ANALYSIS OF MACHINE LEARNING APPLIED TO BOARD GAMES

Adam Amanbaev, Jonathan Hallström, Alvar Edvardsson
Hugo Åkerfeldt, Romeo Patzer

Supervisor: Ulf Backlund

ABSTRACT

In October 2022 DeepMind announced an artificial intelligence in a preprint [1] that was called AlphaTensor. It utilized *artificial neural networks* in combination with *heuristic algorithms* in order to solve the unsolved problem of finding faster algorithms for matrix multiplication. AlphaTensor was succesful and has shed light on the process of finding new algorithms. In this paper we aim to study the effectiveness of different *neural networks* and *heuristic algorithms* such as the one used in AlphaTensor. More precisely, we intend to analyze the efficiency of those networks and algorithms in combination with varying *optimizations, parameters, hyperparameters* and *architectures* applied to the classic game *Connect Four*.

Keywords — Machine Learning, Supervised Learning, Reinforcement Learning, Neural Network, Deep Learning

Preface

The only true wisdom is in knowing you know nothing.

–Socrates

Background

The concept of artificial intelligence was only popularized during the first half of the 20th century through science fiction. Computers before 1950 lacked key prerequisite for intelligence: they could not store commands, only execute them immediately. Allen Newell, Cliff Shaw and Herbert Simon created in 1956 what is often considered the first artificial intelligence program and it catalyzed the next 20 years of research in this field. Gordon Moore predicted in 1965 that the memory and speed of computers would double every year and the prediction, known as Moore's law, has been true ever since. The growth led to new possibilities for the field of artificial intelligence and many landmark goals were achieved during the 1990s and 2000s. In 1997 reigning world chess champion and grandmaster Gary Kasparov was defeated by IBM's computer program Deep Blue [2]. 19 years later Deepmind's artificial intelligence AlphaGo [3] beat the South Korean grandmaster Lee Sedol regarded as one of the best Go players of all time. Go is due to its large search space considerably more complex than chess and AlphaGo taught itself how to play without any proprietary knowledge in comparison to Deep Blue that was heavily influensed by the pre-coded strategies. Shortly after the success of AlphaGo Deepmind began working on protein structure prediction. The artificial intelligence AlphaFold [4] is now by a huge margine the most accurate in prediciting protein structures and it is accelerating research in nearly every field of biology. As this field of research is becoming a substantial part of today's science we find it important to analyze different aspects of it.

Summary of Notation

I am a forest, and a night of dark trees: but he who is not afraid of my darkness, will find banks full of roses under my cypresses.

— Friedrich Nietzsche, Thus Spoke Zarathustra

Contents

§1. Introduction

> A thinker sees his own actions as experiments and questions–as attempts to find out something. Success and failure are for him answers above all.
>
> — Friedrich Nietzsche

§1.1 Background

Artificial neural networks [5] are computing systems consisting of connected nodes called artifical neurons and are inspired by biological neural networks that constitue human brains. The artificial networks have recently been used in different forms in order to solve various complex and abstract problems. A recent example is Deepmind's AlphaTensor [1] which in October 2022 found a faster algorithm for matrix multiplication.

The greatest factors of an artificial neural network's success is its architecture and how it is applied to the problem presented. AlphaZero [6] and AlphaGo [3] are computer programs whose artifical neural networks only differ in the method of application. Even though AlphaZero uses the same type of network it beats AlphaGo 100 to 0 in the board game Go which demonstrates how delicate artificial networks are. In this project we take a closer look at the intricacies of artificial neural networks with the help of two board games, Connect Four and Othello.

§1.2 Analysis

In this paper we aim to analyse the effects various algorithms, parameters and training methods have on rate and quality of learning of an artificial intelligence with the help of a board game.

§1.3 Delimitations

§1.3.1 Choice of Game

This project is limited to a less complex game due to the limitations of computing power and time. Board games such as chess or shogi have large search spaces due to the number of different moves possible from each board state and are therefore not suitable for this project.

§1.3.2 Connect Four Implementation

No time will be put into making our *Connect Four* board user-friendly since it is not the purpose of this project. Time will on the other hand be spent making the board and its operations efficient in order to make training and playing consume less time.

§2. Connect Four

Connect Four is a two-player connection game, in which two players take turns dropping tokens of their chosen color into a vertically suspended grid with seven columns and six rows. A dropped token takes the lowest available space within the column it was dropped in and the objective of the game is to be the first to form a line of four tokens of the chosen color. Either horizontally, vertically or diagonally. Each player starts the game with 21 tokens of their chosen color. Let the colors, without loss of generality, be black and white and let the player with the black tokens be the first player. If all of the 42 spaces in the grid are filled and none of the players has won the game is drawn. Following are five example games. The first four games are won by the player with white tokens by having creatied a line of four white tokens vertically, horizontally and diagonally respectively. The last game where both players have used their 21 tokens and not won illustrates a tie.

