Research Review: Mastering the game of Go with deep neural networks and tree search.

Daniel Legorreta

July 25, 2017

The article " Mastering the game of Go with deep neural networks and tree search" presents the techniques used by the AlphaGo system. It is a system game Go that achieved a 99.8% winning rate against other Go programs,and defeated the human European Go champion by 5 games to 0.

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing it to its enormous search space and the difficulty of evaluating board positions and moves. The research proposed uses "value networks" to evaluate board positions and "policy networks" to select moves, and new search algorithms that combine Monte Carlo simulation with value and policy networks.

All games of perfect information have an optimal value function, $v^*(s)$, which determines the outcome of the game, from every board position or state s, under perfect play by all players. These games may be solved by recursively computing the optimal value function in a search tree containing approximately b^d possible sequences of moves, where b is the game's breadth (number of legal moves per position) and d is its depth (game length),but the effective search space can be reduced by two general principle. The depth of the search may be reduced by position evaluation and the breadth of the search may be reduced by sampling actions from a policy $p(a|s)$ that is a probability distribution over possible moves a in position s.

In order to use the techniques of Deep Learning for this problem, they pass in the board position as a 19×19 image and use convolutional layers to construct a representation of the position. They use this network to implement two strategies as the classics, reduce the effective depth and breadth of the search tree. That two strategies are replaced for value network to evaluate position and policy network for sampling actions. Finally, they efficiently combine the Policy and value networks used in Monte Carlo Tree Saerch (MCTS). Therefore this combination of techniques implies that AlphaGo can be seen as: $AlphaGo = Deep\ Learning + Reinforcement\ Learning + MCTS$.

Their neural network was trained using a pipeline consisting of several stages of machine learning. In summary, in the first stage they used supervised learning (SL) of policy networks, in the second stage they used Reinforcement learning (RL) of policy networks and in the final stage they used reinforcement learning

of value networks.

For the first stage the input s to the policy network is a simple representation of the board state. The SL policy network $p_\sigma(a|s)$ alternates between covolutional layers with weights σ, and rectifier nonlinearities. A final softmax layer outputs a probability distribution over all legal moves a. The second stage of the training pipeline aims at improving the policy network by policy gradient RL. The RL policy network p_ρ is identical in structure to the SL policy network,and its weights ρ are initialized to the same values $\rho = \sigma$. The final stage of the training pipeline fuses on position evaluation, estimating a value function $v^p(s)$ that predicts the outcome from position s of games played by using policy p for both players. This neural network has a similar architecture to the policy network, but outputs a single prediction instead of a probability distribution.

Finally, they combine the policy and value networks in an MCTS algorithm that selects actions by lookahead search. Evaluating policy and value networks requires several orders of magnitude more computation than traditional search heuristics. To efficiently combine MCTS with deep neural networks, AlphaGo uses an asynchronous multi-threaded search that executes simulations on CPUs, and computes policy and value networks in parallel on GPUs. The final version of AlphaGo used 40 serch threads, 48 CPUs, and 8 GPUs.

The technique in this research introduced a new search algorithms that successfully combines neural network evaluations with Monte Carlo rollouts. This allowed to solve one of the games that was a challenge for the artificial intelligence for many years.