Elaborato esame di Human Computer Interaction: Deep Reinforcement Learning in a Boat Game

Mandelli Lorenzo
E-mail address
lorenzo.mandelli@stud.unifi.it

June 13, 2021

1 Introduzione

L'intento di questo elaborato è quello di verificare se opportune strategie di *Deep Reinforcement Learning* (*DRL*) possano mostrasi efficaci per la realizzazione di personaggi non-giocanti (*NPCs*) in un videogioco tali da avere un comportamento credibile e in grado di fornire una sfida di gioco piacevole e stimolante per l'utente. A questo scopo è stato realizzato un prototipo roguelike a turni in cui gli agenti avversari sono realizzati attraverso metodi di *DRL* attraverso il framework realizzato da Alessandro Sestini, Alexander Kuhnle e Andrew D. Bagdanov e descritto nell'articolo [1]. Gli agenti ottenuti sono stati testati quantitativamente e qualitativamente per poter determinare se il loro comportamento finale risultasse convincente secondo gli obbiettivi prefissati. Infine è stato data una valutazione al framework utilizzato per la realizzazione degli agenti.

2 Prototipo

Al fine di poter sperimentare specifiche tecniche di *DRL* è stato realizzato un semplice prototipo di un videogioco roguelike in cui due personaggi, rappresentanti due barche, si sfidano per cercare di affondarsi a vicenda.
Il videogioco, la cui finestra è mostrata in figura 1, si basa su un combattimento a turni, in cui a ogni turno un personaggio può decidere se muoversi verso una casella della mappa disponibile o sparare per cercare di danneggiare l'avversario. Di seguito sono riportati le caratteristiche dei vari elementi presenti nel prototipo.

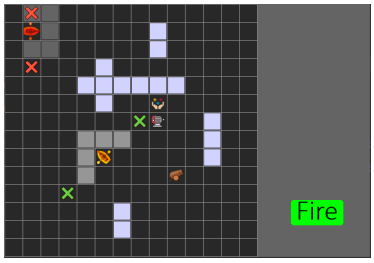

Figura 1: Prototipo di videogioco roguelike a turni realizzato. La barca gialla rappresenta il giocatore, la barca rossa l'agente addestrato.

2.1 Personaggi

I personaggi durante la durata della partita possiedono le seguenti caratteristiche:

- **Hp**: il numero di punti ferita di cui dispongono. Ogni volta che un personaggio è colpito dall'avversario il suo numero di punti ferita è ridotto di uno. Se questo valore diventa pari a zero il personaggio risulta essere sconfitto.

- **Range**: la distanza, ovvero il numero di caselle, a cui è possibile sparare. Di default i personaggi possiedono un valore di Range pari a 2, ma è pos-

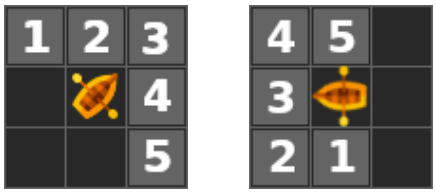

Figura 2: Schema delle azioni di movimento attuabili dai personaggi del videogioco.

sibile incrementarlo attraverso specifici potenziamenti.

- **Direction**: la direzione del personaggio nella mappa corrispondente all'ultima direzione verso cui esso si è spostato. Essendo il movimento e la traiettoria di fuoco dei personaggi non omnidirezionali, la direzione che essi assumono rispetto agli assi della mappa risulta essere una caratteristica fondamentale nella dinamica della partita.

Per quanto riguarda le azioni eseguibili dai personaggi, essi ne possono compiere un totale di 6 diverse:

- **Azione numero 0**: corrisponde all'azione di sparare. La dinamica dello sparo consiste nel verificare se l'avversario risulti essere in una delle caselle distanti al più il valore di *Range* e perpendicolari alla direzione *Direction* assunta dal personaggio che sta sparando. Meno formalmente i personaggi possono sparare lungo le direzioni individuate dai lati della barca e se l'avversario risulta essere lungo tale traiettoria e a distanza minore del valore di *Range* esso viene danneggiato.

