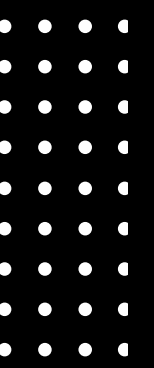
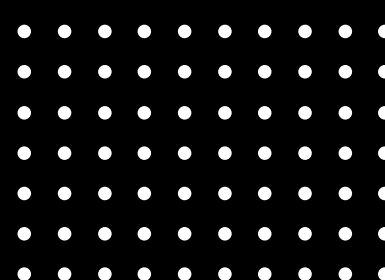


SALVATORE
SIRICA

PingPongAI

Reinforcement Learning per un
Ambiente Competitivo





01

Introduzione

Apprendimento per rinforzo

Attraverso l'implementazione di algoritmi di apprendimento per rinforzo, come Q-Learning e SARSA, questo progetto si propone di addestrare agenti intelligenti a competere in un ambiente simulato di gioco Pong.

L'obiettivo principale è confrontare le performance di questi algoritmi per valutarne i punti di forza e i limiti, con particolare attenzione alla loro applicabilità in un contesto competitivo e dinamico.

02

Ambiente e Ricompese

L'ambiente di gioco Pong è stato progettato per simulare partite tra due agenti. Le caratteristiche principali includono:

- Multiplayer: Due agenti possono competere simultaneamente.
- Stati dinamici: La velocità e la posizione della palla cambiano continuamente, richiedendo una rapida adattabilità.
- Personalizzazione: Costruito utilizzando OpenAI Gym per garantire modularità e compatibilità con diversi algoritmi RL.

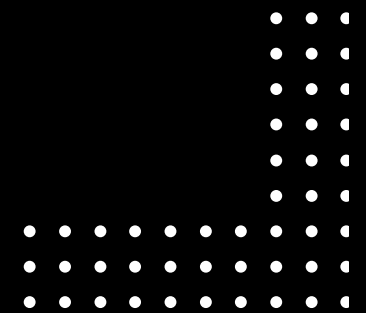
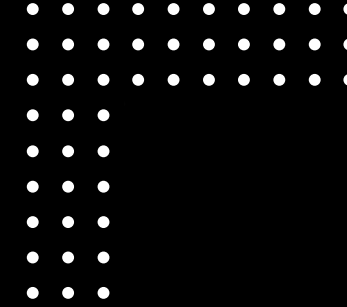
02

