SUMMARY

Of

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

The game of Go has been viewed as one of the most challenging games for AI due to its enormous search space and difficulty of search space. In a game like Go whose breadth and depth are around 250&150 respectively, exhaustive search is infeasible. So they've developed a program AlphaGo using two main principles based on Monte Carlo Tree search (MCTS) and deep convolutional neural network which can achieve superhuman performance.

Policy network and value network are trained using deep convolutional neural network for selecting moves and evaluation of position. For a given current board state, Policy network gives us a probability distribution of legal moves for next move selection. There are two kinds of policy networks which are trained.

First is a 13-layer supervised learning (SL) policy network pσ, which is trained from 30 million positions from the KGS Go Server. We also trained a faster but less accurate rollout policy pπ(a|s), using a linear softmax of small pattern features.

Second is reinforcement learning (RL) policy network pρ, which improves from SL policy network by policy gradient reinforcement learning. Since the outcome is expected to be maximized, weight ρ is updated by stochastic gradient ascent in the direction to achieve it. Value network, which focuses on position evaluation, predicts the outcome for a possible position. The current RL policy network is played against itself from a previous iteration selected randomly and is used for value network training in such a way that prevents overfitting by regression. When played head-to-head, the RL policy network won more than 80% of games against the SL policy network.

AlphaGo makes use of these previously created policy and value networks by applying the Monte Carlo Tree search algorithm to them for selecting the actions by lookahead search in game-play. Each edge (s, a) of the search tree stores an action value Q(s, a), visit count N(s, a), and prior probability P(s, a). The simulation starts from the root state and at each time step t of each simulation, an action a_t is selected from state s_t . At the end of simulation, the action values and visit counts of all traversed edges are updated. After the completion of simulation, the most visited move is chosen from the root position.

The final version of AlphaGo used 40 search threads, 48 CPUs, and 8 GPUs. A distributed version of AlphaGo was also implemented that made use of multiple machines, 40 search threads, 1,202 CPUs and 176 GPUs.

For evaluating AlphaGo, an internal tournament was run among variants of AlphaGo and several other Go programs, like commercial programs Crazy Stone and Zen, and the open source programs Pachi and Fuego. And the results have shown that AlphaGo is much stronger than any previous Go program, winning 494 out of 495 games (99.8%) against other Go programs. For a greater challenge, AlphaGo was played against four handicap stones (that is, free moves for the opponent) and AlphaGo won 77%, 86%, and 99% of handicap games against Crazy Stone, Zen and Pachi respectively.

Authors of the Paper : David Silver, Aja Huang, Chris J. Maddison , Arthur Guez , Laurent Sifre , George van den Driessche , Julian Schrittwieser , Ioannis Antonoglou , Veda Panneershelvam , Marc Lanctot , Sander Dieleman , Dominik Grewe , John Nham , Nal Kalchbrenner , Ilya Sutskever , Timothy Lillicrap , Madeleine Leach , Koray Kavukcuoglu , Thore Graepel & Demis Hassabis