Summary: Mastering the game of GO with deep neural networks and tree search

Jaehwan Park

1. Goals or Techniques introduced

AlphaGo efficiently combines policy and value networks with Monte Carlo tree search (MCTS).

MCTS uses Monte Carlo rollouts to estimate the value of each state in a search tree. MCTS has 4 steps: selection, expansion, evaluation, and backup.

<Training pipeline>

1. **Supervised learning(SL) of policy networks**: AlphaGo build on prior work on predicting expert moves in the game of Go using SL with 13 convolution layers. A final output is a probability distribution over all legal moves.
2. **Reinforcement learning(RL) of policy networks**: AlphaGo play games between the current policy network and a randomly selected previous iteration of the policy network. The terminal reward at the end of the game is +1 for winning and -1 for losing.
3. **Reinforcement learning(RL) of value networks**: it focuses on position evaluation, estimating a value function with position (s) and policy (p). AlphaGo estimate the value function for our strongest policy, using the RL policy network. It outputs a single prediction instead of a probability distribution. AlphaGo train the weights of the value network by regression on state-outcome pairs, using SGD to minimize the MSE between the predicted value and the corresponding outcome.

AlphaGo learns rollout policy and SL policy network from human expert positions. From SL policy network initialization, it learns RL policy network. Then, value network is trained from self-data set by RL policy network.

Searching with policy and value networks: MCTS selects actions by lookahead search. Each edge of the search tree stores an actions value, visit count and prior probability. The tree is traversed by simulation, starting from the root state. When the traversal reaches a leaf node at step L, the leaf node may be expanded. The leaf position is processed just once by the SL policy network. The leaf node is evaluated in two ways: first, by the value network; second by the outcome of a random rollout played out until terminal step using the fast rollout policy. At the end of simulation, the action values and visit counts of all traversed edges are updated.

2. Results

AlphaGo defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go.

The final version of AlphaGo used 40 search threads, 48 CPUs, and 8 GPUs. A distributed version of AlphaGo exploited multiple machines, 40 search threads, 1202 CPUs and 176 GPUs.

The single machine AlphaGo has winning rate 99.8% (494 out of 495 games) against other Go programs. A distributed version of AlphaGo was winning 77% of games against single-machine AlphaGo and 100% of its games against other programs.