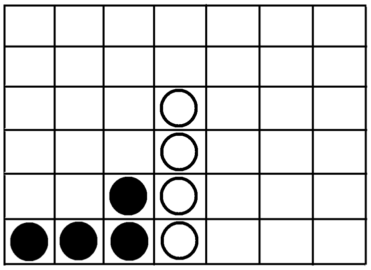

Figure 1: The player with white tokens wins with four white tokens connected in a line vertically

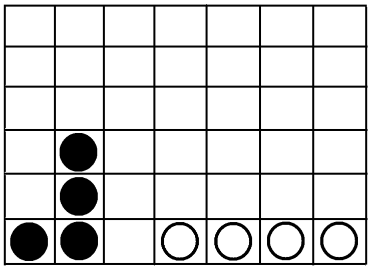

Figure 2: The player with the white tokens wins with four white tokens connected in a line horizontally

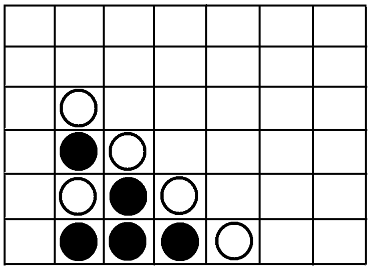

Figure 3: The player with the white tokens wins with four white tokens connected in a line diagonally

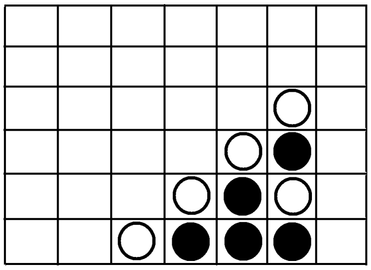

Figure 4: The player with the white tokens wins with four white tokens connected in a line diagonally

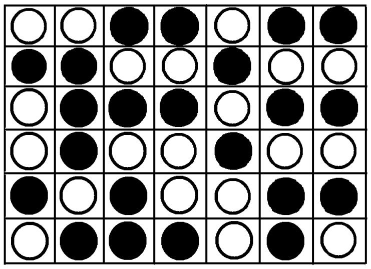

Figure 5: All 42 tokens have been used and no line of four equal-colored tokens was created. It is therefore a tie.

§3. Reinforcement Learning

§3.1 Introduction

The nature of learning is often associated with the foundational idea of learning by interacting with the surrounding environment. When a child learns how to ride a bicycle, the child either falls or manages to stay in balance. Everytime parents speak to their children they indirectly correct the children's sentences by repeating them correctly. Constantly interacting with the environment produces a wealth of information about consequences of actions, about the cause and effect of them and about what actions to take in order to achieve a certain goal. These kinds of interactions undoubtedly play a major role throughout our lives as we learn about ourselves and the environment surrounding us. Whether it is learning how to ride a bike or how to speak, we are always aware of how the environment responds to our actions. By behaving in certain manner and choosing actions carefully, we can reach the goals we seek. *Reinforcement learning* offers a computational approch to learning from interaction. By primarily exploring idealized learning situations and evaluting the effectiveness of various learning methods, *reinforcement learning* focuses a lot more on goal-oriented learning from interaction than any other machine learning approach.

§3.2 Examples

Following are examples in order to better grasp the concept of *reinforcement learning* and how it could be applied to real life problems.

- Adam left his house five minutes ago and is on his way to school. He suddenly starts questioning whether he locked the front door or not. Turning home would at this point of time make him late to class but leaving the house unlocked has its risks. Adam makes his decision based on the time he gets home after school and how often he has actually forgotten to lock the door.

- Hugo is playing chess against the current chess world champion Magnus Carlsen. In order to not lose, Hugo makes moves by intuitive judgement of the current position and by anticipating possible counterreplies. Since Hugo is familiar with the current position and has played it at home several times, he is quite confident in his decisions.

- Romeo could hardly move his body the first few months after being born. One year later he was running at 1.6 meters per second.

- Jonathan is in a casino and suddenly sees 10 slot machines. He does not know the probability of hitting a jackpot on any of the machines but knows the cost of each lever pull and the reward given by a jackpot. Through repeated selections of slot machines, Jonathan manages to maximize his winnings by concentrating on the best levers.

