AlphaGo paper review

Stanislav Levental

February 20, 2017

Abstract

This review covers basic ideas and techniques introduces by DeepMind team in their paper *Mastering the game of Go with deep neural networks and tree search [1]*

1 Introduction

Go was the last game with open information that where human player was unbeaten, one of the reasons was that Go [2] has huge amount of possible states b^d, where $b \approx 250$ and $d \approx 150$. That give us number of possible states much larger than for example in chess (where $b \approx 35$ and $d \approx 80$). However DeepMind managed to create state-of-the-art Go-paying AI using mixture of Monte Carlo Tree Search algorithm with deep convolutional neural networks.

1.1 Monte Carlo Tree Search

One of the main part in AlphaGo's agent is a *Monte Carlo Tree Search*[3] which is commonly used heuristic search algorithm and was used previously as main part of Go playing AIs. MCTS is using simulation as a evaluation step, so it doesn't need specific heuristic evaluation function, which is usually very hard to figure out. Eventually, MCTS converges to minimax tree [4] when number of evaluations goes to infinity, this mean that MCTS converges to optimal strategy, but that's happening extremely slow, so it's important to choose better moves to explore and then propagate result back. And like Iterative deepening Min-Max Trees MCTS could be interrupted at any step and intermediate result could be used. However, choosing the best move to explore is a tricky thing, one way is to simply choose randomly using unified probability distribution. But previous research shown that using some local features may lead to faster converging to optimal move. In case of AlphaGo this role is played by deep convolutional policy network, which performed very well for predicting human moves. Another issue of MCTS that DeepMind team was able to solve is that exploration/simulation of every tree node might be very costly, so using another neural network for prediction of game outcome based on the game's state was added. Eventually, according to MCTS algorithm - the node that was visited the most number of times was chosen as a next move.

1.2 Convolutional neural networks

One of the main differences from previous attempts of beating human Go player was that DeepMind hardly used deep neural networks with convolutional layers to build their classifiers, they were called policy and value networks. Policy network was responsible for providing probability distribution over possible actions to choose the most optimal one, value network analyzed the whole board and provided probability of winning given current state. DeepMind team used 13 fully connected layers with convolutional features as inputs. Convolutional layers allowed to generate useful features from board's state and use them in neural network's learning. Police network was built, at first, using corpus of played Go games and then improved using reinforcement learning technique by letting network play with randomly selected AI opponent. Value network was built using data generated by artificial games played by policy network with itself, as a result value network could predict good moves 15000 times faster than policy network with almost the same performance.

1.3 Conclusion

Search powered by reinforcement learning mixed with a deep convolutional neural networks are currently state-of-the-art approach for most of the game playing agents. They are allowing to catch different aspects/features of the game which could be found only after very deep and detailed analysis.

References

[1] Silver, David and Huang, Aja and Maddison, Chris J. and Guez, Arthur and Sifre, Laurent and van den Driessche, George and Schrittwieser, Julian and Antonoglou, Ioannis and Panneershelvam, Veda and Lanctot, Marc and Dieleman, Sander and Grewe, Dominik and Nham, John and Kalchbrenner, Nal and Sutskever, Ilya and Lillicrap, Timothy and Leach,

Madeleine and Kavukcuoglu, Koray and Graepel, Thore and Hassabis, Demis *Mastering the game of Go with deep neural networks and tree search*

[2] Go game on wiki. `https://en.wikipedia.org/wiki/Go_(game)`

[3] Monte Carlo Tree Search. `https://en.wikipedia.org/wiki/Monte_Carlo_tree_search`

[4] Bouzy, Bruno. "Old-fashioned Computer Go vs Monte-Carlo Go". IEEE Symposium on Computational Intelligence and Games, April 15, 2007, Hilton Hawaiian Village, Honolulu, Hawaii

[5] Convolutional Neural Networks. `http://cs231n.github.io/convolutional-networks`