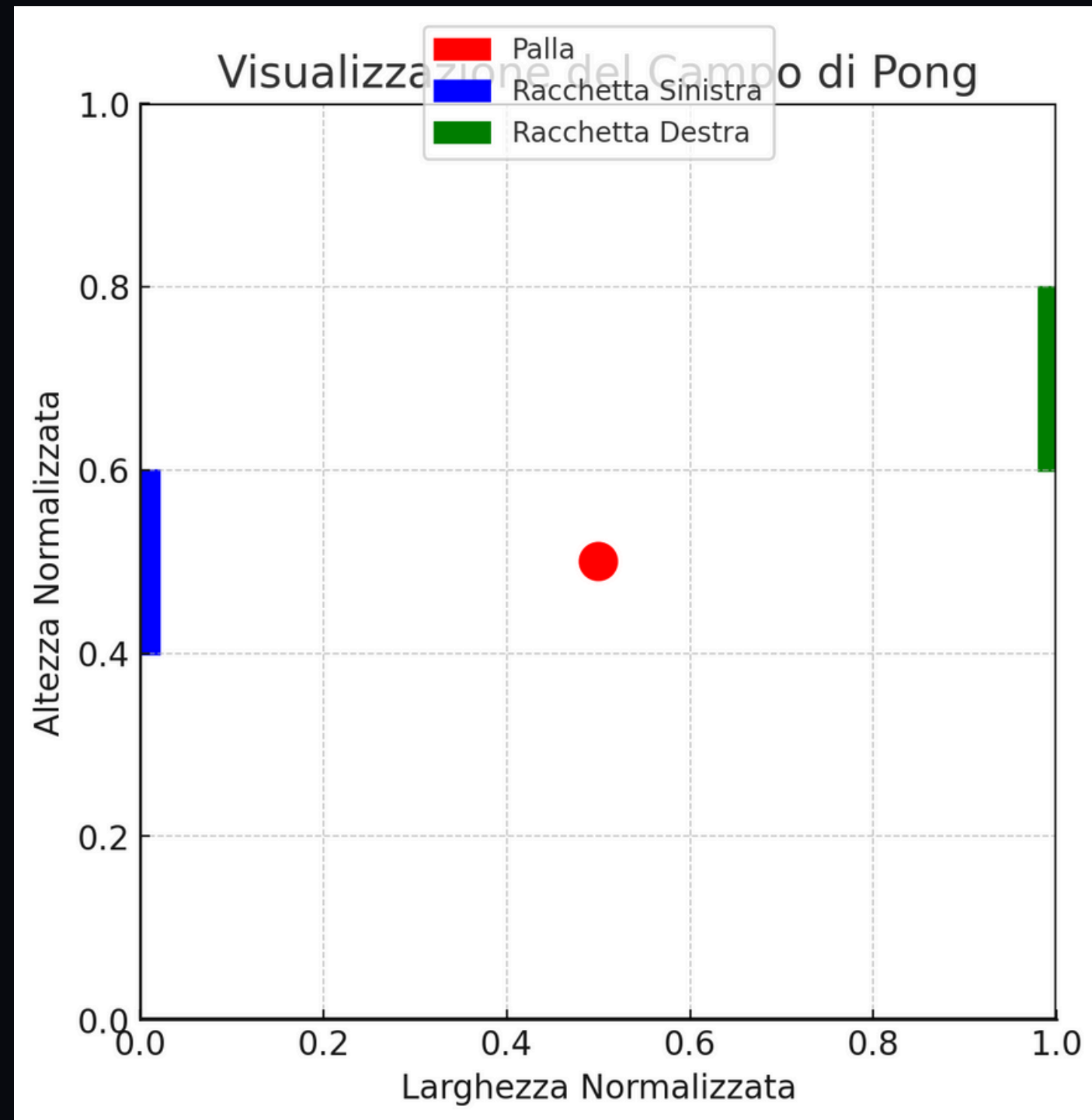
Ambiente e Ricompese

Le ricompense guidano l'apprendimento degli agenti:

- +1 per ogni collisione della palla con la racchetta dell'agente.
- -1 quando la palla supera la racchetta dell'agente.
- 0 durante il movimento standard senza eventi significativi.

Questo schema di ricompense spinge gli agenti a massimizzare il controllo sulla palla e a prevenire il punteggio avversario.





Stati

Gli stati dell'ambiente sono definiti da parametri chiave che descrivono il contesto del gioco, tra cui:

- La posizione della palla (x, y).
- La velocità della palla (vel_x, vel_y).
- La posizione delle racchette sinistra e destra (y_{left}, y_{right}).

Questi stati vengono discretizzati per consentire una gestione efficiente della Q-Table da parte degli algoritmi.

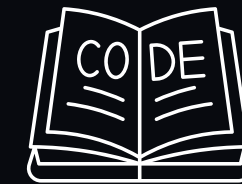
04

Azioni degli agenti

Nel contesto del gioco Pong, le azioni disponibili per ciascun agente si limitano a tre:

1. Spostarsi verso l'alto.
2. Spostarsi verso il basso.
3. Rimanere nella posizione attuale.

Queste azioni sono rappresentate come valori discreti e vengono selezionate dall'agente in base alla politica di apprendimento in uso (Q-Learning o SARSA).



`_ACTIONS = (_STAY, _MOVE_UP, _MOVE_DOWN)`

05

Base Agent

Il Base Agent rappresenta la struttura comune su cui si basano gli algoritmi Q-Learning e SARSA. Implementando alcune funzionalità principali

01.

POLITICA DI ESPLORAZIONE

Utilizzo di una strategia epsilon-greedy per bilanciare esplorazione ed esploitazione

02.

GESTIONE DELLA Q-TABLE

La Q-Table è una struttura di dati che associa ogni stato-azione a un valore

03.

ADATTABILITÀ

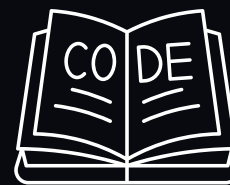
Parametri come il tasso di apprendimento *alpha* e il fattore di sconto *gamma* sono aggiornabili dinamicamente per ottimizzare l'addestramento

06 Q-Learning

Il Q-Learning è un algoritmo di apprendimento per rinforzo off-policy che aggiorna la Q-Table sulla base delle seguenti transizioni:

- Stato attuale (s).
- Azione scelta (a).
- Ricompensa ricevuta (r).
- Stato successivo (s').

L'aggiornamento avviene tramite l'equazione di Bellman



```
next_max = max([self.q_table[(next_state, a)] for a in self.actions])  
new_q = old_q + alpha * (reward + gamma * next_max - old_q)  
self.q_table[(state, action)] = new_q
```


07 SARSA

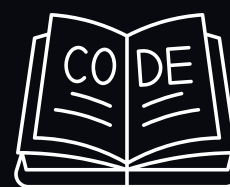
Il SARSA (State-Action-Reward-State-Action) è un algoritmo di apprendimento per rinforzo on-policy che aggiorna la Q-Table considerando l'azione effettivamente scelta nel nuovo stato.

