Summary of "Mastering the game of Go with deep neural networks and tree search"

Goals and Techniques

The goal of this paper is to introduce a new AI technique for playing Go. Previously, Go was thought to be the most challenging game for artificial intelligence due to its immense search space: $b \approx 250$, $d \approx 150$ compared to 35 and 80, respectively, for chess, making exhaustive search impossible. In addition, value functions that evaluated the board state at sub-maximum depths were believed to be intractable. The strongest Go programs prior to AlphaGo used Monte Carlo tree search (MCTS) and Monte Carlo rollouts to reduce the search space by using a policy function to sample long sequences of moves without branching to search to maximum depth. These programs achieved the level of a weak amateur player.

AlphaGo uses neural networks to create value functions and policy functions that effectively reduce the depth and breadth of the search tree. First, a 13-layer supervised learning (SL) policy network was trained using expert human moves from 30 million Go positions. In addition, a fast policy function using Monte Carlo rollouts. Then, a reinforcement learning (RL) policy network was trained by playing games against randomly selected previous iterations of the previous policy network to prevent overfitting to the current policy. Finally, the value network was trained using reinforcement learning to predict the winner of games played by the RL policy network against itself. To minimize overfitting, a new self-play data set consisting of 30 million distinct positions was generated.

The policy and value networks were combined in an MCTS algorithm to select actions by lookahead search. Each edge of the search tree has an action value, visit count, and prior probability, and the tree is traversed by simulation. At the end of the simulation, the action value and visit counts of traversed edges are updated, and the algorithm chooses the most visited move from the search position.

Results

The initial SL policy network predicted expert moves with an accuracy of 57.0% using all input features, compared to 44.4% for the previous state-of-the-art. The fast rollout function achieved 24.2% is 2μs compared to 3ms for the policy network. Without using search, the RL policy network won 85% of games against the strongest open-source Go program, compared to only 11% for the previous-best SL network. Interestingly, the SL policy network performed better than the stronger RL policy network, possibly because the RL optimizes for the best possible move, while humans select from a range of promising moves. Nevertheless, the value function derived from the RL policy network performed better than that derived from the SL policy network.

AlphaGo was evaluated against several other Go programs (all based on high-performance MCTS algorithms) and won 494 out of 495 (99.8%) of games. AlphaGo also won the vast majority of games when playing with a handicap. Finally, AlphaGo was evaluated against a professional European Go player, winning all five games. This was the first time a computer program defeated a human professional player. In doing so, the policy and value networks allowed AlphaGo to evaluate thousands of times few positions than Deep Blue did when it defeated Kasparov in chess. In summary, AlphaGo successfully combined Monte Carlo rollouts and deep neural networks trained using supervised and reinforcement learning to conquer one of AI's "grand challenges."