ le azioni dei suoi l nemici. Allora si pone:

$$NashQ^i(s, Q_1, ..., Q_n) = \max_{\pi \in \Pi(X_1 \times ... \times X_k)} \min_{y_1, ..., y_l \in Y_1 \times ... \times Y_l} \sum_{x_1, ..., x_k \in X_1 \times ... \times X_k} \pi(x_1)...\pi(x_k) Q_i(s, x_1, ..., x_k, y_1, ..., y_l).$$

Ovvero una generalizzazione dei due casi estremi, in cui ci sono soltanto due agenti amici:

$$NashQ^1(s, Q_1, Q_2) = \max_{a_1 \in A_1, a_2 \in A_2} Q_1(s, a_1, a_2)$$

o nemici:

$$NashQ^1(s, Q_1, Q_2) = \max_{\pi \in \Pi(A_1)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi(a_1) Q_1(s, a_1, a_2)$$

3.2 Convergenza

Ancora una volta è sufficiente applicare il lemma 2.11 per concludere la convergenza dell'algoritmo, analogamente a quanto visto per il Nash Q-learning. A differenza di quest'ultimo, nel caso del FFQ-learning, la convergenza è sempre assicurata sotto le ipotesi 1-2, anche se non è detto che i valori appresi dall'algoritmo corrispondano a quelli di un equilibrio di Nash. Vale comunque il seguente teorema:

Teorema 3.5. *L'algoritmo Foe-Q (tutti gli altri agenti sono nemici) converge ai valori di un equilibrio di Nash se il gioco ha un equilibrio di tipo adversarial.*

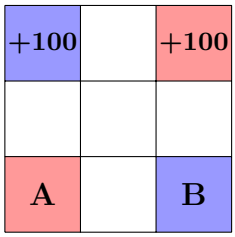
L'algoritmo Friend-Q (tutti gli altri agenti sono amici) converge ai valori di un equilibrio di Nash se il gioco ha un equilibrio di tipo coordination.

Dimostrazione. Per il lemma 2.13 è sufficiente osservare che il valore di un equilibrio di tipo coordination in uno stage game è la massima payoff e che nel caso di un equilibrio di tipo adversarial è il minimax. □

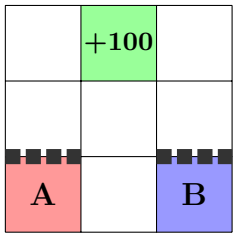
4 Esempi e conclusioni

4.1 Grid-world games

In questa sezione si discutono brevemente i comportamenti degli algoritmi FFQ e Nash-Q in due semplici giochi:



(a) Grid-world Game 1

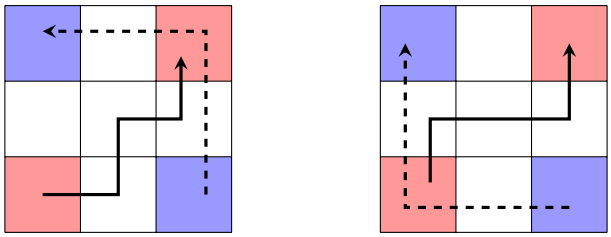


(b) Grid-world Game 2

Figura 2: Nel gioco 1 i due agenti A e B devono raggiungere le caselle opposte della griglia muovendosi contemporaneamente ed un passo alla volta nelle direzioni N,E,W,S (quando possibile). Il primo a raggiungere l'obiettivo ha una ricompensa pari a +100. Se i giocatori raggiungono il proprio obiettivo contemporaneamente ricevono entrambi la ricompensa. Nel caso in cui muovendosi finiscono nella stessa casella, allora rimbalzano indietro e subiscono un danno pari a −1. Nel gioco 2 le regole sono le stesse, tuttavia la casella obiettivo è comune ad entrambi i giocatori. Inoltre se un giocatore decide di attraversare la barriera (linee tratteggiate) allora con probabilità 0.5 riuscirà nel suo intento, altrimenti resterà fermo.

Nel gioco 1 entrambi i giocatori possono ottenere il massimo punteggio, attraverso numerosi equilibri di tipo *coordination* come mostrato in figura 3.

Dall'implementazione si può verificare che l'algoritmo Nash-Q riesce a convergere ai corretti valori, nonostante l'ipotesi 3 del teorema 2.16 sia violata più volte durante l'apprendimento (ciò in particolare mostra la non ottimalità delle assunzioni prese in considerazione da Hu e Wellman). L'algoritmo Friend-Q converge agli stessi valori, come ci si aspetterebbe dalla struttura del gioco. Si osservi però che la presenza di diversi equilibri di tipo coordination con stessa payoff complica la scelta della policy, ad esempio A potrebbe giocare secondo l'equilibrio a), B secondo l'equilibrio b) con risultato pessimo per entrambi.

