



Applicazione del Reinforcement Learning a Tetris

Di Mattia Polticchia
Anno accademico 2019/2020

Sommario

Introduzione

Reinforcement Learning

Agente Tetris

Strumenti Utilizzati

Analisi degli Esperimenti

Conclusioni

Introduzione

Agenti basati su Machine Learning

È uno dei campi di studio nell'ambito dell' Intelligenza artificiale,
si occupa di apprendere modelli di comportamento di agenti
basati sui dati di addestramento

Quanto appreso viene poi utilizzato per guidare gli agenti alla risoluzione di
problemi

Introduzione

Tecniche di Apprendimento Automatico

Abbiamo tre principali modalità per l'apprendimento:

1. Apprendimento Supervisionato
2. Apprendimento Non Supervisionato
3. Apprendimento per Rinforzo

Reinforcement Learning

Caratteristiche:

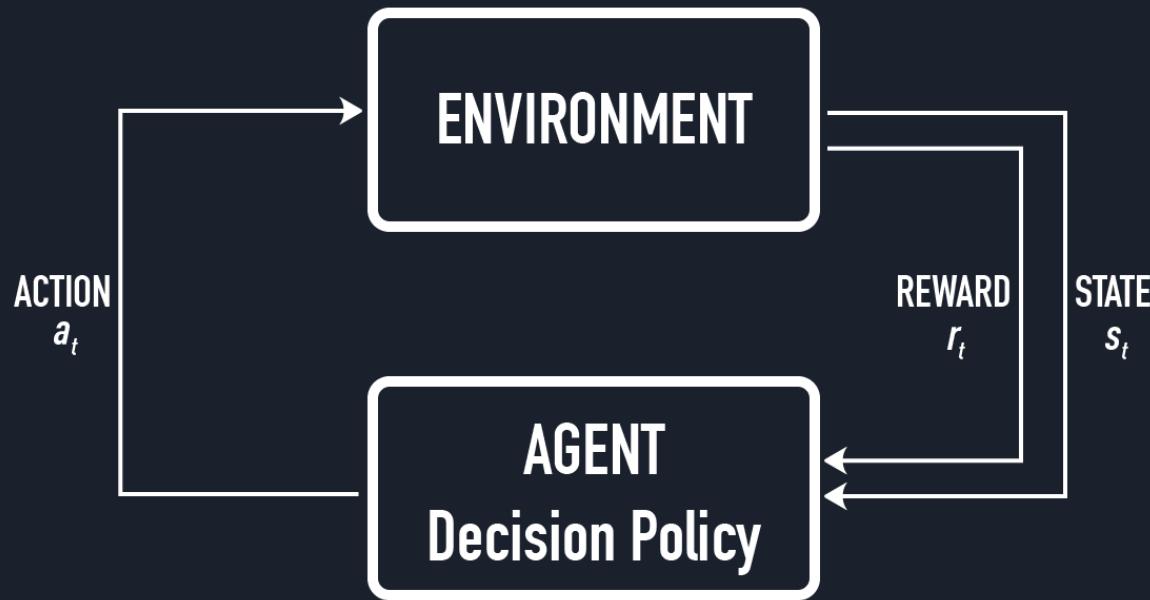
Viene utilizzato in ambienti non deterministici per apprendere una *policy*: un criterio per un *agente* che decide l'*azione* da effettuare nello *stato* corrente

L'ambiente fornisce un premio, *reward*, di entità variabile in risposta alle azioni compiute dall'agente

Questo permette all'agente di produrre associazioni azione-risultato che utilizzerà per le scelte successive con l'*obiettivo di massimizzare il reward*

Reinforcement Learning

Schematizzazione del rapporto Agente-Ambiente



Q-Learning

Questo algoritmo deve il suo nome alla Q , così definita:

$$Q : S \times A \rightarrow \mathbb{R}$$

che stima la convenienza di scegliere di eseguire l'azione $a \in A$ in uno stato $s \in S$

L'apprendimento di Q per uno stato s , avviene aggiornando Q tenendo conto

- del valore del *reward* corrente
- del valore Q atteso negli stati successivi ad s , pesato con un fattore di sconto γ
- del tasso di apprendimento α che pesa quanto il nuovo valore Q tende a prevalere sul vecchio valore

Q-Learning

Dopo un determinato numero di step t avremo che l'agente si trova in uno stato s_t seleziona un'azione a_t , osserva il reward r_t ottenuto ed entra nello stato s_{t+1} andando ad aggiornare Q

