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

Go is a board game with large search space (branching factor ≈ 250, depth ≈ 150), which makes exhaustive search impossible. Existing algorithms can reduce the depth of searching by replacing subtree with an approximation value. In addition, the width of searching can also be reduced by sampling from a policy, which is a probability distribution of all legal moves. However, existing algorithms can only achieve strong amateur play since they are limited to simple policies, and the value function cannot represent nonlinearity within input features. Thus, the goal of this work is to build fast and efficient algorithms to play Go with professional human players.

In this paper, the authors developed an algorithm called AlphaGo, which combined two neural networks and Monte Carlo Tree Search.

The first neural network is called policy network. It takes a 19x19 image of board position as input, and output probability of each legal move. This policy network is trained by two steps.

First, supervised learning (SL) was used to predict expert moves. The policy network is trained on 30 million positions from the KGS Go server. It consists of 13 layers alternating between convolutional layer and rectifier nonlinearity, and output probability with a final softmax layer. Stochastic gradient ascent was used to train the weights in the network to maximize the likelihood of human move. At the end of this step, the policy network predicted expert move with an accuracy of 57.0% with all input features.

Second, reinforced learning (RL) was used to improve the policy network. The weights learned from the previous supervised learning was set to be the initial value. The current policy network was used to play against a randomly selected previous iteration of the policy network. Reward was set to be +1 for winning and -1 for losing. Weights are adjusted by stochastic gradient ascent to maximize the outcome. At the end of this step, the RL policy network won 80% against the SL policy network.

The second neural network is called value network. Its structure is similar with the policy network except that it outputs a single value as state evaluation which estimate the likelihood of a particular move leading to a win eventually. Similar stochastic gradient descent is used to update weights in the network. To avoid overfitting from high correlation between successive positions. A self-play data set was generated from 30 million distinct positions from distinct games. Each game was play between the training RL network and itself until game termination. The results and positions were then used to train the value network.

AlphaGo plays the game by doing Monte Carlo Tree Search with policy and value networks trained in previous steps. The simulation started from the root and traversed the whole tree. The action was selected to maximize a mix of action value, prior probability and visit count, which were updated at the end of each simulation. At the end of search, the algorithm chose the most visited move.

AlphaGo played 99.8% winning rate against other Go programs. It also won 5-0 against a professional 2 dan go player, Fan Hui.