Punti di Forza:

- Migliora la coerenza tra apprendimento e comportamento durante l'esplorazione.
- Ideale in ambienti stabili e prevedibili.

Limiti:

- Più lento rispetto a Q-Learning in ambienti dinamici.
- Esplorazione limitata dalla politica corrente.



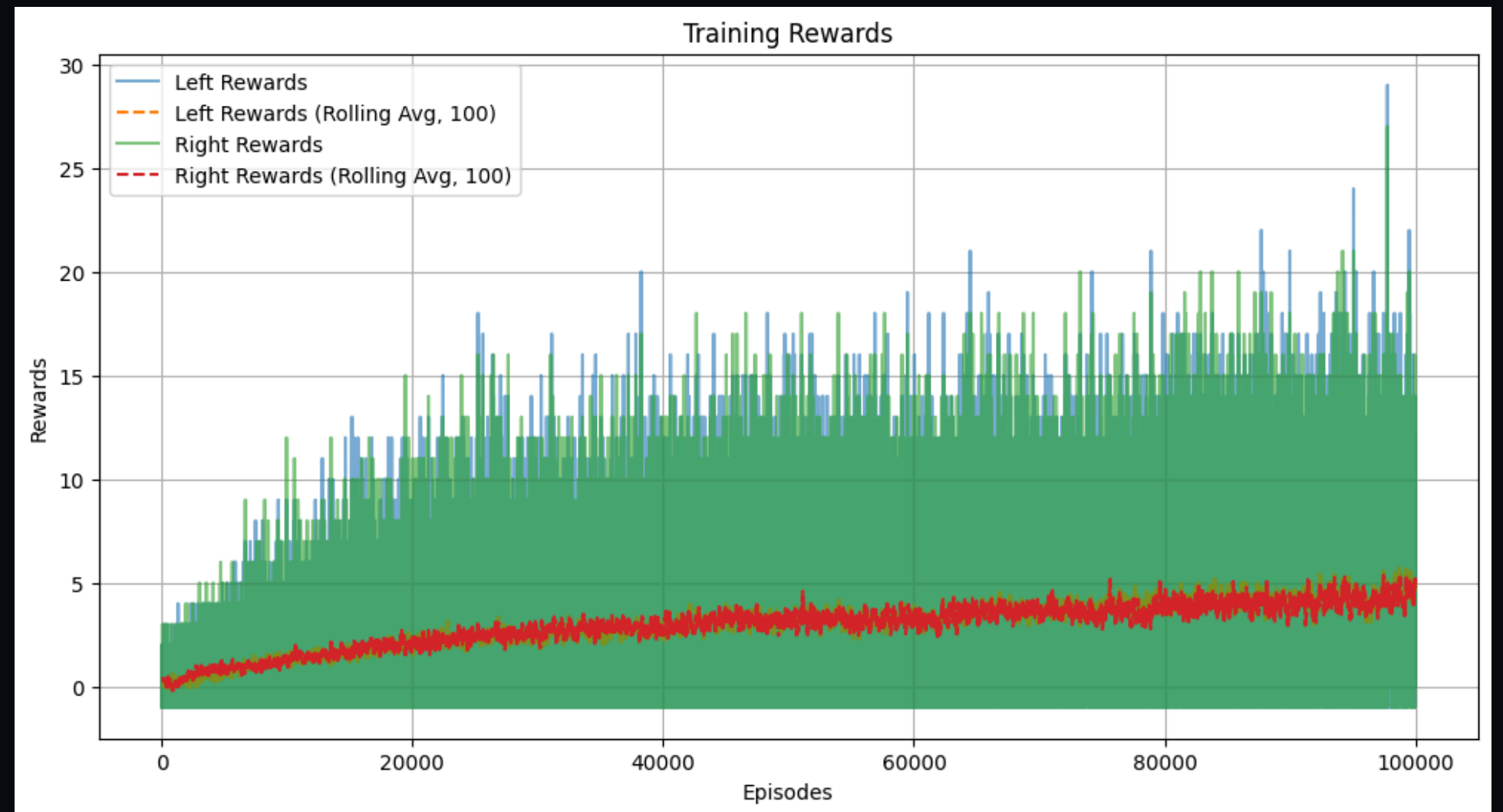
```
next_action = self.get_action(next_state)
next_q = self.q_table[(next_state, next_action)]
new_q = old_q + alpha * (reward + gamma * next_q - old_q)
self.q_table[(state, action)] = new_q
```

08

Addestramento Iterativo

L'agente apprende iterativamente attraverso un ciclo continuo di selezione delle azioni, osservazione delle ricompense e aggiornamento della Q-Table:

