Build a Game-Playing Agent: Research Review
Mohan Murugesan

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

Introduction

AlphaGo program is viewed as a most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. These techniques are based on the use of deep neural networks which made the "AlphaGo" program feasible in practice. AlphaGo program has introduces the method for evaluating board positions using 'value network' and selecting moves using 'policy network'.

Go 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 250 and it is the game's breadth (number of legal moves per position) and d is 150 and it is depth (game length) and exhaustive search is infeasible.

However the effective search space can be reduced by two general principles.

i) The depth of the search may be reduced by position evaluation: truncating the search tree at state s and replacing the subtree below s by an approximate value function $v(s)\approx v^*(s)$ that predicts the outcome from state s.

ii) 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 (Monte Carlo rollouts search to maximum depth without branching at all, by sampling long sequences of actions for both players from a policy p)

Go uses supervised learning (SL) policy network $p\sigma$, which is learning directly from expert human moves. This provides fast, efficient learning updates with immediate feedback and high-quality gradients. It is also using fast policy $p\pi$ that can rapidly sample actions during rollouts. Reinforcement learning (RL) policy network $p\rho$ has also been used to improve the SL policy network by optimizing the final outcome of games of selfplay. This adjusts the policy towards the correct goal of winning games, rather than maximizing predictive accuracy. Finally, a value network $v\theta$ has been used that predicts the winner of games played by the RL policy network against itself. AlphaGo program efficiently combines the policy and value networks with MCTS.

Results:

To evaluate AlphaGo, internal tournament has been conducted among variants of AlphaGo and several other Go programs, including the strongest commercial programs Crazy Stone and Zen, and the strongest open source programs Pachi and Fuego.

All of these programs are based on high-performance MCTS algorithms. Further open source program GnuGo has been included, a Go program using state-of-the-art search methods that preceded MCTS. All programs were allowed 5 s of computation time per move.

The results of the tournament suggest that single machine AlphaGo is many dan ranks stronger than any previous Go program, winning 494 out of 495 games (99.8%) against other Go programs. AlphaGo also won 77%, 86%, and 99% of handicap games(with four handicap stones that is, free moves for the opponent) against Crazy Stone, Zen and Pachi, respectively. The distributed version of AlphaGo was significantly stronger, winning 77% of games against single-machine AlphaGo and 100% of its games against other programs.

During the match against Fan Hui, AlphaGo evaluated thousands of times fewer positions than Deep Blue did in its chess match against Kasparov; compensating by selecting those positions more intelligently, using the policy network, and evaluating them more precisely, using the value network—an approach that is perhaps closer to how humans play.