- **Azioni numero 1-5**: corrispondono alle azioni possibili relative al movimento. L'azione 1 corrisponde al muoversi nella casella a 90 gradi verso propria sinistra, l'azione 2 verso la casella a 45 verso la propria sinistra, l'azione 3 ad andare diritti, l'azione 4 ad andare verso la casella a 45 gradi verso destra e infine l'azione 5 ad andare verso la casella a 90 alla propria destra. Lo schema del movimento è riportato in figura 2

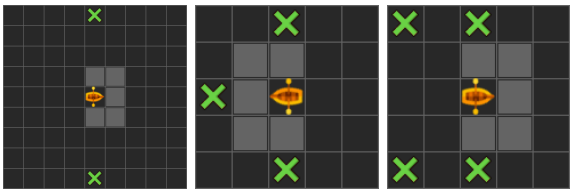

Figura 3: Riassunto grafico dei potenziamenti: **(a)** aumento del raggio di attacco, **(b)** aggiunta dell'attacco frontale, **(c)** aggiunta dell'attacco posteriore.

2.2 Mappa

La mappa di gioco è costituita da una griglia di dimensioni 14x14 in cui al suo interno sono presenti un numero casuale di ostacoli non attraversabili e vari tipologie di potenziamenti. I potenziamenti sono particolari elementi del gioco che permettono al personaggio che gli acquisisce di aumentare le sue caratteristiche in modo che risulti più facile sconfiggere il nemico.
I potenziamenti acquisibili sono di tre tipi:

- **Aumento del Range**: il valore del range aumenta da 2 a 4.

- **Aggiunta attacco frontale**: permette di poter sparare anche in avanti e non solo lungo i lati

- **Aggiunta attacco posteriore**: aggiunge due direzioni di tiro corrispondenti alle diagonali posteriori rispetto alla direzione della barca.

I diversi effetti dei potenziamenti sono riassunti graficamente in figura 3.

3 Addestramento degli agenti

La realizzazione degli agenti per mezzo di tecniche di *DRL* è stata realizzata attraverso il framework realizzato da Alessandro Sestini, Alexander Kuhnle e Andrew D. Bagdanov e proposto nell'articolo [1].

3.1 Deep Reinforcement Learning

L'apprendimento per rinforzo (*RL*) è una tecnica di apprendimento automatico che punta alla realizzazione di agenti autonomi in grado di scegliere azioni da compiere per il conseguimento di determinati obiettivi. Il loro addestramento è realizzato secondo una politica basata su tentativi e errori (*trials and errors*) tramite interazione con l'ambiente in cui sono immersi.
L'apprendimento per rinforzo profondo (*DRL*) combina l'apprendimento per rinforzo e l'apprendimento profondo consentendo agli agenti di prendere decisioni da dati di input non strutturati senza la gestione manuale dello spazio degli stati e permettendo di accettare input molto grandi (ad esempio ogni pixel visualizzato sullo schermo in un videogioco) per decidere quali azioni eseguire per ottimizzare un obiettivo (ad esempio massimizzare il punteggio del gioco).

3.2 Struttura della rete

La struttura della rete utilizzata, riportata in figura 4, è composta da 4 rami che prendono rispettivamente in ingresso:

- Il contenuto di tutta la mappa di gioco di dimensioni 14×14 codificato come interi:
  - 0: una cella vuota.
  - 1: un ostacolo della mappa.
  - 2: il giocatore.
  - 3: l'agente.
  - 4: il limite della traiettoria di attacco raggiungibile del giocatore.
  - 5 il limite della traiettoria di attacco raggiungibile dall'agente.
  - 6, 7, 8 rispettivamente la posizione dei tre tipi di potenziamenti possibili presenti nella mappa.
- Una mappa locale di dimensioni 3×3 centrata nella posizione dell'agente.
- Una mappa locale di dimensioni 5×5 centrata nella posizione dell'agente.
- Un vettore di 8 elementi che codifica le caratteristiche dell'agente e del giocatore riguardo al numero di Hp, al valore di Range, alla direzione rispetto alla mappa e a un flag che indica se l'avversario è a tiro.