All of the examples share features that are easily overlooked due to their simplicity. They all involve interaction between a decision-making agent and an environmnet, where the agent is trying to achieve a particular goal without knowing the full extent of the environment nor the consequences of its actions. The agent's actions are able to affect the environment (e.g., Hugo making a move against Magnus Carlsen changes the state of board) thereby affecting the actions available to the agent in the future. None of the actions, in all of the examples, can be predicted fully and the agent must instead observe the environment and how it changes based on actions in order to learn how to properly behave.

§3.3 Elements of Reinforcement Learning

As previously illustrated there are two main components of *reinforcement learning*, an *agent* and an *environment*. It is one the other hand possible to recognize three more elements of a reinforcement learning system. Namely a policy, a reward signal and a value function.

A *policy* describes an agent's way of deciding actions at given times. It could roughly be defined as a mapping from a state of the environment to actions to be taken at that state.

Further, the *reward signal* determines the goal of the learning agent. After each decision the agent makes, the environment sends a reward signal and the goal is to maximize the total reward after all of the actions have been made. The reward signal thus determine what actions are and bad respectively. This could be compared to pain and pleasure in human beings. When a child falls of its bike it recieves a negative reward signal in the form of pain. On the other hand, maintaining balance on the bike for a certain amount of time sends a positive reward signal in the form of happiness. Since the child, the learning agent, wants to experience joy instead of pain the goal becomes riding a bike without falling off of it.

Lastly, whereas the *reward signal* immediately tells the agent whether an action was good or bad, the *value function* tries to tell whether an action is good or bad in the long run. It could roughly be defined as the expected total reward after a certain action while a reward signal only informs the agent about the immediate reward.

§3.4 Markov Decision Process

A *Markov Decision Process*, MDP, is a more rigorous and mathematical way of describing a seqentual decision-making system. A MDP can formally defined as MDP($S, A, \Upsilon, P, \gamma$) where each parameter is defined as following:

- S is the set of all the possible states the environment take on.
- A is the set of all possible actions that can be taken within the environment.
- Υ is the reward signal returned for each action taken.
- P is a probability transition matrix
- γ is the discount factor.

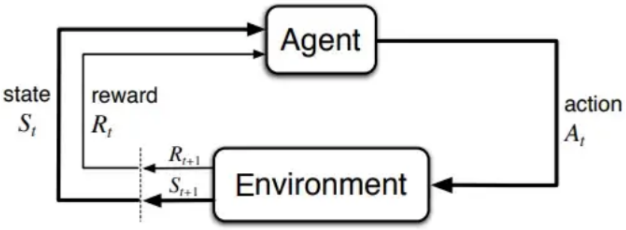

Figure 6: Illustration of Markov Decision Process

§3.5 Q-Learning

§3.6 Double Q-Learning

§3.7 Monte Carlo Tree Search

§3.7.1 Upper Confidence Bound

§4. Deep Learning

§4.1 Introduction

§4.2 Basic Architecture of Neural Networks

§4.2.1 Perceptron

§4.2.2 Fully Connected Neural Network

§4.3 Backpropagation

§4.3.1 Gradient Descent

§4.3.2 Loss Function

§4.4 Complications with Neural Network Training

§4.4.1 Overfitting

§4.4.2 Vanishing and Exploding Gradient

§4.4.3 Difficulties in Convergence

§4.4.4 Local Optima

§4.4.5 Computational Challenges

§4.5 Training Neural Networks

§4.5.1 Backpropagation in Detail

§4.5.2 Preventing Extreme Gradients

§4.5.3 Gradient Descent Strategies

§4.6 Generalizing Neural Networks

§4.6.1 Regularization

§4.6.2 Dropout

§4.7 Convolutional Neural Network

§4.7.1 Historical Background

§4.7.2 Architecture of a Convolutional Network

§4.7.3 Training a Convolutional Network

§4.7.4 Applications of Convolutional Networks

§5. Heuristic Algorithms

§5.1 MiniMax

§5.2 AlphaBeta Pruning

§6. Method

§6.1 Implementation

§7. Results

§8. Conclusions

A. Appendix

A.1 Source Code

https://github.com/chopingu/hajar

References

[1] Huang A Fawzi A, Balog M. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 2022.

[2] F Hsu M Campbell, A.J Hoane. Deep blue. *ScienceDirect*, 2002.

[3] A. Maddison Silver, D. Huang. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

[4] Pritzel A Jumper J., Evans R. Highly accurate protein structure prediction with alphafold. *Nature*, 2021.

[5] Hardesty Larry. Explained: Neural networks. *MIT News Office*, 2017.

[6] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 2018.