Homework 2
CS 8803, Spring 2019
Adaptive Control and Reinforcement Learning
Georgia Institute of Technology

Designing Artificially Intelligent Tetris Players

Akshay Krishnan, Nithin Shrivatsav Srikanth and Prateek Vishal

1 Introduction

Tetris is a difficult game for humans to master. Tetris game starts with a game board with falling tetrominoes. Each horizontal line that is filled clears out. The number of such lines cleared is indicative of the score. The objective of the game is to place the tetrominoes to clear out maximum number of lines or obtain maximum score. A greedy approach could be used to solve this problem by making the best choice at any particular stage of the game. However, since the choices made at any stage affects the placement of tetrominoes after some time steps in the future, it is not easy to take decisions to place imminent tetrominoes without considering the impact of the choice at a later stage of the game. The game ends when the user runs out of legal moves. There are seven tetrominoes which can each be moved to any orientation and along any column of the Tetris board. This defines the action space. The state space is defined by the configuration of the board at any stage of the game and the falling tetromino piece. In our implementation, we have made use of Noisy Cross Entropy Method to design AI Tetris players.

2 Methodology

The state space of the MDP involved in this problem is very large and therefore, we consider policy based methods to solve this problem. We also use features to represent the state in a low dimensional manner. Ideas from [1] are used to simplify the game specifications to a form that can be solved using Reinforcement Learning.

2.1 Feature Functions

Each state-action pair (s, a) is characterized by a vector of 8 features that are computed based on the state s' that results from taking action a in state s. The features used are :

1. Landing height : The height at which the current piece fell.
2. Eroded pieces : The number of squares cleared by the piece.
3. Row transitions : Number of transitions from empty to filled cells and vice versa from along every row.
4. Column transitions : Number of transitions from filled cells to empty cells and vice versa along columns for every row.
5. Number of holes: The number of empty cells that have at least one filled cell above.

6. Cumulative wells : The sum of accumulated depths of wells (a well is an empty cell on top of a column).
7. Hole depth : The number of filled cells above a particular empty cell, over all columns.
8. Row hole : The number of rows that contain an empty cell.

These features have been proposed by experts and have been effective in modelling the state of the Tetris game. The evaluation function is a linear combination of the proposed features.

2.2 Policy

We parameterize the policy as a linear function of the features that we use to represent the state:

$$\pi_\theta(s) = \arg\max_a \theta^T f(s,a)$$

where $f(s,a)$ are the features that represent taking action a in state s, and θ are the parameters.

2.3 Optimizing θ

We use **noisy cross entropy method** as stated in [2] - a black box optimization strategy to find the optimal policy.

2.4 Rewards

The number of lines cleared gives the total reward. This is what we have used to train our Tetris player.

3 Cross Entropy Method

Cross Entropy Method is a black box optimization methodology based on importance sampling. It finds use in applications when the evaluation of the policy is not very expensive and is not prone to stochasticity present in the environment. This method finds an application in the game of Tetris where we can use the optimization strategy to search for all possible weight vectors to get the one that maximizes score.

We use Gaussians to estimate the distribution of the parameters. The Gaussian is initialized arbitrarily. Samples for the parameter θ are generated from this distribution. An elite set of these samples are selected based on the their average score on an episode of playing the game. The distribution is then re-fit to increase the likelihood of the elite set. This process is repeated for a large number of iterations (we assume that the parameters of the distribution converge). The sample with the highest score after these iterations is used as the optimal samples.

As given in [2], addition of noise term in step 13 of the algorithm significantly improves the performance of the algorithm in terms of average scores. The different noise functions used include linearly decreasing noise, hyperbolic noise and constant noise.