I 3 rami che prendono in ingresso le mappe globali e locali sono seguiti da un livello di embedding che trasformano le matrici di input in delle rappresentazioni continue. Sono poi seguite da uno strato convoluzionale con 32 filtri di dimensioni 3×3 e da un altro strato convoluzionale 3×3 con 64 filtri.
Il ramo che prende in ingresso il vettore delle caratteristiche dell'agente è invece seguito da uno strato di embedding di dimensione 64 e poi da uno strato completamente connesso (FC) di dimensione 256.
Le uscite di tutti i rami sono poi concatenate per formare un unico vettore che viene fatto passare attraverso uno strato FC di dimensione 256. Una rappresentazione one-hot dell'azione intrapresa nel passaggio precedente è aggiunta al vettore che viene passato infine attraverso un livello LSTM.

La scelta dell'utilizzo combinato delle rappresentazioni locali e globali della mappa ha lo scopo di migliorare il punteggio raggiunto dall'agente e la qualità complessiva del suo comportamento, il loro utilizzo infatti aiuta a valutare sia la situazione generale dell'ambiente che le particolari configurazioni locali vicine all'agente.

3.3 Funzione di Reward

Analogamente a quanto riportato nell'articolo [1] è stata utilizzata una funzione di Reward estremamente sparsa del tipo:

$$R(t) = -0.01 + \begin{cases} -0.1 & \text{mossa non consentita} \\ +10.0 * \text{Hp} & \text{in caso di vittoria} \end{cases}$$

3.4 Curriculum

Essendo la mappa del gioco di grandi dimensioni (14x14), anche maggiori a quelle utilizzate

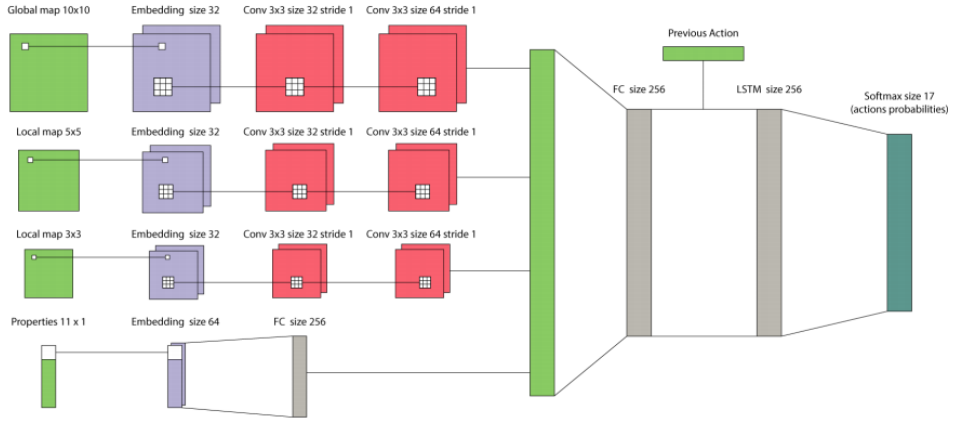

Figura 4: L'architettura di rete realizzata nell'articolo [1] per l'addestramento degli NPC.

