Research Report - AlphaGo

- Atul Acharya

In 2016, DeepMind, a research division of Google, announced AlphaGo [1], a game-playing AI agent for the board game Go. The game of Go has long been viewed as the most challenging of classic games for AI to solve, owing to its enormous search space and the difficulty of evaluating board positions and moves. Go, like the game Isolation, is a finite, two-player game of perfect information; however, Go has very high branching factor leading to enormous search space. Until recently, most AI experts believed a computer win for Go was still about a decade ago; this is why AlphaGo's spectacular performance in 2016 and 2017 against human experts is milestone event in AI.

AlphaGo introduces a new approach to playing Go: it uses ***value networks*** to evaluate board positions, and ***policy networks*** to select moves. These deep neural networks are trained using *supervised learning* from human-expert games, and *reinforcement learning* from games at self-play. Using this combination, AlphaGo defeated the current human champions Lee Sedol and Ke Jie [2].

At its core, AlphaGo performs actions similar to an agent playing Isolation. It searches through the game tree and estimates the value of the most likely end states, assuming that an adversary will seek to maximize its reward, and minimize the AI agent's reward. However, a minimax search is intractable for Go; with a branching factor b of ~250 and depth d of ~150, Go is significantly more complex than chess ($b \sim 35$, $d \sim 80$). But AlphaGo introduces a Monte Carlo Tree Search (MTCS), a statistical search technique based on repeated simulations, that achieves greater depth along promising lines of play than minimax alone can provide.

To guide its search, AlphaGo uses deep convolutional neural networks (deep CNN) that analyze the 19x19 game board images. The deep CNN consists of convolutional layers and nonlinear activation functions leading to a final softmax layer that outputs probabilities of a value network. This reduces the effective depth and breadth of the search tree to evaluate positions using the value network, and sample actions using a policy network.

A policy network is trained from human experts using supervised learning (SL) techniques. This provides fast, efficient learning updates with immediate feedback and high-quality gradients. Along with this, a fast policy is trained that can rapidly sample actions from rollouts. Next, a different policy network is trained using reinforcement learning (RL) techniques, that improves the SL network by optimizing the final outcome of games at self-play against thousands of simulated games. This adjusts the policy towards the correct goal of winning games, rather than maximizing prediction accuracy. Finally, AlphaGo trains a value network that predicts winners by playing *against itself*.

Performing these training and evaluating tasks does not come cheap. In fact, AlphaGo requires large and advanced hardware to train: a distributed version of AlphaGo used 1,202 CPUs along with 176 GPUs to train, using an asynchronous multi-threaded search executing simultaneously[1]. The final version of AlphaGo used 40 search threads, 48 CPUs, and 8 GPUs [1].

By combining policy and value networks with MTCS, AlphaGo ends up with a human level performance, beating the 99.8% of games, and human experts like Lee Sedol and Ke Jie. By playing against humans and re-learning its own techniques, AlphaGo gets better with age. The 2017 games were described as “God-level” by Ke Jie [2], who added “AlphaGo is improving too fast; it is like a different player this year than last year.”

References:

[1] *“Mastering the game of Go with deep neural networks and tree search”* - https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf
[2] *“Google’s AlphaGo Defeats Chinese Go Master in Win for A.I.”* - https://www.nytimes.com/2017/05/23/business/google-deepmind-alphago-go-champion-defeat.html