Research Review on AlphaGo Nature Paper

In a deterministic game with perfect information, such as Chess, Isolation, Tic Tac Toe, Go, etc., one can evaluate the optimal value function v*(s) that determines the outcome of the game from any move, under unlimited time and computational resources. However, some class of games, such as Go and Chess has an exponential explosion of states that disallows us from exhausting the search tree in reasonable time. For example, the game of Go requires around 250 (branching factor / number of possible moves per round) to the power of 150 (depth / number of moves) possible sequences of moves. Due to our computational limit at this time, various approaches have been developed to make better decisions based on the extent that we can search within the given time in game play. For example, one of the key approach is cutting off search and using evaluation function(s). However, the game of Go has eluded us in extracting meaningful evaluation functions until AlphaGo with its value / policy networks and rollout policy with Monte Carlo Rollouts.

Even with the recent advances in computational capacities, we have yet to beat the professional Go players with the best expansive search tree approach, Monte Carlo Tree Search (MCTS). Thus, AlphaGo developed an algorithm that combines policy networks (deep neural networks) and rollout policy within a MCTS. So, instead of using a traditional search heuristic function in the leaf nodes of a Monte Carlo simulation, AlphaGo used a combination of policy network and a quick Monte Carlo Rollout policy network + value function for a better evaluation function.

The heuristic function used by AlphaGo incorporates the supervised learning policy network divided by the number of visits to that node and the value network + the rollout policy network.

Explanation of each components

Similar to the policy network, the weights π of the Monte Carlo rollout policy are trained from 8 million positions from human games to maximize log likelihood by stochastic gradient descent.

The supervised learning (SL) policy network is composed with convolutional layers weights σ and rectifier activation function. A final softmax layer outputs a probability distribution over all legal moves a. This is trained with 30 million positions from KGS Go Server. It outputs a probability distribution over all legal moves.

The reinforcement learning (RL) policy network used the outcome of the supervised learning policy network weights and structure, but trained using the goal to win the game, vs. predicting professional players' moves. With policy gradient reinforcement learning, weights are updated using stochastic gradient ascent. It outputs a probability distribution over all legal moves.

The value network has a similar architecture to the policy network, but, with weights θ and outputs a single prediction. It uses the RL weights to start and trains over the gameplay data generated by games played by RL policy network. This prevents from overfitting from the KGS data.

As mentioned above, a big portion of the evaluation function is the combination of value function, based on the value network, and the Monte Carlo rollout policy. However, a single evaluation of the

value function approached similar accuracy of the Monte Carlo rollout with 15,000 times less computation.

Overall, evaluating the policy and value networks as evaluation function requires several orders of magnitude more computation when compared to a traditional search heuristics. So, AlphaGo utilizes the strengths of GPU for policy and value networks and CPU for Monte Carlo Simulation coordinated by an asynchronous multi-threaded program.

The combination of these allowed a much better outcome that can beat all existing MCTS Go programs with a winning rate of 99.8%. Most importantly, it successfully won against a professional Go player 5-0.