1. Osservazione dello Stato
2. Selezione dell'Azione
3. Ricezione della Ricompensa
4. Aggiornamento del Modello



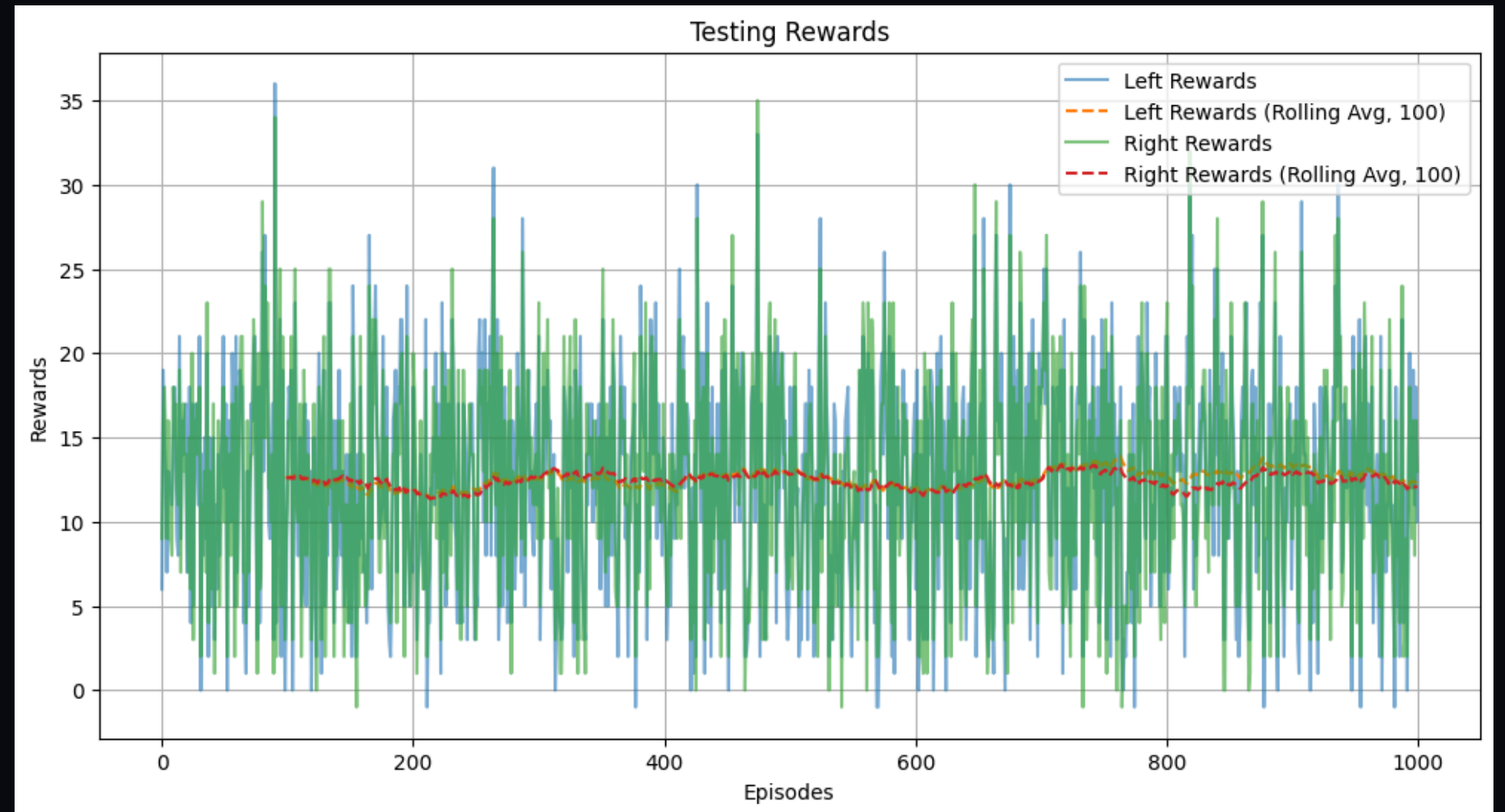
Q-Learning VS SARSA 500k episodi

Risultati

Dopo l'addestramento degli agenti utilizzando Q-Learning e SARSA, le performance sono state valutate su 1000 episodi di testing per ciascun modello.

I risultati evidenziano che il Q-Learning ha prestazioni più stabili e una media di ricompensa più elevata oltre ad una capacità di adattarsi a situazioni dinamiche grazie alla sua politica off-policy.

D'altro canto SARSA presenta delle ricompense meno consistenti, con una media inferiore rispetto a Q-Learning dal momento che la politica on-policy limita l'esplorazione efficace dell'ambiente.



Q-Learning VS SARSA 1000 episodi



Grazie

Per la vostra attenzione

Indirizzo Email

s.sirica2000@gmail.com