AlphaGo Research Review

Jason Mancuso

March 2017

I decided to read DeepMind's Nature paper on AlphaGo. I'll forgo the common introduction, assuming that most of those interested in game-playing AI will have heard of AlphaGo. This review will cover the objective, methods, and results of the paper.

Objective

AlphaGo is an agent designed to play Go competitively. Although many agents have been made with similar purpose, the state of the art previous to AlphaGo was "strong amateur play" using Monte Carlo tree search along with basic parametric policies and/or value functions. One objective for AlphaGo was to leverage deep learning to outperform this state of the art. The ultimate objective of the research was to create an agent that can compete with professional Go players.

Methods

AlphaGo uses Monte Carlo tree search as its main search algorithm. Monte Carlo tree search is similar to depth-first search in that its first step is to traverse a branch down to a leaf node. The selection process differs widely by application, but AlphaGo chooses branches with maximum action value plus a bonus, which is a function of a stored prior probability associated with that action.

Heuristic search algorithms have had limited success for Go agents due to the game's massive branching and depth factors (≈ 250 and ≈ 150, respectively). Progress had been made by including policy and value functions to shorten the tree search. A policy is a probability distribution over potential actions, corresponding to the probability that an agent will take a certain action according to the current game state. A value function is a measure of long-term reward given a state. Using a policy and a value function, one can construct a Markov decision process and turn the problem into a reinforcement learning problem by engaging in and learning from self-play. Using a Markov decision process is mathematically convenient, as for any finite MDP there will always be at least one optimal policy. Additionally, using a value function to predict the value of each branch instead of traversing the full depth as we would in minimax makes tree search more computationally affordable.

In general, there is no closed-form solution for computing an optimal policy, since its Bellman equation is non-linear. For this reason, DeepMind uses a deep convolutional neural network to approximate the optimal policy. The network is trained via supervised learning on a dataset of master-level games, then trained further through reinforcement learning. Once a branch of the game tree is traversed to the leaf node, if the node is not terminal, it is expanded into possible actions according to the policy network. The policy network outputs probabilities, which are stored as prior probabilities for the associated actions.

AlphaGo uses two other neural networks at this point. The outputs from a value network and a fast-rollout policy network are combined via a mixing parameter $\lambda \in (0, 1)$ to evaluate the quality of the leaf node. The value network is trained through reinforcement learning and evaluates the leaf node based on the game state at that node. The fast rollout policy network plays out a random rollout from the leaf node's game state and yields the outcome. The action state values of the branch leading to the leaf node are then updated based on these evaluations.

Results

To evaluate AlphaGo's performance, an internal tournament was run against leading Go programs based on high-performance MCTS algorithms, both open-source and commercial. AlphaGo won 494 out of 495 of its matches ($p = 99.8\%$) against these programs. To further evaluate AlphaGo, 4-stone handicap matches were played against Crazy Stone, Zen, and Pachi, where AlphaGo won 77%, 86%, and 99% of matches, respectively. The distributed version performed even better, winning 77% of its matches against single-machine AlphaGo. Finally, the distributed AlphaGo played a five-match series against Fan Hui, a professional 2 dan player and winner of the European Go championships in 2013, 2014, and 2015. AlphaGo beat Fan Hui five games to zero, becoming the first computer program to beat a Go professional without handicap. The authors note that this event was previously believed to be at least a decade away.