AlphaGo Review

July 25, 2017

Overview

In this review, we will be examining the techniques used in the AlphaGo playing agent and the insights that have arisen from this state of the art research. In a broad view, the AlphaGo team used value networks and policy networks in order to evaluate board positions. The team has proven the ability of AI applications to be applied to complex game that are regarded as too difficult for computers to play at a professional human level.

Scheme of AlphaGo program Evaluation

1. Value - The program evaluates the position using a convolutional NN
2. Supervised Learning - Training deep neural networks with master-level games
3. Reinforcement Learning - Training a self-playing agent with professional games.
4. Monte Carlo Search Trees - Using rollouts to find an outcome of a position

Major Findings

The game og Go has a large searchspace in the order of B^a where $B = 250$ and $a = 150$. This is segnificantly larger than chess with $B = 35$ and $a = 80$, approximately. The AlphaGo team improved upon the preceeding Go-playing programs, which have been using Monte Carlo Search Trees mostly exclusively, by replicating human decision making with a value network and policy network.
The architecture consisted of a convolutional NN for the value evaluation, which examined the 9x9 square board. The policy network picked moves after being trained upon thousands of professional games and self-play using reinforcement learning and supervised learning. With RL, the progam played against itself and was scored based on whether a position led to a win or loss, then using stochastic gradient ascent, paths that led to victory were rewarded. The policy was updated each time and played against the policy preceeding it, which thwarted overfitting. With SL, the program learned from professional games to pick the right move as the pros did and checking the outcome of that move. This was a triky part as assessing the same position with only a few pieces moved led to overfitting, so the team checked very different positions within the same game.
The algorithm the team used had alphabeta pruning at its core. Whenever reaching a leaf node the value network assessed the position and using rollout the policy network evaluated it with a heuristic function. Then, it

propegated back the outcome. This is a simplistic scheme of how it was done, but testifies to the concept the team has relied upon of using two separate networks for decision making.
In terms of hardware, the team used mostly CPU's with a few GPU's for the regular version. They then created a distributed version which used a staggering amount of computation power that won all games against all of the competing Go-playing programs.
The team showed an approach which they deem as similar to human decisoin-making processes which invloves separate evaluation and move-picking strategy. In the decision process, the next move is calculated with MCST and heuristic evaluation and based on reinforcement learning. This combination creates a very accurate evaluation of a game position with little computation, as we are relying on RL and using rollouts for evaluation.
Overall, the team came up with a novel approach that incorporated deep neural networks and MCST in order to assess complex game positions with minmal computation expenses.