Fase	step iniziale	step finale	Dim mappa
Fase 1	0	$2e6$	9x9
Fase 2	$2e6$	∞	14x14

Tabella 1: Fasi del curriculum.

nell'articolo di riferimento [1] (10x10), è stato necessario per l'addestramento ricorrere a una procedura di *curriculum learning* in cui le dimensioni della mappa sono aumentate durante l'addestramento.
Il curriculum realizzato imposta una soglia a *2e6* step eseguiti dall'agente come riportato nella tabella 1. All'inizio dell'addestramento la mappa percorribile dai personaggi è di dimensioni 9×9, dopo $2e6$ passi le dimensioni diventano pari a 14×14.
Durante tutta la durata dell'addestramento gli agenti si sono confrontati contro avversari che impiegano mosse casuali, in modo ottenere una maggiore esplorazione degli stati di gioco possibili.

4 Risultati

4.1 Risultati quantitativi

I risultati quantitativi rappresentanti l'andamento del valore di reward medio e dell'entropia in funzione del numero di episodi di addestramento sono mostrati in figura 5.
L'andamento crescente della curva di reward mostra che l'agente riesce a comprendere sempre meglio le dinamiche del gioco, fino al raggiungimento della saturazione a un valore massimale, che corrisponde al massimo grado di apprendimento dell'ambiente raggiunto. Il picco verticale della curva corrisponde invece alle fase di variazione dell'ambiente dovuta all'utilizzo del *curriculum*: in questo passaggio infatti l'agente si trova in un contesto nuovo e più complicato di quello visto in precedenza e ha quindi una temporanea ricaduta delle sue prestazioni seguita da un rapido miglioramento.

4.2 Risultati qualitativi

Al fine di dare una valutazione anche qualitativa delle capacità degli agenti è stato formulato un test di giocabilità che consiste in un insieme di 9 domande da sottoporre a un gruppo di 10 candidati. I candidati hanno provato il prototipo per 20 minuti e successivamente hanno risposto alle domande fornite dal test di giocabilità fornendo valori compresi tra 1 e 7, dove 1 corrisponde a una risposta fortemente negativa e 7 fortemente positiva. Lo schema delle domande poste e dei valori ottenuti sono mostrati in tabella 2.

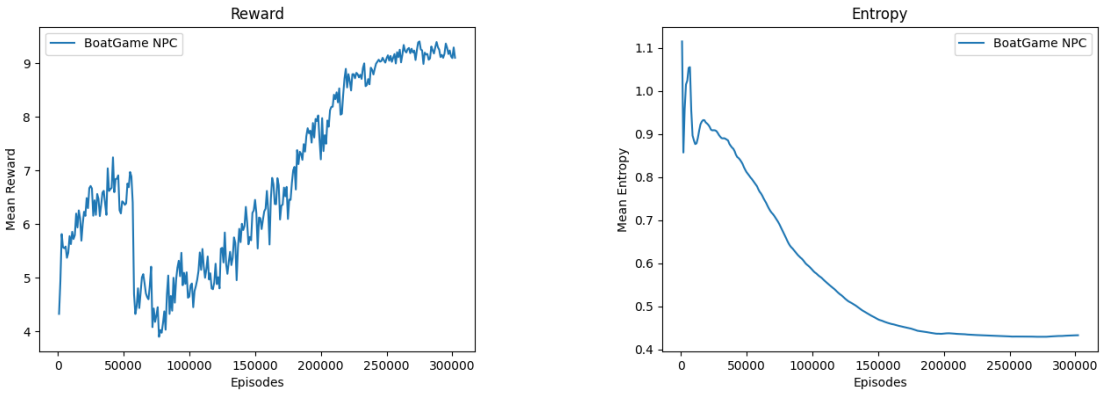

Figura 5: Grafici dei valori di reward medio e di entropia al variare del numero di epoche ottenuti durante l'addestramento degli agenti.

