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

Introduction :

As we all know the complexity of the Go game, here is an AI agent named computer Go that uses ‘value networks’ to evaluate board positions and ‘policy networks’ to select moves. These deep neural networks are trained via supervised learning from human experts and reinforcement learning by self-playing. By the help of Monte Carlo Simulation with value and policy networks, AlphaGo program has achieved 99.8% winning rate against Go programs and defeated defeated the human European Go champion by 5 games to 0.

Methodology :

The exhaustive basic search tree for Go’s game consists of 250^{150} nodes which is definitely a huge number. However, an effective search can reduce this search space by two principles. Depth-reduction, where at some state s it is predictable to know the approximated value function $v(s) \approx v^*(s)$ that replaces the its whole sub-tree. The other is Breadth-reduction, the breadth of the search may be reduced by sampling actions from a policy $p(a|s)$ that is a probability distribution over possible moves a in position *s*. For example, Monte Carlo rollouts search to maximum depth without branching at all, by sampling long sequences of actions for both players from a policy p. Averaging over such rollouts can provide an effective position evaluation, achieving superhuman performance in backgammon and Scrabble , and weak amateur level play in Go. An estimation for every state *s* the search tree is done by Monte Carlo Tree Search (MCTS) through Monte Carlo rollouts.

In order to reduce the depth and breadth, a deep neural network is used for this purpose. As the well-known the unprecedented performance in visual domains, the board is represents as 19x19 image. To train the network, firstly by Supervised Learning (SL) to train the policy network. Next, by Reinforcement Learning (RL) to improve the pre-trained policy network by optimizing the final outcome of games of self-play.

The policy network is used to output the possible actions (moves) the agent might take at some state of the game. The value network alongside with fast rollout policy network in

Monte Carlo search are used to predict the value of each action. The final version of AlphaGo uses 40 searching threads, 48 CPUs and 8 GPUs.

By the final tournament that was run against open-source Go programs, AlphaGo won 494 games out of 495 (win rate: 99.8%). For providing a greater challenge for AlphaGo, it plays also with four handicaps stones (4 free moves for the opponent) and it shows win accuracy of 77%, 86%, and 99% of handicap games against Crazy Stone, Zen and Pachi, respectively.

There is also a distributed version of AlphaGo that is significantly stronger, winning 77% against single-machine AlphaGo and 100% against other Go programs. Finally, DeepMind team evaluated the distributed version of AlphaGo against Fan Hui, a professional 2 dan, and the winner of the 2013, 2014 and 2015 European Go championships .In 2015, Fan Hui competed in a formal five-game match. AlphaGo won the match 5 games to 0. This is the first time that a computer Go program has defeated a human professional player, without handicap.