# DQN

Iniziamo con:

REWARDS = {

    "win": 1.0,           # vittoria

    "lose": -1.0,         # sconfitta

    "draw": 0.5,          # pareggio

    "valid\_move": 0.0,    # mossa valida ma non vincente

    "invalid": -0.5,      # mossa non valida (colonna piena)

}

In fase training:

*learning\_rate*=1e-3,

*buffer\_size*=50000,

*batch\_size*=64,

*exploration\_initial\_eps*=1.0,

*exploration\_final\_eps*=0.05,

*exploration\_fraction*=0.3,

*gamma*=0.99,

*target\_update\_interval*=500,

Il trainign si compone di 4 fasi:

**fase 1 -> DQN vs Random [100\_000 timesteps]**

**fase 2 -> DQN vs RuleBasedL1 [200\_000 timesteps]**

**fase 3 -> DQN vs RuleBasedL2 [400\_000 timesteps]**

Problema sulle mosse illegali, ne fa troppe??? Dopo proviamo aumentando penalità anche

Dopo il training attraverso file agent\_evaluator:

200 partite DQN vs Random

200 partite DQN vs L1

200 partite DQN vs L2

Vediamo che DQN non si difende bene, quindi possiamo provare a impostare delle rewards intermedie per quando attua file di 3 pedine per migliorare attacco, o quando blocca una fila di 3 dell’avversario e vediamo come varia.

REWARDS = {

    "win": 1.0,           # vittoria

    "lose": -1.0,         # sconfitta

    "draw": 0.5,          # pareggio

    "valid\_move": 0.0,    # mossa valida ma non vincente

    "invalid": -2,      # mossa non valida (colonna piena)

    "create\_three": 0.3,   # nuova tripletta creata

    "block\_three": 0.2     # tripla avversaria bloccata

}

Il trainign si compone di 4 fasi:

**fase 1 -> DQN vs Random [100\_000 timesteps]**

**fase 2 -> DQN vs RuleBasedL1 [200\_000 timesteps]**

**fase 3 -> DQN vs RuleBasedL2 [400\_000 timesteps]**

Mi chiedo se ha senso far vedere grafici rewards visto che mano mano giochiamo verso avversari sempre piu forti quindi naturalmente perde di piu rispetto ai precedenti l’agente, quindi avremo piu penalità.

E il giocare come 4 fase DQN contro se stesso ha senso? NO

Dopo il training attraverso file agent\_evaluator:

200 partite DQN vs Random

200 partite DQN vs L1

200 partite DQN vs L2

La mia domanda è: giocando con avversari come L2 non rischia di non imparare visto che perde spesso? Diamo delle rewards piccole per ogni mossa valida?

**Contro RuleBasedL2**: l’avversario blocca le mosse vincenti → il tuo agente perde molto spesso.

In questa situazione, le sue reti (DQN o PPO) ricevono quasi sempre reward = -1 (sconfitta).

Senza esperienze positive (vittorie o mosse promettenti), non può costruire una strategia utile.

**Soluzione: Curriculum Learning**

Devi far crescere la **difficoltà dell’avversario gradualmente**.  
👉 L’idea è: prima impara a vincere contro i deboli, poi affronta i più forti.

Problema del determinismo quando giocano dqn vs ppo.

 In **valutazione contro avversari rule-based o random** → deterministic=True (così vedi la “vera strategia”).

 In **match DQN vs PPO** → usa **uno deterministico e l’altro no**, oppure randomizza l’apertura della partita → così hai varietà e puoi raccogliere statistiche affidabili.

Perché abbiamo scelto di usare reti neurali, invece che ad esempio un qlearning base? Per lo spazio 6\*7 grande.

Abbiamo deciso di confrontare dqn e ppo perché una on policy e altra off policy.

Difficoltà su come impostare rewards, iperparametri, come gestire le mosse illegali (finire partita o non finirla? E se finirla sostituirla con mossa random o riprovare? E il problema che se mettevamo random usava penalità su una mossa che poteva essere anche giusta? E sul riprovare ora quante volte riprova?)

In futuro sicuramente si pitrebbero provare le CNN come miglioramento. Ora non le usiamo.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  | **X** |  |
|  |  |  |  | O |  |
|  |  |  |  | O |  |
| X | X |  |  | O | X |

***Difese e Attacchi***

* Difesa
  + Se un agente ha una fila di 3 pedine, ad esempio O ha 3 pedine

Se X blocca le 3 pedine allora -> Difesa attuata, altrimenti mancata

* Attacco
  + Se agente ha una fila di 2 pedine, ad esempio X ha 2 pedine

Se X aggiunge la 3 allora -> Attacco riuscito, altrimenti no

* + Se agente ha una fila di 3 pedine, ad esempio X ha 3 pedine

Se X aggiunge la 4 allora -> Attacco riuscito (vince anche), altrimenti no