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

Parth Pankaj Tiwary

1. A BRIEF SUMMARY OF THE PAPER'S GOALS OR TECHNIQUES INTRODUCED

This paper discusses a new approach to the game of Go, and introduces methods for evaluating board positions using 'value network', and selecting moves using 'policy network'. Creating game playing agent for Go has been challenging in the past, and to develop a game playing which can beat the Go world champion has been all the more challenging, largely because of the fact that the number of nodes in the game tree of Go can be b^d, which given the branching factor b=150 and depth of the game tree d=150 is numerous, and exhaustively searching the tree is infeasible.

Searching such enormous search space in game playing is not feasible, this proposition requires us to reduce the search space, and this paper briefly describes the two methods for doing so:

a) The depth of the search may be reduced by position evaluation: truncating the search tree at state s and replacing the sub-tree below s by an approximate value function v(s)=v*(s) that predicts the outcome from state s.

b) 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 roll outs search to maximum depth without branching at all, by sampling long sequences of actions for both players from a policy p.)

1.1 Architecture employed for the game agent

The board position for each game state is represented as a 19x19 pixels image, and is further fed into the convolution network to construct the representation of the position. The search space is reduced using a 'value network' for evaluating positions and 'policy network' is used for sampling moves.

1.2 Training policy and value network

a) A policy network is trained, using supervised learning directly on the human experts move. This as the paper summarizes, provides fast efficient learning updates with immediate feedback and high quality gradients.

b) The policy network is further trained using reinforcement learning which optimizes on the previous trained policy network, and directs it to adjust well to the winning goal of the game playing, rather than maximizing predictive accuracy which it picks up from supervised learning trained policy network.

c) Finally a reinforcement learning value network is trained which predicts the winner of games player by reinforcement learning policy network.

The game agent AlphaGo combines both the 'policy network' and the 'value network' using Monte Carlo Tree Search.

2. A BRIEF SUMMARY OF THE PAPER'S RESULTS.

2.1 Evaluation of the strength of AlphaGo and results

AlphaGo was evaluated based on the internal tournament between different variants of AlphaGo and Go programs based on high performance MCTS algorithms like Crazy Stone, Zen, Pachi and Fuego. Further open source programs like GnuGo which are based on state-of-the-art methods which preceeded MCTS were also included in the tournament.

The results from the tournament suggest that AlphaGo is many dan ranks above the other Go players which were included in the tournament. The results of the tournament were as follows.

AlphaGo winning 494 out of 495 games (99.8%) against other Go programs. To provide a greater challenge to AlphaGo, we also played games with four handicap stones (that is, free moves for the opponent); AlphaGo won 77%, 86%, and 99% of handicap games 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. AlphaGo further competed against Fan Hui the European Championship winner for 2013, 2014, and 2015, and AlphaGo won the tournament as 5 games to 0. This is the first time a Go playing agent has defeated a human professional player in the game of Go.