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

Daniele Pestilli

This article, written by the DeepMind Team, explains how a Go playing program was developed based on a combination of deep neural networks and tree search. This AI consistently played at a higher level than some of the best human players, achieving one of artificial intelligence's greatest challenges and allowing for the possibility to use these new AI techniques in other ostensibly intractable fields.

The AlphaGo approach uses "value networks" to evaluate board positions and "policy networks" to select optimal moves. The first step involves using what is called a Monte Carlo Tree Search (MCTS) which estimates the value of each state in a search tree. This tree grows larger as the number of simulations increases, and the relevant values to determine winning moves become progressively more accurate. The depth of the search tree can be reduced by position evaluation, truncating the search tree and replacing the subtree below by an approximate value function that predicts the outcome from a given board state.

While the search tree recursively computes an optimal value, the policy value allows for the search tree to disregard its exhaustive 250^{150} potential moves. This works by finding the probability distribution over possible moves given a certain position.

These techniques are combined with a 3-step Machine Learning pipeline: the first stage takes into account expert moves in the game of Go using supervised learning. This machine learning agent uses a 13-layer policy network that draws from 30 million positions, yielding an accuracy of 57.0% using all input features, and 55.7% using only raw board position and move history as inputs.
The second stage improves the policy network via reinforcement learning. The third stage of this training pipeline performs a position evaluation, estimating the value of positions based on games played by a given policy for both players.

These combined techniques allowed AlphaGo to achieve a 99.8% winning rate against other Go programs (CrazyStone, Zen, Pachi, Fuego) as well as to defeat the human European Go champion by 5 games to 0.

To achieve these remarkable results in gameplay, AlphaGo used an astonishing 40 search threads, 48 CPUs, and 8 GPUs. A distributed version that exploited multiple machines, 40 search threads, 1,202 CPUs and 176 GPUs was also created.

As a result, AlphaGo has succeeded in one of AI's major challenges: that of yielding consistently good results against human and computer opponents in a decision-making task given an intractable search space, which was previously thought to be infeasible using the latest policy and/or value functions.