Algorithm 1: Cross entropy methods for policy optimization in Tetris

```
1  Initialize random $\theta \epsilon R^d$
2  Initialize random $D \sim \mathcal{N}(\mu, \sigma)$
3  Initialize iteration = 0
4  **while** $iteration < max\_iterations$ **do**
5      Sample $\theta$ $N_s$ times according to distribution $D$;
6      **for** $i$ =1 to $N_s$ **do**
7          Compute $s_i = \frac{1}{N_K}(R(\theta_i))$
8      **end**
9      Sort $\theta_i$ according to $s_i$ in descending order
10     Select $N_e$ best samples of $\theta_i$
11     $\mu \longleftarrow$ mean $\theta_{i1,to,N_K}$
12     $\sigma^2 \longleftarrow$ standard deviation $\theta_{i1,to,N_e}$
13     $\sigma \longleftarrow$ square root($\sigma^2$ + noise)
14 **end**
15 $\theta_{optimal}$ = max($\theta_i$)
```

4 Results

The Tetris Players designed produced good results and gave an average score of over 10,000. We experimented with the use of different noise functions. We have attached the results of using linear noise. On an average, when run over 20 trials, our Tetris player scored 52,862.6 with a high score of 169785.

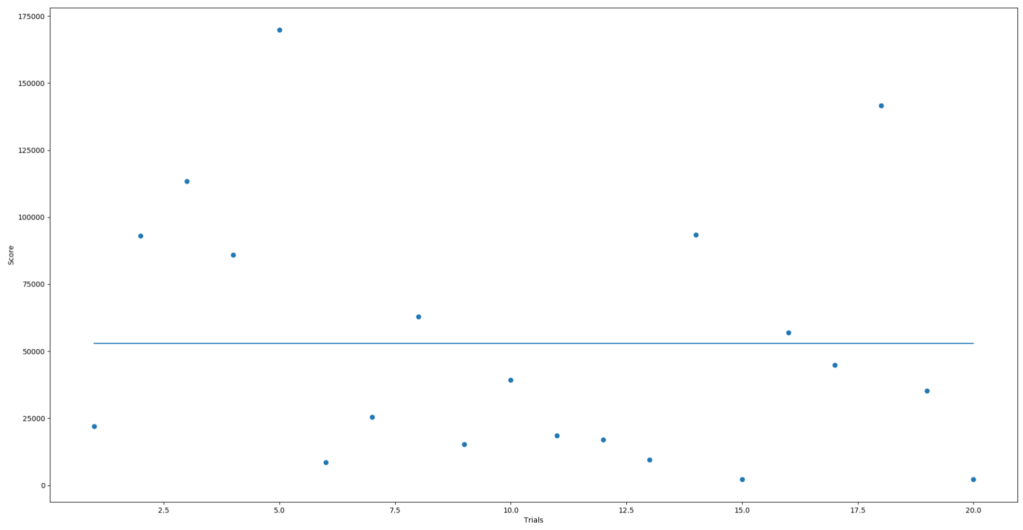

Figure 1: Scores vs Trials

5 Conclusion

We have explored Black Box optimization method to design an AI Tetris Player. Our implementation cleared an average of $52,862.6$ lines on average with a maximum score of 169785. As shown earlier, the addition of noise term makes a big difference to the convergence of the algorithm. We realized that when we used no noise or a constant noise, the results were sub optimal. This analysis also gives makes sense in the Exploration-Exploitation paradigm. The use of a decreasing noise implies that initially, we would like to explore more and after a few steps of the iteration, we would like to exploit more or in other words work with the best samples we have.

We also briefly explored Natural Policy Gradient method as given in [3]. However, we were not able to successfully train the agents to meet the objective of the game of clearing 10,000 lines on average. Further work in this direction would involve development of better strategies for the policy and making use of advantage functions instead of using reward while calculating policy gradients.

References

[1] A. Boumaza, "How to design good tetris players," *hal-00926213*, 2013.

[2] I. Szita and A. Lrincz, "Learning tetris using the noisy cross-entropy method," *Neural Computation*, vol. 18, no. 12, pp. 2936–2941, 2006.

[3] S. Kakade, "A natural policy gradient," in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, (Cambridge, MA, USA), pp. 1531–1538, MIT Press, 2001.