L'esecuzione termina, momentaneamente, quando s_t è uno stato finale e al valore di $Q(s_f, a)$ viene dato il valore r dello stato finale

$$Q^{new}(s_t, a_t) \rightarrow \underbrace{Q(s_t, a_t)}_{\text{Vecchio Valore}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{Tasso di apprendimento}} \cdot \left(\underbrace{r_t}_{\text{Ricompensa}} + \underbrace{\gamma}_{\text{Sconto}} \cdot \underbrace{\max_a Q(s_{t+1}, a_{t+1})}_{\text{Valore futuro massimo}} - \underbrace{Q(s_t, a_t)}_{\text{Vecchio valore}} \right)$$



Agente Tetris



Il nostro obiettivo è stato quello di addestrare un agente a giocare a Tetris utilizzando reinforcement learning basato su Q-learning



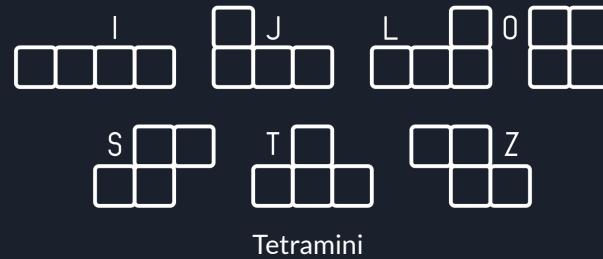
Agente Tetris

Tetris

Tetris è un videogioco inventato da Aleksej Leonidovič Pažitnov
il gioco è divenuto famoso fin dagli anni '80 nella sua versione Arcade

Lo scopo del gioco è quello di completare linee orizzontali di blocchi utilizzando i tetramini cercando di massimizzare il punteggio

Alcune versioni offrono dei punti anche per il numero di tetramini posizionati attraverso la spinta verso il basso





Agente Tetris

Considerazioni

Tetris è caratterizzabile come insieme di sotto problemi NP-completi:

- Massimizzare il numero delle linee eseguite
- Massimizzare il numero di tetris fatti
- Minimizzare l'altezza massima occupata da un tetramino
- Massimizzare il numero di tetramini posizionati prima della fine della partita



Agente Tetris

Considerazioni Finali

Le meccaniche di gioco di Tetris lo rendono un gioco in cui non si può vincere

È anche dimostrato che esiste sempre una sequenza di pezzi che porta alla fine della partita

Il numero degli stati che il gioco può ottenere è 7×2^{200} : questo rende la ricerca della policy esatta per la soluzione impossibile



Agente Tetris: Strumenti Utilizzati

01 Stable Baselines

Libreria scritta partendo dal progetto Open AI, offre la possibilità di creare agenti per il Reinforcement Learning ed interfacciarli con ambienti

02 Gym Tetris

Sviluppata da Christian Kauten basata sulle librerie Open AI.
Questa libreria fornisce l'ambiente utilizzato dall'agente.



Agente Tetris

Perché il Reinforcement Learning?

Si presta bene alla risoluzione di problemi NP, come dimostrato anche dal progetto Alpha GO

Una delle sue assunzioni è che l'ambiente rispetti le caratteristiche del processo di Markov; Tetris le soddisfa in quanto non necessita di informazioni non presenti nel gioco per prendere decisioni

Agente Tetris: Rapporto Agente Ambiente

Per permettere all'agente di esplorare gli passiamo una schermata di gioco e dei dati inerenti alla partita

In risposta l'agente fornisce una mossa di gioco che poi verrà eseguita





Agente Tetris: Q-learning

Algoritmi utilizzati:

- Deep Q-Learning

Sfrutta una rete neurale convoluzionale profonda

- Double Q-Learning

Risolve il problema della propagazione dell'errore nelle valutazioni di Q che spesso possono rallentare l'apprendimento



Analisi Dell'Esperimento

Il nostro esperimento è stato diviso in tre prove di test:

Prima prova: utilizza il DQN con i parametri di tuning forniti da Stable Baselines

Seconda prova: modifica dei parametri di tuning e aggiunta del Double DQN

Terza prova: eseguito lasciando i parametri di tuning della seconda prova,
ma modificando i tasti premibili dall'agente



Analisi Dell'Esperimento

Prima Prova

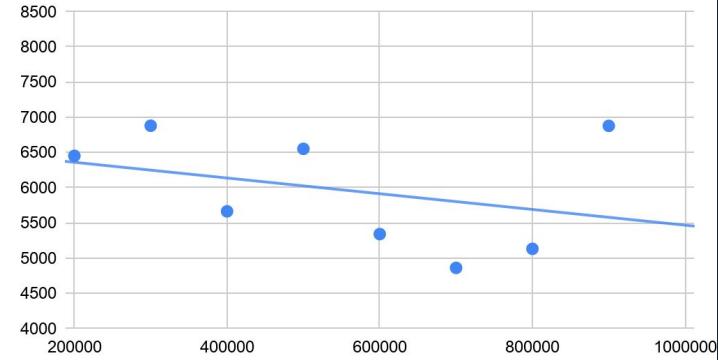
La prima prova ha avuto un buon rapporto tempo/addestramento, si è arrivati ad un milione di step