Le domande sono state formulate in base ai principi di *Credibilità, Imperfezione* e *Prior-Free* proposti nell'articolo [1].
Le domande 0,1,2,3,7 e 8 corrispondono al principio di *Credibilità*, ovvero cercano di stabilire quanto l'NPC sia credibile e quanto possa essere ritenuto intelligente. In particolare le domande 7-8 si concentrano sulla capacità dell'agente di rispondere adeguatamente a diverse configurazioni dell'ambiente e dell'avversario dovute alla presenza casuale dei potenziamenti nella mappa. Un agente per essere ritenuto pienamente credibile dovrebbe essere in grado di recarsi verso i potenziamenti, comprendendone l'utilità, e di variare il proprio comportamento in base ai potenziamenti acquisiti dal proprio avversario (ad esempio evitare di stare davanti al nemico se questo ha acquisito il potenziamento che aggiunge il cannone frontale).
Le domande 4-5-6 indagano invece sul principio di *Imperfezione* secondo il quale gli agenti realizzati non devono essere imbattibili in quanto ciò comporterebbe per l'utente un'esperienza di gioco spiacevole.
Infine le domande 1-2 corrispondono al il principio di *Prior-Free* ovvero quanto gli *NPC* sono stati in grado di estrapolare strategie in modo indipendente attraverso il meccanismo di tentativi ed errori del *DRL* e che quindi una codifica manuale da parte degli sviluppatori di comportamenti complessi non sia richiesta per la riuscita di questo task.
Come si evince dai valori ottenuti in tabella 2 i candidati hanno trovati gli agenti in grado di discernere correttamente le dinamiche di gioco, ma dotati però di un comportamento non sempre convincente. L'agente infatti si mostra competente a muoversi nella mappa e a sparare al nemico non appena esso si trovi sulla propria traiettoria di fuoco, ma è stato anche possibile notare come l'agente non riesca a sconfiggere un avversario che resta fermo sul posto e non si mostra interessato all'acquisizione di alcun potenziamento, nonostante ne abbia compreso i relativi effetti (domande 7 e 8).
L'agente infatti tende a acquisire potenziamenti solo se si trovano nel suo percorso di movimento verso il nemico e tende a non deviare mai per cercare di acquisirne uno o per ostacolare il nemico che sta andando in direzione di essi.
Allo stesso modo se il giocatore resta fermo (attraverso l'azione di sparo che non comporta alcun movimento) per un prolungato numero di turni l'agente tende a disporsi in prossimità di esso, senza essere però in grado di colpirlo, ed aspettare anch'esso sul posto o a ripetere una successione di spostamenti periodica. Questo comportamento è interpretabile nel fatto che l'agente cerca di porsi in modo da anticipare le prossime mosse del nemico e non di cercare

N°	Domande	μ	σ
0	Ritieni che i nemici fossero intelligenti?	3.90	1.30
1	Ritieni che i nemici seguissero una strategia?	3.00	1.00
2	Ritieni che i nemici abbiano commesso mosse controintuitive?	4.80	1.33
3	Quanto ritieni importante l'essere in grado di predire le mosse future?	5.90	1.37
4	Ti sei sentito in grado di sconfiggere il nemico?	6.00	1.00
5	Il comportamento dei nemici rappresentava una sfida?	2.90	1.30
6	Ritieni che i nemici abbiano commesso alcuni errori?	4.9	0.83
7	Ritieni che il comportamento dei nemici cambiasse in base alla presenza di potenziamenti nella mappa?	2.1	1.04
8	Ritieni che il comportamento dei nemici cambiasse in base ai potenziamenti che acquisivi?	5.1	1.04

Tabella 2: Test di giocabilità per la valutazione qualitativa degli agenti e del prototipo realizzato.

di entrare in traiettoria di fuoco per vincere subito la sfida. Questa dinamica, dovuta alle intrinseche meccaniche di gioco e di per se corretta, fa sembrare però poco credibile l'agente a un giocatore reale in quanto:

- Sembra che all'azione di restare fermi corrisponda sempre la medesima azione da parte del NPC una volta che si trova a una particolare distanza.
- Non spiega il motivo per cui l'agente non cerchi di porsi in una situazione più vantaggiosa in cui può minacciare sia la disposizione corrente del giocatore sia quella futura.

