August 7th 2017

How did Computers Master the Game of Go - A Paper Review

By Yanfei Wu

In the paper titled “Mastering the game of Go with deep neural networks and tree search” published in *Nature* in January 2016, Silver *et. al* from Google DeepMind introduced their novel approach to develop a powerful Go program which significantly outperformed other Go programs and also defeated a human professional player in a full-sized game.

The game of Go is a challenging artificial intelligence task because of its enormous search space and super complex optimal solution. Previous efforts to tackle this task were based on Monte Carlo tree search (MCTS) which uses Monte Carlo rollouts to estimate the value of each state in a search tree. But the prior work has been limited to shallow policy or value functions based on a linear combination of input features.

The Google DeepMind team solved this grand challenge of Go by **combining deep neural network with tree search**. They passed in the board position as a 19x19 images and used convolutional layers to construct a representation of the positions. They used these deep neural networks to reduced the effective depth and breadth of the search tree.

The neural networks they constructed contain several stages of machine learning:

- a 13-layer supervised learning (SL) policy network directly from expert human moves for fast and efficient learning updates. With an input of a simple representation of the board state, and trained on randomly sampled state-action pairs using stochastic gradient ascent, the network outputs a probability distribution over all legal moves.
- a reinforcement learning (RL) policy network to improve the SL policy network by optimizing the final outcome of games of self-play towards winning games. It has identical structure to the SL policy network and the same weights as the SL network are initiated. By playing games between the current policy network and a randomly selected previous iteration of the policy network, the weights are updated by stochastic gradient ascent to maximize reward
- a reinforcement learning value network to predict the winner of games played by RL policy network against itself. Unlike naive approach that predicts game outcomes from data consisting of complete games which causes overfitting, the authors generated a new self-play dataset consisting of distinct positions sampled from separate games to mitigate overfitting.

AlphaGo then combines the policy and value networks in an MCTS algorithms that selects actions by lookahead search. Each simulation traverses the tree by selecting the edge with maximum action value plus a bonus that depends on the edge’s stored prior probability. When the traversal reaches a leaf node, the leaf node may be expanded and the new node is processed once by the SL policy network and the output probabilities are stored as prior probabilities for each action. At the end of a simulation, the leaf node is evaluated with both value network and rollouts to compute the winner. Action values are updated to track the mean value of all evaluations n the subtree below that action.

The strength of AlphaGo was evaluated through tournament among variants of AlphaGo and other Go programs based on either MCTS or a search methods preceded MCTS. The results showed that single-machine AlphaGo had a 99.8% winning rate against other Go programs and the distributed-version of AlphaGo was even stronger. The distributed-version of AlphaGo also won a match with a human professional player, which is the first time that a computer Go program has defeated a human professional player, without handicap, in the full game of Go.