Reinforcement Learning 2020: Assignment 4
Self-Play

Aske Plaat
rl@liacs.leidenuniv.nl

April 15, 2020

1 Introduction

The objective of this assignment is to build intuition on deep reinforcement learning in games, where self-play is used to create an environment for generating the rewards for training. You will implement a deep neural network player for the game of Hex and experiment with various self-play algorithms.

Self-play is old. Samuel's Checkers player in the 1950s used a rudimentary form of self-play to learn a better evaluation function. In the early 1990s, Tesauro's TD-Gammon used self-play to create a Backgammon player strong enough to beat the human champion. However, without a doubt, the most famous self-play success is AlphaGo Zero, the Go playing program that taught itself to play Go entirely by self-play, without any heuristic domain knowledge, *tabula rasa*.

Deep reinforcement learning is computationally intensive, and self-play is even more so. Therefore we will choose Hex as our game, a simple, fast game, and we will choose a small board size, 7x7, to allow for plenty of training to occur.

DeepMind has not published the source code of AlphaGo. However, the scientific publications contain enough detail for many re-implementations to have been created, most of which are public.

For this assignment, you may find the following resources useful:

- Chapter 7 of Learning to Play.
- There is a variety of existing AlphaZero implementations from which you may pick one
- Shantanu Thakoor, Surag Nair, Megha Jhunjhunwala and others have written AlphaZero General (A0G), a re-implementation fully in Python. A0G runs on PyTorch, TensorFlow, or Keras. Implementations of 6x6 Othello, Gomoku, Tic Tac Toe, and Connect4 are available. The Python code is extensible for your own game implementation. A writeup and documentation are available on the github site.

- PolyGames by Facebook

When you get stuck, your first resource should be the book for this course. If you find that confusing or it does not answer your question, please write an email to `rl@liacs.leidenuniv.nl`. Consult the resources of the assignment, ask questions at the classes, or the question hour.

2 Hex - 3 points

Install an AlphaZero implementation and familiarize yourself with it. Run the 6x6 Othello player, play against a trained player. Look at `Coach.py` for the main structure of the self-play, and train a player, see if you can follow the training steps.

Implement a 7x7 Hex player via AlphaZero. It helps self-play training to use a GPU. See the previous assignment for tips on using a GPU.

Learn the Hex player by self-play. Write scripts to automate playing against each other.

3 Tournament - 3 points

Perform a thorough analysis of the performance of the learned player. Run a tournament to determine the Elo rating of the following players:

- ID-TT alpha-beta Hex player
- MCTS Hex player
- AlphaZero self-play Hex player 1
- AlphaZero self-play Hex player 2

Use the 7x7 board size. Plot Elo graphs. Perform a computation how many games should be played.

Learn TWO AlphaZero players independently but equally long. See if they are equal in strength or whether they differ significantly. If so, what does this tell you about the learning process?

If the performance of the self-play player disappoints, consider a third player, that has learned longer.

4 Hyperparameters - 3 points

See if you can improve the performance of the Hex player. You can find default values to most of these hyperparameters in `main.py`. However, hyperparameters aren't the only thing that influence the training process. Report in detail on everything you tried and its effect.

Again, make a plan first, making a rough estimate of the training time, and evaluation time this will take. It is better to do a small experiment well than to be too ambitious and running out of time.

Report on your results, using the scripts of the previous section.

5 Report

Your submission should be a self-contained report that should be at least 6 to 8 pages if not even more with figures etc. The page amount we expect of you might vary depending on your layout. Your code should be uploaded to blackboard and ready to run and free of syntax errors. Your report should contain:

1. Make sure that your report is a self-contained document with an introduction, methods and conclusion, so even someone not following the course would understand what you did and why
2. Also report on your general approach and issues you encountered
3. Be sure to properly reference all information that wasn't gained as insight through your data or is describing your experiments. Especially things you believe to be common knowledge, e.g. infos about AlphaZero or MCTS
4. For AlphaZero Hex implementation: relevant source code files, ready to interpret and run, on blackboard. Short user documentation in report
5. For all sections: Supporting scripts for running, testing, and performance evaluation on blackboard. Short user documentation in report
6. Tournament results (approach, positions used, outcome)
7. For Hyperparameter tuning: Report on experiment design and results
8. One overall conclusion, lessons learned, observations

Self-play is computationally demanding. Your compute resources are limited. Choose your experiments carefully and realistically. The grading for this assignment will take the computational resource in account (do not try to do the impossible but make the most of the resources you have). Describe carefully what you do, and what the outcome was, and what you think might be going on will get you points.

The report must be handed in on **15 May 2020 before 23:59**. For each full 24 hours late, a full point will be deducted.