In generale NPC sembra che possieda un comportamento di tipo puramente reattivo e che quindi non sia in grado di adoperare strategie a lungo termine come il dirigersi verso potenziamenti o cercare di costringere in nemico a una serie di mosse, tenendolo sempre sotto scacco, per cercarlo di intrappolarlo e vincere la partita.

Questi comportamenti non desiderati sono attribuibili al fatto che l'addestramento è stato eseguito unicamente contro un avversario che esegue mosse casuali, che non prevede quindi lo stare fermi sul posto per un numero elevato di mosse consecutive (statisticamente è un evento molto raro) e che non comporti la necessità di accumulare potenziamenti per vincere la sfida (a causa dell'elevata semplicità).

Per ovviare a queste problematiche una soluzione sarebbe introdurre durante la fase di training delle fasi di *self-play* in cui l'agente affronta vecchie versioni di se stesso in modo che sia costretto a ricorrere sempre più a strategie offensive e competitive, come l'acquisizione di potenziamenti, e non solo passive e reattive.

Per quanto riguardo invece le dinamiche base del gioco, ad esempio la scelta di quando sparare e quando riposizionarsi, l'agente ha mostrato di averle comprese e di saper fornire all'utente una sfida moderata come testimoniato dai valori ottenuti dalla domanda 5. Il punteggio lievemente troppo basso rispetto a quanto desiderato è influenzato anche dal suo comportamento riguardo a strategie a lungo termine trattato esaustivamente in precedenza. Ad eccezione di queste, l'agente tende a commettere anche alcuni errori più legati alle regole di base del gioco come evidenziato dalla domanda 6, ad esempio posizionarsi sulla traiettoria nemica. Questi comportamenti tendono però ad accadere non troppo frequentemente e rendono l'agente fallibile e non perfetto, in accordo con il principio di *imperfezione*, evitando che il gioco entri in una situazione di stallo che comporti un'esperienza spiacevole per l'utente.

5 Valutazione del Framework

Il framework utilizzato [1] per introdurre meccaniche di DRL nel prototipo roguelike realizzato si è dimostrato molto facile da usare e adattabile alle specifiche dinamiche richieste dal gioco. Attraverso la semplice ereditarietà di poche classe e di uno script di inizializzazione permette infatti la completa realizzazione degli agenti, la definizione delle dimensioni degli input della rete e dei valori del *curriculum*.
Per quanto riguarda i risultati ottenuti, esso ha permesso di mostrare come nel caso preso in esame, ovvero un gioco con dinamiche di movimento non omnidirezionali e necessità di adoperare strategie complesse per evitare situazioni di stallo, il solo training contro un avversario dal comportamento casuale non permette di ottenere agenti pienamente convincenti e gradevoli per l'utente, ed è quindi necessario ampliare l'addestramento con meccaniche di *self-play*, che sono comunque realizzabili attraverso il framework, ma che non sono state adoperate in questo elaborato.
Una menzionabile criticità riscontrata consiste nel dover implementare due volte la struttura che modella la successione dei turni di gioco, dell'agente e del giocatore, in quanto questa tende a differire per l'esecuzione e per l'addestramento. Durante di fase di training infatti i turni si succedono uno dopo l'altro in modo automatico, mentre durante la reale esecuzione è necessario aspettare l'input dell'utente ed è quindi necessario un controllo a livello di eventi di gioco. Questa ridondanza di codice può portare al verificarsi di errori. Per rendere più consistente la struttura del gioco sarebbe possibile definire, tramite interfaccia, una struttura comune per entrambe le fasi, da poter poi alternare attraverso un parametro.

Bibliografia

[1] A. Sestini, A. Kuhnle, and A. D. Bagdanov. Deepcrawl: Deep reinforcement learning for turn based strategy games, 2020.