AIND-Isolation - Research Review

In this research paper, the team at Google Deepmind has built a Go program called AlphaGo, which is a state-of-the-art combination of supervised learning (convolutional neural network), reinforcement learning and Monte Carlo Tree Search (MCTS), and successfully beat the European Go champion Fan Hui in a formal five-game series by 5 to 0, which is the first in history by any Go program to defeat a human professional Go player.

After IBM’s supercomputer Deep Blue defeated the World Chess Champion Garry Kasparov in May 1997, the game of Go has always been referred to as the last frontier of human intelligence. Due to its enormous search space of 250^150 possible moves (*b^d*, where *b* is the breadth of the game tree and *d* is the depth of the game tree), and difficulty in coming up with a sound position evaluation function, it has been a grand challenge to computer scientists to build a Go program which can achieve professional level play.

There are two general principles in effectively reducing the search space (thus achieving better play performance):

1. To reduce the depth of search, we can truncate the search tree at state *s* and replace the subtree below it with an evaluation function *v(s)* which predicts the value.

   For example, for chess, we can assign each piece with a value and have an evaluation function as the sum of remaining pieces at current state. However, for the game of Go, it is not that easy to find a suitable evaluation function because of its game complexity. At one state, the black may seem to be dominating the game board; a few moves in, it can be trapped by the white and all stones got captured.

2. To reduce the breadth of search, we can sample actions from a policy *p(a|s)*, which is a probability distribution of possible moves *a* in state *s*.

   For example, one of the approach is Monte Carlo Tree Search

> (MCTS). The idea is to run many game simulations without branching at all, and store the final outcome (win or loss). At first, the tree search may be totally random. But as the number of simulation increases, we will have a better idea about which path and moves may lead to better results, and it will ultimately converge to optimal play. Before the breakthrough achieved by AlphaGo, all leading Go programs are based on MCTS, which have achieved strong amateur level.

The team at Google Deepmind come up with a novel approach to apply the two principles above. They still use MCTS, but they did not start the tree search from scratch. Instead, they approach the game board as an image classification problem. Essentially, the game board at state *s* is a 19x19 image with black and white stones. By obtaining 30 million game positions from the KGS Go Server, they have trained a 13-layer convolutional neural network (i.e. supervised learning policy network), to predict expert moves, which has achieved a surprisingly high accuracy of 57%. They have also trained a faster but less accurate (24.2%) rollout policy, which just takes 2µs to sample an action instead of 3ms, which is 1,500 times faster, and is particularly useful in creating self-play data later on.

Now that they have a policy network to predict human expert moves, they feed the output of it to a reinforcement learning policy network through self-play. This is needed because after all, the goal is not to achieve the highest accuracy in predicting moves (which can be wrong), but to achieve the highest possible winning rate. By self-playing 1.28 million games with itself, AlphaGo has won 85% of games against Pachi, the strongest open-source Go program.

The final stage is to come up with a sound position evaluation function. Here they have used the same convolutional neural network as in the policy network, except the final layer is changed to a single scalar output, which is the expected winning rate. They have tried to train it on the 30 million game positions which they have obtained before, but this leads to overfitting because the value network seems to memorize the games and fail to generalize to new positions. To solve this problem, they have to train the value network on a new set of data, which is created with the help of the faster rollout policy earlier.

AlphaGo effectively combines the policy network and value network with MCTS, which they called asynchronous policy and value MCTS (APV-MCTS). There are 4 steps in the APV-MCTS when traversing the game tree:
a. Selection: The edge with maximum action value *Q*, plus a bonus *u(P)* which is dependent on the prior probability *P*, is selected.
b. Expansion: If a leaf node is expanded, it is processed by the policy network and prior probabilities P for each action will be stored.
c. Evaluation: at the end of simulation, the node is evaluated in two ways: 1. using the value network; 2. use the fast rollout policy to run to the end of the game and compute the winning rate.
d. Backup: Action values *Q* are updated by the mean value of all evaluations.

The results achieved by AlphaGo are now in history. Not just it has defeated European Go champion Fan Hui (which is 2 dan), but it went on and defeated arguably the best human Go players Lee Sedol (9 dan, 4-1) and Ke Jie (9 dan, 5-0). The one game that Lee Sedol has won against AlphaGo is probably the first and the last, as the team at Google Deepmind has further improved AlphaGo to AlphaGo Master, which has achieved superhuman level play and beat 60 human expert Go players online in a row.