CSE/AMATH-579 Intelligent Control Through Learning & Optimization

Homework 3: Tetris

Instructor: Byron Boots *Due: 1:29 pm November 21, 2022*

Instructions

Submit your write-up and code in a `.zip` file through Canvas. Your submission should follow the structure below:

```
[YourUWNetID].zip
├── [YourUWNetID].pdf (your 4-page report, MUST be typeset, LaTeX highly recommended)
├── player.py (your tetris player, more instructions in the main text)
├── [your training scripts, if not included already in player.py]
└── [any auxiliary files, e.g. saved weights, helper functions, etc.]
```

This assignment may be completed in teams of up to 3 students. Note that this assignment may take significantly more time than previous assignments. Please plan accordingly. Since this is an open-ended assignment, we will NOT be able to answer questions specific to your implementation. However, we can provide help on high-level ideas.

Policy on using external resources: You are expected to implement your own Tetris player **WITHOUT** using any existing implementations of optimization algorithms, e.g. from `scipy.optimize`, or RL algorithms. The only exception is that you can use auto-differentiation libraries, e.g. `tensorflow` and `pytorch` to train your neural networks (if you choose to use them).

1 The Problem

The goal of this project is to build a Tetris-playing learner. The assignment is straightforward: build a learning algorithm that improves with experience. Your goal should be to build a learner that can clear approximately $10,000$ lines before losing.[1] You have wide latitude to design a player. Think carefully about how to specify the cost function, the state representation, and the learning algorithm itself. The game is implemented in OpenAI Gym. You must use the included implementation of the game and submit a **python** `player.py` for the given implementation. This file should load your agent, run 20 games and report the average reward. Note that this script should not run training from scratch. However, you may wrap the code in any language of your choice during training.

[1] Depending on your choice of algorithm and your computational resources, you might have to train your model for **days** before you start clearing many lines. So plan accordingly!

2 System Description

You will work with a simplified version of the game of Tetris. In Tetris, a random sequence of Tetriminos fall down the playing field. The objective of the game is to manipulate Tetriminos by rotating and shifting them so that they form a solid horizontal line with no gaps. When such a line is formed, it disappears and any blocks above it fall down to fill the space. In the simplified game, there are no dynamics: you simply rotate and shift the Tetriminos and then place them in the column of choice.

3 What to Turn in

Turn in a zip file containing your **code** and a **4 page report** describing what you implemented.

4 Evaluation

4.1 Performance

We will evaluate the performance of your Tetris player based on the maximum and average lines it clears. As a rough estimate, a good player would clear approximately 10,000 lines (or more) on average – you should aim for this to get full credit for performance. A middle-of-the-road player would probably clear around $1,000 - 5,000$ lines, and a poor player would clear less than 200 lines.

4.2 Report

While there is no fixed template for the report, it should be structured as a scientific paper that covers (but is not limited to) the following:

- **Related Works**: Briefly summarize some approaches that have been tried in the literature ([1] is a great resource). This does not have to be comprehensive, but is intended to help you think of different strategies. Since training your models can be quite time consuming, we encourage you to think through different approaches before settling on your final algorithm.
- **Cost Function**: Describe what you used as a cost function and why.
- **Rewards**: Provide concrete definitions of your rewards[2], and include some motivation for why you picked them.
- **State Representation**: Describe how you represent state. If your algorithm computes features from states, clearly define them and provide motivation for why you picked them.
- **Approach**: Clearly describe your approach and the algorithm you used. Note that you do not necessarily have to use RL to solve this problem (e.g. you can use a variant of the cross-entropy method [1]).

[2]Note that you need to specify the reward function in `TetrisEnv._get_reward()`. The given environment only returns zero reward.

- **Evaluation**: Describe the performance of your agent, and make sure to include maximum and average lines cleared over 20 games.

References

[1] Boumaza, Amine. "How to design good Tetris players." (2013). `https://hal.inria.fr/hal-00926213/document`