Policy appresa: Come strategia di gioco, l'agente ha iniziato ad impilare i pezzi verso uno dei due lati della griglia fino al termine della partita

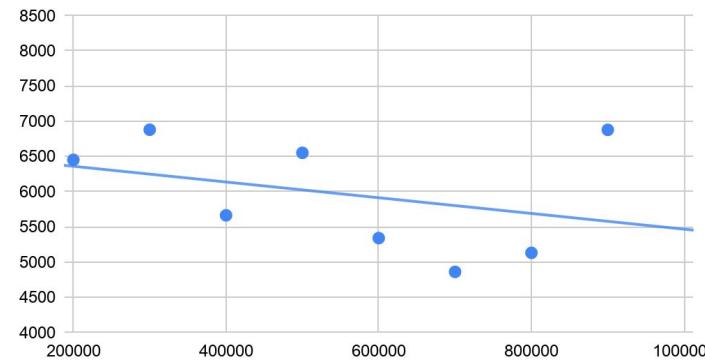
Dati Prima Prova

Learning Steps	Reward Medio	Step Medi
200000	-20	6891
300000	-20	6454
400000	-20	6881
500000	-20	5666
600000	-20	6553
700000	-20	5343
800000	-19	4864
900000	-19	5134
1000000	-20	6879

Prima Prova: Step rispetto Learning Step



Prima Prova: Step rispetto Learning Step





Analisi Dell'Esperimento

Seconda Prova

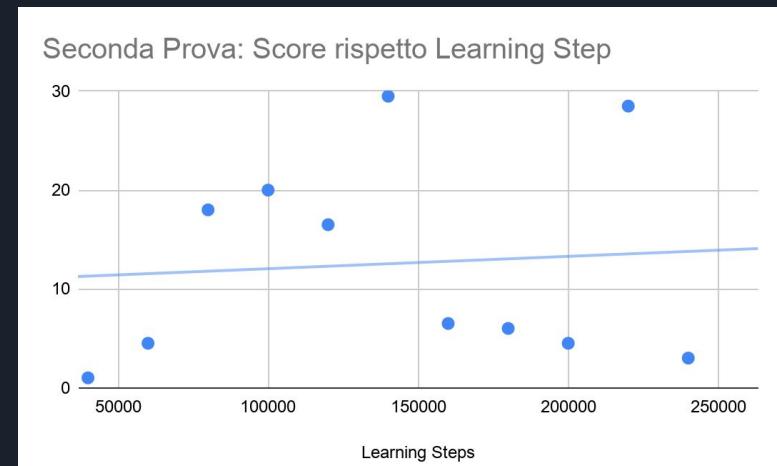
I parametri di tuning utilizzati hanno fatto aumentare drasticamente i tempi di allenamento

Policy appresa: l'agente ha “scoperto” che spingendo verso il basso i tetramini otteneva un punteggio dall’ambiente e di risposta, e ha basato su questo la sua strategia

I tempi di sopravvivenza in partita sono aumentati notevolmente

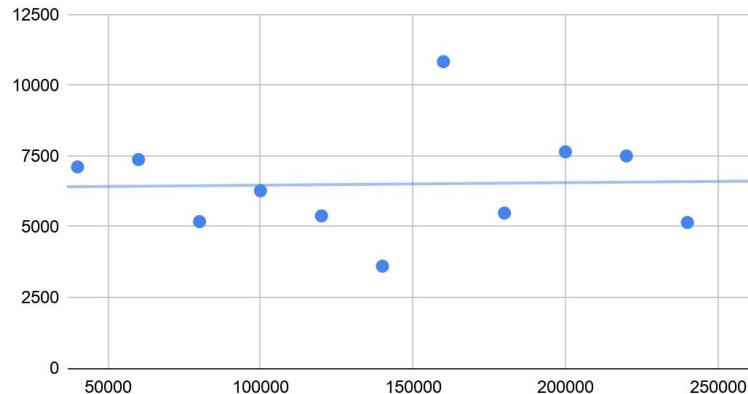
Dati Seconda Prova

Learning Steps	Score Medio	Reward Medio	Step Medi
20000	0	-20	5951
40000	1	-19	3300
60000	1	-19	7111
80000	5	-16	7371
100000	18	-2	5178
120000	20	0	6268
140000	17	-4	5381
160000	30	10	3605
180000	7	-14	10830
200000	6	-14	5480
220000	5	-16	7643
240000	29	9	7499
260000	3	-17	5148

