Research Review

Mastering the game of Go with deep neural networks and tree search (link)

Introduction

The AlphaGo Research paper presents the technical background behind Alpha Go, a computer program, a system of algorithms, that were able to beat a human at the game of Go for the first time.

Goals and Techniques

The goal of the paper is to efficiently describe all the components of the AlphaGo system and how it performed against both other Go programs but also, most importantly, against human professional Go players.

Alpha Go used a combination of techniques in order to achieve its results. As a high overview, it used a supervised learning policy network and a fast policy rollout that were trained from human expert games. Once the policy network was trained, it was used to initialize a reinforced learning network and improve it by playing against itself. This would result in a value network that would be then trained by regression to predict the outcome of a board position. When a board position would be fed into the policy network (which constituted of convolutional layers of both supervised learning and reinforced learning networks), this would return a probability distribution over the legal moves from that board position. On the other hand, the value network would return a scalar value that predicted the outcome of the board position that represented the initial input.

Supervised Learning component

The supervised learning policy network constituted of 13 layers. By itself, it was able to predict expert moves with a success rate of 57%, having all the input features (55.7% with only raw board positions and move history). The other researched Go algorithms were only able to achieve a 44.4% prediction rate. The DeepMind team tried different sizes for the network and observed that a larger network would produce higher accuracy ratings but would be slower, while smaller networks, represented by the rollout policy, would produce faster results but at lower accuracy ratings. Therefore, they used a combination of the two in the final system.

Reinforced Learning component

The reinforced learning network was used to improve the policy network. It played the a new policy network against a random opponent from a pool of previous policy networks. After the improvement, the reinforced learning network won 80% of the games against the supervised learning network. It also achieved a 85%

winning rate against Pachi, the best open-source Go program at that time. Another noted thing was that the reinforced learning approach used no search at all, in comparison to previous results at that time, where supervised learning approached achieved only a 11% winning rate against Pachi.

Reinforced Learning of value networks

The final stage of Alpha Go's system was the position evaluation. This was done by estimating a value function that predicted outcome from a given board position of the game, by using the same policy network for both players. This value function was approximated with a value network. Ideally, the value function needs to be calculated under perfect play, but the DeepMind team estimated it using the strongest policy network.

One issue that they had to overcome was overfitting. This behavior was caused by predicting game outcomes directly from data of complete games. The issue with this was represented by successive positions that are strongly correlated. In order to mitigate overfitting, the team generated more games from self-play (AlphaGo playing different versions of itself) with 30 million distinct positions. By training on this new set of positions and game outcomes, they were able to minimize the Mean Squared Error between the estimated outcome and the real outcome to 0.226. Moreover, this minimized the overfitting.

Computations

AlphaGo was developed as both an asynchronous system and a distributed one. It performed the search (through Monte Carlo Search Trees) on CPUs and computed the policy and value networks, in parallel, on GPUs. The asynchronous version of AlphaGo had 40 search threads, 48 CPUs and 8 GPUs.

Results

AlphaGo was initially tested in a tournament against the best commercial and open-source Go program, most of them based on high-performance Monte Carlo Search Trees algorithms. The single-machine version of Alpha Go achieved a 99.8% winning rate, while the distributed one got 100% wins. Moreover, when tested against a professional human Go player, Alpha Go won 5-0 against the 3-time European Champion Fan Hui. This was the first time a computer program was able to beat a professional Go player. Another notable achievement is that in this match, AlphaGo searched thousands of times fewer positions than DeepBlue did against Kasparov, in their famous match of chess.