Deep Reinforcement Learning Othello

Thesis subtitle

Master Thesis

Thomas Steinmann

University of Bern

Month and Year

Abstract

Abstract (max. 1 page)
Name of the Supervisor, Group, Institute, University, Supervisor
Name of the Assistant, Group, Institute, University, Assistant

Contents

1 Introduction

With recent successes such as AlphaGo defeating the reigning world champion and fast progress towards fully autonomous cars by Tesla as well as Weymo it is clear that Reinforcement Learning is the solution to many problems that could not even be tackled before. Many of these powerful implementations rely on equally powerful machines in order to train them, often requiring over hundred CPUs and GPUs for weeks on end. Inspired by these grand achievements and the technology behind them but lacking comparable resources I settled on a more achievable goal: Othello. This simple board game has accompanied me since the first semester when we were tasked to implement a search based Othello player. During my masters studies I suspected that a superior player could be created using machine learning techniques. This thesis documents the way to such a player and its key components.

2
Related Works

2.1 Othello

According to [3] Othello is a game played on a square board, usually made up of eight by eight tiles. The opening position is shown in Figure 2.1 Othello stones are black on one side and white on the other.

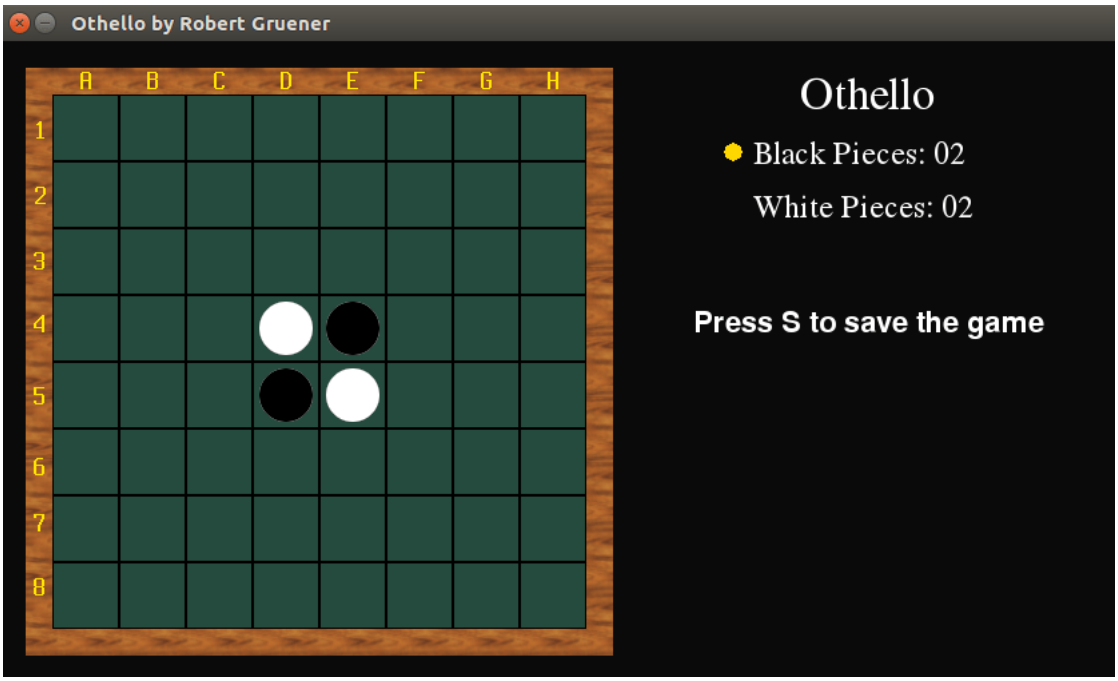

Figure 2.1: The opening position

Players take turns in placing one of these stones in their respective color on the board. Stones can only be placed in such a way that the new stone *traps* one or more of the opponents stones inbetween itself and any other stone of the same color. All opposing stones that were trapped by placing the new stone are flipped and now belong to the player who trapped them. If a player cannot perform a legal move he simply passed and his opponent places the next stone. The game ends if both players pass successively or there are no free tiles left on the board. The player who controls more stones at this time wins the game.

2.2 Heuristic Player

The player we call the *Heuristic* Player in this thesis is often used for benchmarking because of its decent performance, low computational cost and deterministic nature. It utilises a heuristic table which assigns a value to each tile on the board. These values are symmetrical for eight axes through the board. The table gives high values to tiles such as the edges that are of high value because they lead to stones that cannot be captured later in the game. Tiles around the edges however receive a low value because occupying one of them may allow the opponent to occupy the adjacent corner.