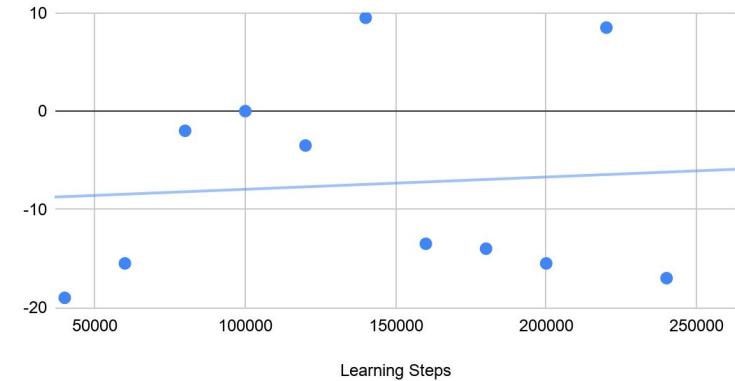


Dati Seconda Prova

Seconda Prova: Step rispetto Learning Step



Seconda Prova: Reward rispetto Learning Step





Analisi Dell'Esperimento

Terza Prova

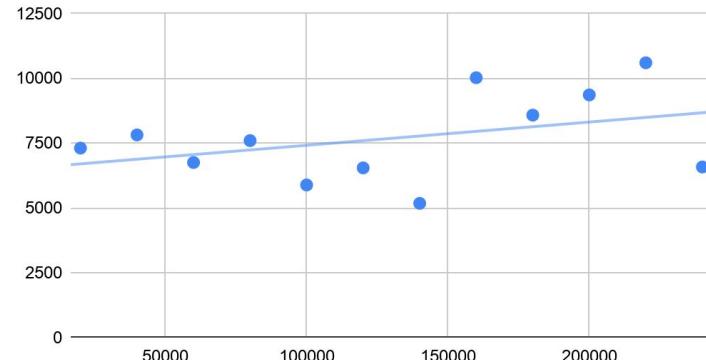
In questo esperimento si è rimossa la possibilità di ottenere punti eliminando la mossa “down” del controller

L'agente ha comunque migliorato il suo tempo di sopravvivenza ma non ha dato segni di sviluppo sulla ricerca del punteggio

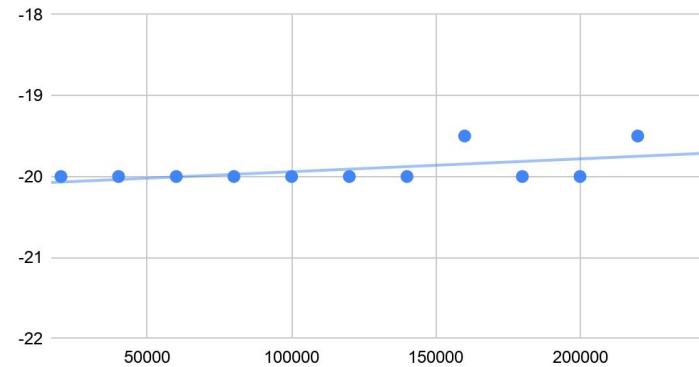
Dati Terza Prova

Learn Steps	Reward Medio	Step Medi
20000	-20	7307
40000	-20	7816
60000	-20	6752
80000	-20	7596
100000	-20	5886
120000	-20	6544
140000	-20	5177
160000	-20	10021
180000	-20	8579
200000	-20	9359
220000	-20	10597
240000	-20	6580

Terza Prova: Step rispetto Learning Steps



Terza Prova: Reward rispetto Learning Steps





Conclusioni

Tetris necessita di strategie decisionali e di sopravvivenza che prevedano tutti i possibili posizionamenti del tetramino, cercando di massimizzare i punti rispetto alla situazione attuale della partita.

Questo lo rende ancora un problema aperto nel campo del Reinforcement Learning, siamo riusciti ad ottenere dei miglioramenti rispetto al problema di aumentare il numero di tetramini posizionati prima della fine della partita

In futuro: focalizzare sui sottoproblemi NP-completi significativi del Tetris come ad esempio:

- Minimizzare l'altezza massima occupata da un tetramino
 - Massimizzare il numero delle linee eseguite
- 

Grazie
dell'attenzione

