Reinforcement Learning 2020: Assignment 2
Adaptive Sampling

Aske Plaat
rl@liacs.leidenuniv.nl

March 11, 2020

1 Introduction

The objective of this assignment is to build intuition on a reinforcement learning algorithm, and to experiment with central reinforcement learning concepts such as the exploration/exploitation trade off and averaged sampling to approximate the value function. You will implement a Monte Carlo Tree Search based Hex player and experiment with its performance.

You will build on the experience with a Hex program and with Elo-based performance evaluation from the previous assignment.

Monte Carlo Tree Search, or MCTS, was introduced in 2006 and has since been used in many applications. Its advantage is good performance in applications where it is hard to find a heuristic evaluation function. It has propelled the field reinforcement learning significantly, and artificial intelligence in general, by showing that great successes of heuristic planning (such as in Chess) can also be accomplished without the hard work of building domain specific heuristic functions.

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

- Chapter 5 of Learning to Play.
- the MCTS website here by Ankit Choudhary. It provides a nice example run of MCTS, and links to a Pyhton implementation of MCTS.
- re-use relevant parts of the Hex and Elo implementation of assignment 1.

When you are 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 MCTS Hex - 3 points

Implement an MCTS based Hex engine. You may re-use the user interface, move generator and other useful parts of assignment 1. The minimax search and heuristic evaluation function will be replaced by MCTS.

The objective of this assignment is to become familiar with the MCTS algorithm. Part of this is to implement the algorithm yourself. Calling functions from another package is not the objective, you should study other implementations, but then implement all functions yourself.

Test the program by playing against it manually, give it test positions, and run it against the minimax player from assignment 1. Challenge: How do you test an algorithm with a non-deterministic random play out?

3 Experiment - 3 points

Re-use the Elo script to evaluate the strength of your program. Play the following programs against each other:

- ID-TT alphabeta player against MCTS player

Determine the Elo rating of each program. How many games should be played for the rating to stabilize statistically? Vary resources to your program: number of playouts (N-simulations) for MCTS, search depth of ID-TT alphabeta player. You can also choose for equal-time (per move) for the two players.

4 Tune - 3 points

MCTS has two important parameters that influence performance: the number of simulations N that it is allowed to do, and C_p the UCT exploration/exploitation parameter. See also Section 4.5.1 of Learning to Play. We call N and C_p hyperparameters, because they determine the working of the algorithm in general, and are on a higher level than the regular function parameters.

Perform a hyperparameter analysis of these two hyperparameters N and C_p. Hyperparameter analyses can be computationally intensive (they take long to run).

Choose your experimental design carefully and report on your design choices in great detail.

Choose wisely. "The best is the enemy of the good"—Voltaire, a.k.a. "Premature optimization is the root of all evil"—Donald Knuth. No experimental results to report because of lack of computational resources, no points. Better to be able to report on some reasonably adequate results that are computationally feasible, than a perfect design that is computationally infeasible.

5 Report

Your submission should consist of a report in pdf of at most 8 pages with Elo graphs of the experiments and a textual report on your findings, explaining the outcomes of the experiments, and the implementation process. Your code should be uploaded to blackboard and ready to run and free of syntax or other errors. Your report should contain:

1. General approach, issues encountered
2. For MCTS implementation: Source Code, ready to interpret and run, on blackboard. Short user documentation in report
3. For all sections: Supporting scripts for running, testing, and performance evaluation on blackboard. Short user documentation in report
    - This means that we want to be able to know how to reproduce the results of your experiments
    - e.g., if you have one file per experiment: For task 4 run "tune_mctst.py" and find the result in figure.png / data.json
    - e.g., If you like writing DRY code and everything is in one file: For task 4 run "a2.py –task tune" and find the result in figure.png / data.json
4. For MCTS implementation: Test results (approach, positions used, outcome)
5. For Hyperparameter tuning: Report on experiment design and results
6. One overall conclusion, lessons learned, observations

The report must be handed in on **20 March 2020 before 23:59**. For each full 24 hours late, a full point will be deducted.