2.3 Search based Algorithms

Othello is widely used to teach search based game theory algorithms, most notably the min-max algorithm and its optimization the alpha-beta pruning. [4] The general idea of these methods is to build a search tree of all relevant moves for both players and the resulting game states. Ideally a complete graph for such a game could be computed and a computer player would just choose a move that results in a win for every single turn. However the search space for most games is so big that such a graph is not feasible with the computational resources available. The ancient chinese game of Go for example is said to have more possible game states than there are atoms in the universe. The challenge lies therefore in approximating a complete graph as close as possible and with maximal efficiency. For this the search tree is usually truncated at a certain depth or after a given time has passed and the rest of the tree is approximated with a so called value function that evaluates the value of the state where the tree stopped. This value function traditionally consists of a set of given features such as possible moves for each player, save stones that cannot be captured again or just greedily the number of stones a player controls. Search based algorithms can also be used to improve the *Heuristic Player* described in section 2.2. In most search based algorithms this would make the player nondeterministic however.

2.4 Machine Learning based Algorithms

Machine learning techniques for Othello are related to search based algorithms but often do not perform a search at play time. A straight forward approach is to inprove a search based algorithm by learning its value function. [1] used machine learning to learn his own heuristics function, similar to the one used by the *Heuristic Player*, in his alpha-beta search algorithm. A more advanced technique basically moves the search from play time to training time by simulating thousands or millions of games and uses them to train an agent. This agent learns a value function that captures the knowledge gained during the search and predicts the value of a possible move without having to perform another search at play time. Such an approach was used by [2] to master Go. They went one step further still and used training time search in order to train an agent and then improved that agents decision again with another search at play time.

2.5 Reinforcement Learning

Reinforcement Learning levarages special algorithms to train an agent on a task while performing it. In regular intervals the agent receives feedback, reinforcing behaviour that leads to good results while discouraging less performant ones. This is handled by awarding a positive or negative reward respectively, whenever the algorithm can determine something good or bad happened based on the agent's actions.

2.5.1 Monte Carlo Learning

Monte Carlo Learning is the most basic variant of a group of reinforcement learning algorithms called Monte Carlo Methods. In contrast to many other reinforcement learning algorithms they do not require full knowledge of the environment that they are applied o. Instead they work on experience alone. This experience can be gathered in the real environment and learning can therefore be done on the fly, without any prior knowledge. Another very powerful approach is to simulate the environment which allows for simpler aggregation of training data that would be difficult to obtain in the real world, such as driving cars, steering rockets or playing millions of board game iterations on a physical board.

In Monte Carlo Methods a task can be split in two major parts. First the reinforcement algorithm which is generating training data from real or simulated experience and an according label. Second a supervised learning task where an agent is trained on the data generated from the first part. Those two parts are mostly independent of each other, as long as the structure of in and outputs match. This means that the same reinforcement algorithm can be used with a variety of different learning agents such as, in the case of this thesis, multiple different neural networks.

The basic Monte Carlo Algorithm simply simulates an entire episode, evaluates the final state and labels each training sample with the resulting label. In a board game this would mean playing a single game, computing the winner and labeling the game state for each time step with the final result.

2.5.2 Temporal Difference Learning

The Temporal Difference algorithm is a variant of the basic Monte Carlo algorithm decribed above. In contrast to its predecessor it does not assign the same label to each game state but rather computes each label based on the prediction of the subsequent state's value. This can be better understood when analysing the algorithm from the end of an episode. The final label is still the result of the game just like in the basic Monte Carlo algorithm. check this againThe label of the second to last game state however is computed from its current value as well as the value of the final state. This allows for a so called online version of the algorithm which does not have to complete an entire episode before it can learn from the gathered experience, but can apply changes at every time step of the episode. Although the online version has weaker convergence guarantees and since standard Othello episodes have a maximum length of 30 game states per player there is a guaranteed reward in relatively few time steps which makes offline Temporal Difference Learning more suitable for this thesis.

2.5.3 Monte Carlo Tree Search

3
Thesis Objectives

1. Framwork for othello agents 2. Playground for RL algorithms, network optimizations, regularizations, etc. 3. High performance Othello Agent

3.1 Othello

One of the main difficulties in applying reinforcement learning to Othello is the very delayed and sparse reward. The only evaluation the learning algorithm can make is the result of the game, every other measure would introduce prior knowledge and therefore influence the strategy that is learned by the player. This constrains the learning algorithm to a single reward per game, rather than one for every action taken like for example in a jump and run game.

4
Implementation

5
Validation

6
Conclusion

7
Future Work

Bibliography

[1] Kevin Anthony Charry. An intelligent othello combining machine learning and game specific heuristics. Master's thesis, Louisiana State University, 2008.

[2] Demis Hassabis et al. Mastering the game of go without human knowledge. *Nature*, 2017.

[3] British othello federation. `http://www.britishothello.org.uk/rules.html`.

[4] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, third edition, 2009.