RESEARCH REVIEW: MASTERING THE GAME OF GO WITH DEEP NEURAL NETWORKS AND TREE SEARCH

By D. Silver et. Al. Nature 2016

Summary of the paper's goals or techniques

This paper introduces the methods that were used to develop AlphaGo, the first computer program that could successfully win the game of Go against professional human players.

Compared to the isolation game in AIND (which was already intractable) the game of Go is by many orders of magnitude more complex. So far, the best techniques used in solving Go have been using variations of Monte Carlo Tree Search (MCTS).

This work extends substantially on those approaches by using deep convolutional neural networks and reinforcement learning to evaluate the positions of the game and to model the policy that selects actions into an MCTS algorithm.

Using a database of 30 million human expert positions from the KGS Go Server, the authors trained a 13-layer convolutional policy network using supervised learning (SL network p_sigma) to model the policy for selecting moves. The output of this network is a probability distribution over all legal moves.

To improve the policy network, the authors used a gradient reinforcement learning setting where the current policy p_rho played against randomly selected earlier iterations of the policy network, starting from p_sigma.

They also trained a much faster rollout policy p_pi using a linear software of small pattern features. This policy was used for the rollouts to estimate the values of game situations until terminal states.

Based on the 30 million human expert positions the value network v_theta was trained. This network has a similar structure as the policy network, however, it outputs a single scalar

value which is similar to the score value in the isolation game setting. Stochastic gradient descent was used for the training. As successive positions of a single game are strongly correlated, this lead to overfitting. To solve this problem, the authors generated a new dataset from the self-play games and sampled positions from different games.

The policy and value network are then combined into an MCTS algorithm. Starting from the root node, the game tree is traversed by simulation. At each time step the action maximizing the q-value plus a bonus (to force exploration) is selected and expanded. Leaf nodes are evaluated by the value network v_theta, but also by a random rollout using the fast rollout policy p_pi until termination. These evaluations are combined into the leaf evaluation. The edges accumulate the visit count.

At the end of the simulation all values in the tree are updated and the algorithm chooses the most visited action from the root node.

Summary of the paper's results

To evaluate the strength of AlphaGo, the authors ran the software against a range of open source and commercially available Go programs. All of those programs are based on MCTS.

AlphaGo won 494 out of 495 games, i.e. 99.8% against the other programs. When using 4 handicap stones, the winning rates were still larger than 77% against the best opponent. The distributed version of AlphaGo won 77% of the matches against AlphaGo and 100% of the matches against other programs.

As a follow up, the program was also playing against a professional human (2 dan) player and won the match 5 games to 0, making it the first time a computer could beat a professional human Go player.