Tetris with Reinforcement Learning

Min-Chih, Hsu
michsu.cs09@nycu.edu.tw

Chih-Yi, Chen
vicki.cs09@nycu.edu.tw

Ynu-Chen, Chang
yunchen1220.cs09@nycu.edu.tw
National Yang Ming Chiao Tung University
No. 1001, Daxue Rd., East Dist., Hsinchu City 300 , Taiwan (R.O.C.)

Abstract

We want to make Tetris – our game of memory – more easily to play by using Reinforcement Learning(RL). This paper compares the performance of Q-learning and Deep Q-learning on Tetris game, and find the better solution of agents. Our strategy would take next piece and hoding piece into consideration. Moreover, players will get extra score if eliminating rows consecutively (i.e., Combo).

1. Introduction

During this semester, we had tried to implement neural networks on Pacman, which lead to a remarkable performance. Since Tetris is one of the most popular video game in the world, created by Alexey Pajitnov, it is a complex and difficult game, so we want to utilize the deep reinforcement learning approach to overcome our childhood enemy.

To provide a point of comparison for our work, we started by developing a simple baseline, we tried to implement Q-learning by initializing a q-table, undating the action of each state for every step we took.

Our approach is Deep Q Learning, by considering more states than Q-learning, we use greedy algorithm to choose action and states pairs, while training, it would greedily pick the action with the highest score.

So far, various methods are s are able to yield reasonable policies for Tetris game. However, without considering the real-world gaming environment(such as combo and switch piece), most of them didn't act like a human expert players.

In this paper, we first briefly provide the review related to Tetris agents. Second, introduce how we implement baseline and our approach, also, how we add the strategy to make it has better performance. At last, we present our experimental results and compare the difference between algorithms.

2. Related Work

Deep reinforcement learning has been widely used on video games. One of the seminal work in this field was from DeepMind Technologies' 2013 paper [4]. This paper built a convolutional neural network and trained it with a variant of Q-learning. And their result are extremely impressive in six of the seven games it was tested on.

When it comes to Tetris, reinforcement learning algorithms have been applied on several works [1–3]. Most of these algorithms formulate Tetris as a Markov decision process thus the AI can try out all possible moves for given pieces. Some of the previous approaches [1, 3] use features relating to the bumpiness, the number of holes, and height of the game board, so the training function we use is derived and extended from these papers.

3. Methodology

We assume that our reinforcement learning task follows a finite Markov Decision Process (MDP).

The data used for our learning algorithms consists of tuples of $(previous\ state, reward, state, done)$

State There are totally 9 states we gave. Including cleared rows, bumpiness of columns, number of holes, landing height, cell transitions(row and column), amount of eroded cells, aggregate height.

Action $action = (column, rotation)$, which refer to the position and rotated mode of piece, the tuple$(action, state)$ will be sent to function to find the best action and take it. **Reward** In real world environment, player can get higher score if eliminating several rows at the same time, so we give agent extremely high reward to encourage it doing Combo action.

Epsilon-greedy Algorithm

During training progress, we use the standard epsilon greedy policy, where the agent takes the estimated optimal action most of the time and only takes random action with probability ϵ.

3.1. Q Learning(SARSA update)

SARSA is an algorithm for learning a Markov decision process policy. Our Q value updated according to the following rule:

$$Q(s,a) \leftarrow (1-\eta)Q(s,a) + \eta[r + \gamma Q(s',a')]$$

3.2. Deep Q Learning

We used Keras of TensorFlow to construct a network architecture with 4 denses, first 3 are with ReLU non-linearity, and the last one is with a linear activation. Our initial input dimension is 9 and the final output dimension is 1.

3.3. Deep Q-Learning with Holding Pieces

To meet the better performance, we take the hold action into agent's consideration.

There are 2 situations

1. There's no holding piece, if hold, we switch to the next piece.
2. There's holding piece, if hold, we change to the holding piece.

To achieve, we keep the content of **State** and **Reward**, but add aextra element in **Action**. Now, everytime when we're seeking for best action, we have to go through doubled possible states than original DQN.

Action $action = (column, rotation, change)$. The $change$ would be 1 if we're checking the possible state–$(action, state)$ of holding piece or next piece.

4. Experiments

4.1. Deep Q Learning

We use Open AI Gym environment to generate experience and store them in a replay buffer, whose capacity is 20000. The Adam optimizer is adopted to update the network parameter. To enhance the model performance, the ϵ-greedy exploration technique is applied.

With $episodes = 25$, we run 5000 games overall. As shown in Figure 1, with the increment of games (x-axis), our agent clears more lines and gets higher reward in each game.

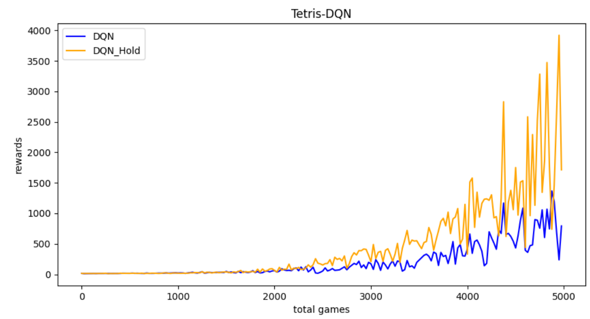

Figure 1. Comparison

DQN with hold action perform better than original DQN once total games exceed 2000. The experiment result demonstrates that our strategy of holding piece is more effective on training.

However, if we run the trained model seperately, model with hold action will end after eliminate 100 lines, as for the model without hold action, it can eliminate over 4000 lines. We guess this is because holding strategy is more complex, so it has to train more games to get our expected performance.

References

[1] Donald Carr. Applying reinforcement learning to tetris. 2005. 1

[2] Alexander Groß, Jan Friedland, and Friedhelm Schwenker. Learning to play tetris applying reinforcement learning methods. In *ESANN*, 2008. 1

[3] Nicholas Lundgaard and Brian Mckee. Reinforcement learning and neural networks for tetris. 2007. 1

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013. 1