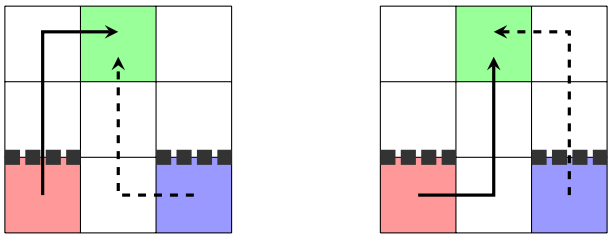


(a) (b)

Figura 3: Esempi di equilibri di tipo coordination per il Gioco 1.

L'algoritmo Foe-Q assegna valore 0 allo stato iniziale dato che l'altro giocatore può sempre impedire all'agente di raggiungere l'obiettivo. D'altra parte la strategia "resta fermo" vale anche nel peggiore degli eventi un punteggio pari a 0, superiore al caso negativo della scelta di equilibri non compatibili per il Friend-Q learning discusso prima.

Il gioco 2 invece non presenta equilibri nè di tipo coordination nè di tipo adversarial. I due equilibri di Nash del gioco sono riportati in figura 4. Hu e Wellman hanno mostrato nel loro articolo che la convergenza dell'algoritmo Nash-Q dipende sostanzialmente da come la funzione $NashQ$ è implementata.



(a) (b)

Figura 4: I due equilibri di Nash del Gioco 2.

L'algoritmo Friend-Q impara ad utilizzare il passaggio dal centro. La ragionevolezza di tale strategia dipende dal comportamento dell'avversario: se en-

trambi ad esempio sono Friend-Q learners allora entrambi sceglieranno il centro, con conseguente danno comune.

Infine l'algoritmo Foe-Q sceglie di evitare la possibilità di conflitto e provare ad attraversare la barriera. Ciò si traduce metà delle volte nel raggiungimento dell'obiettivo, indipendentemente dalla strategia dell'avversario.

4.2 Conclusioni

Come già discusso nelle sezioni precedenti l'introduzione di altri agenti razionali all'interno dell'ambiente complica notevolmente la trattazione del reinforcement learning.

Sebbene il *NashQ-learning* nasca come naturale generalizzazione del *Q-learning* nel framework dei giochi stocastici, da un punto di vista matematico diventa complesso determinare condizioni non eccessivamente restrittive sotto cui gli algoritmi proposti convergano.

Inoltre da un punto di vista computazionale il calcolo delle Q-functions diventa impraticabile al crescere del numero di stati. Per ovviare a questo problema negli ultimi anni si sono sviluppati metodi come il *DeepQ-learning* che sfrutta reti neurali profonde per approssimare tali funzioni.

Infine anche la nozione di "ottimalità" non è immediata. Come illustrato negli esempi precedenti gli algoritmi scelgono strategie differenti, la cui ragionevolezza dipende sostanzialmente dagli altri agenti. Recentemente sono state pertanto approfondite varianti del *NashQ-learning*, come il *Correlated Q-learning*, che prende in considerazione differenti nozioni di equilibrio per generalizzare il *Friend or Foe Q-learning*.

Sono dunque molte e diverse le direzioni in cui è possibile estendere gli algoritmi presentati.

Riferimenti bibliografici

[1] Michael Bowling. Convergence problems of general-sum multiagent reinforcement learning. pages 89–94, 2000.

[2] Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer Science & Business Media, 2012.

[3] Arlington M Fink. Equilibrium in a stochastic n-person game. *Journal of science of the hiroshima university, series ai (mathematics)*, 28(1):89–93, 1964.

[4] Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069, 2003.

[5] Junling Hu, Michael P Wellman, et al. Multiagent reinforcement learning: theoretical framework and an algorithm. 98:242–250, 1998.

[6] Michael L Littman et al. Friend-or-foe q-learning in general-sum games. 1:322–328, 2001.

[7] Michael L Littman and Csaba Szepesvári. A generalized reinforcement-learning model: Convergence and applications. 96:310–318, 1996.

[8] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[9] Csaba Szepesvári. *Algorithms for reinforcement learning*, volume 4. Morgan & Claypool Publishers, 2010.

[10] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.