Informed Search Methods in AI

Fall 2023 Assignment 4

In this assignment, you will experiment with augmenting Breakthrough game-playing agents with an Alpha-Zero style neural network to evaluate both game states and move policies. You are provided with a pre-trained network.

You will implement two different agents.

A4GreedyAgent (agents/a4_greedy_agent.py):

This agent plays greedily, but differently depending on its play mode (set by the 'play_mode' parameter).
In play mode:

- *0:* The agent plays a move uniformly at random.
- *1:* The agent prefers an immediate winning move (if one exists), otherwise a capturing move (if one exists), but else any other move. If many equally preferable moves exist, the agent picks amongst them uniformly at random.
- *2:* The agent uses an evaluation function to evaluate non-terminal states. It expands all possible moves at the current position, scores the terminal states according to their outcome, and the non-terminal states according to the evaluation function (the score is always from the player's perspective to move). It picks amongst equally good moves uniformly at random. This is essentially a one-ply look-ahead search.
  The evaluation function is based on the players' *piece-count difference*, e.g., if *white* has two more pieces than *black* in a given position, it evaluates that position as +2 from *white*'s perspective (and -2 from *black*'s).
- *3:* The agent acts the same as in mode 2, except it uses a *neural-network-based evaluation function* (using the value-head) to score non-terminal states. The network's value is always from the player's to-move perspective. It picks amongst equally good moves uniformly at random.
- *4:* The agent picks the best available move according to the network's policy head, playing the best one. It picks amongst equally good moves uniformly at random.

A4MABAgent (agents/a4_mab_agent.py):

This agent plays according to a multi-arm-bandit strategy (like MCTS, except it does not expand a tree). The 'play_mode' parameter sets the play mode.

The agent runs a fixed number of simulations (set by the 'abort' and 'number' parameters). In each simulation, it first selects a move to simulate (somewhat differently based on the play mode). The simulation playout from that move to the end of the game uses a naïve uniformly random playout policy (provided).

After finishing all the simulations, the agent returns the most visited move.

In selection strategy depends on the play mode:

- *0:* The agent uses the UCT formula for selecting a move to simulate.
- *1:* The agent uses the PUCT formula for selecting a move to simulate. The priors (*label.p[]*) are set in accordance with the neural network's policy head.

Report (report.pdf)

In your report (3-4 pages), briefly describe how you implemented the agents and report on their performance:

- For the *A4GreedyAgent*, play a tournament amongst them all and rank them by performance. Each pair of players should play at least 100 games against each other.
- For the *A4MABAgent*, play a between them two and rank them by performance (each match should consist of at least 100 games). Run several such competitions using different numbers of simulations (e.g., 100, 200, 500). Also, put some effort into choosing a suitable exploration-vs-exploitation C parameter for your selection formula (mention in your report how you did that). Note that the two play modes do not necessarily require the same C parameter.

Hand-In

In *Canvas*, hand in your *report.pdf* and the *a4_greedy_agent.py* and *a4_map_agent.py* files.