人工智能与商业创新作业 2

The AlphaGo -> AlphaGo Zero -> AlphaZero -> MuZero Evolution

姓 名: 孟 念 Muennighoff Niklas
学 号: 1800092850
班 号: 02839210

人工智能与商业创新
(秋季, 2021)

北京大学
光华管理学院
彭一杰老师
高华西助教

2021 年 11 月 17 日

目 录

1 Introduction

This document focuses on the core evolution that has happened in going from AlphaGo to MuZero. This is not meant to be an in-depth analysis of a single iteration, but rather a broad overview of the key ideas that guided this development.

2 Analysis

2.1 AlphaGo

2.1.1 Core Ideas

The AlphaGo model consists of three core parts:

Policy Network This is a Neural Network responsible for predicting the next action a player should take given the current board state and its history. Notably, the board state includes human engineered features, such as how many opponent stones each move would capture.

Value Network: The second, separate neural network, which predicts the likelihood of winning, given the current board state. Like the policy network, this is a CNN with additional human engineered features as input.

MCTS: The third core component is a Monte-Carlo Tree Search algorithm, which randomly simulates games. Together, it works as follows: The model first uses the policy network to select a most promising action based on the current board. The new board state that this action would yield is then evaluated by the Value Network and a "rollout" : Randomly playing the game until the end and storing the winner. These evaluations are combined to yield a probability of evaluating the same new board state again. The model will ultimately choose the move that has been evaluated the most often.

2.1.2 Personal thoughts

I think the combination of Deep Learning (Policy & Value NNs), Reinforcement Learning (Training of the NNs using the reward of winning / losing) and Tree-based algorithms (MCTS) is very fascinating.

Looking at this paper in isolation, there are two suggestions that I have:

- The Policy NN and the Value NN, do very similar things:

  - They both have the same input features
  - Both follow a CNN architecture
  - Both have the purpose of evaluating the board –Only their outputs differ (Probability of each action winning vs overall probability of winning)

  A promising direction would be to combine these two networks to save parameters and maybe even boost performance, as their knowledge could have synergies.

- Another interesting direction would be trying to get rid of the human-engineered features. Since we humans come up with these just by looking at the board, the machine should be able to do the same. I would hypothesize that just using a large model trained longer without these features should be able to reach the same performance.

2.2 AlphaGo Zero

2.2.1 Core Ideas

In the second iteration of the algorithm, they make four key changes to the AlphaGo algorithm:

- Use one NN for both policy and value predictions
- Remove human-engineered features as inputs to the NN
- Do not use full rollouts in the MCTS algorithm
- Only use self-play (The previous AlphaGo model partly used supervised Go training data in training its policy network –In this iteration it is only trained via self-play)

2.2.2 Personal thoughts

I think they worked on the exact right problems and managed the solve them. In addition to the two problems I identified with AlphaGo, they tackled two additional ones: The rollouts, which take a lot of compute as a full game play needs to be simulated until the end; The supervision, which is expensive and hard to collect data

Figure 4 from the AlphaGo paper shows an evaluation of the rollout, Value and Policy network in evaluation and clearly showed that the rollout seems to contribute the least to the model’ s performance. This might have built their intuition that improvements in the value and policy networks could substitute for removing the rollouts.

2.3 AlphaZero

2.3.1 Core Ideas

As the name suggests, AlphaZero is about removing the “Go” from the algorithm. They get rid of several custom “Go” specifics, such as: 1. They generalize the value network output to win / draw / loss, to accommodate for games like chess 2. They remove board augmentations that could be done based on the symmetry of Go. Also the model now plays against its current own weights, rather than a previous state of its weights.

2.3.2 Personal thoughts

Beating the world champions and reaching superhuman performance, it seems they had somewhat maxed out on Go. Trying to take the algorithmic idea and apply it to a broader context, i.e. multiple games, seems like the logical next step.

2.4 MuZero

2.4.1 Core Ideas

MuZero is the second iteration on making the algorithm more general by adding visually challenging Atari games. Key differences from AlphaZero are 1. Not knowing the game rules but learning them 2. Splitting up the Neural Network into a representation, dynamics & prediction model. 3. Generalizing it to multi-agent games.

3 Final thoughts

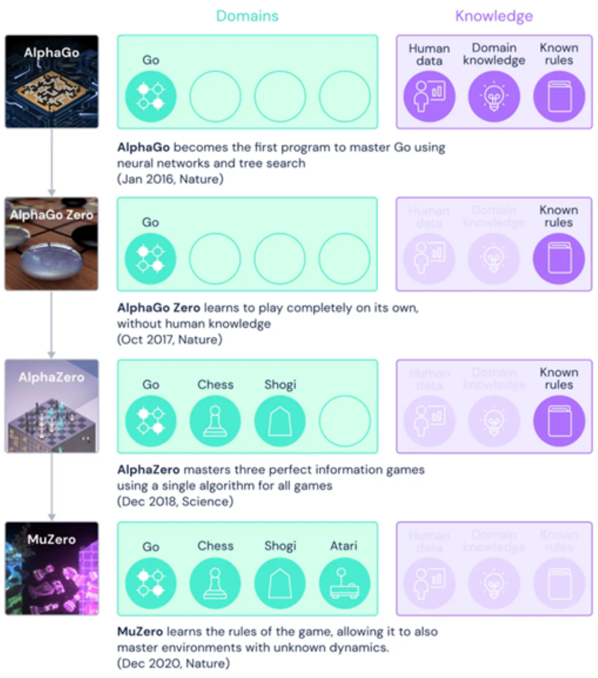
On a high-level, we can describe the four iterations as the first two being focused on performance and the latter two on generalization to other domains. The graph at the end of this paper from DeepMind illustrates this progress. **What' s next?**

To me, there are two logical next steps to work on:

a) **Keep adding domains.** Within the context of games, I would be working on combining MuZero with AlphaStar, DeepMind' s world champion StarCraft agent. StarCraft is more complex than the existing domains AlphaGo masters. Therefore, a MuZero mastering these five games would be a significant step forward. Outside of the context of games, there are a myriad of domains one could try to solve with MuZero or an evolution thereof. Maybe even Protein folding...

b) **Reuse the same model.** Currently, MuZero is retrained for each game. Just like the separate Policy & Value Network in AlpaGo, it seems like there is a lot of overlapping knowledge between a MuZero trained on Chess & one on Shogi. Reusing the same model should save total parameters and compute in contrast to having separate models for each game.

Humans use the same brain to master multiple domains. Reusing the same model for the 4 domains and then working on generalizing it to one new domain after another seems like a straight way towards human-level intelligence.


Domains
Knowledge
AlphaGo
Go
Human data
Domain knowledge
Known rules
AlphaGo becomes the first program to master Go using neural networks and tree search
(Jan 2016, Nature)
AlphaGo Zero
Go
Human data
Domain knowledge
Known rules
AlphaGo Zero learns to play completely on its own, without human knowledge
(Oct 2017, Nature)
AlphaZero
Go
Chess
Shogi
Human data
Domain knowledge
Known rules
AlphaZero masters three perfect information games using a single algorithm for all games
(Dec 2018, Science)
MuZero
Go
Chess
Shogi
Atari
Human data
Domain knowledge
Known rules
MuZero learns the rules of the game, allowing it to also master environments with unknown dynamics.
(Dec 2020, Nature)