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

Go is an elegant and intuitive game of perfect information (no luck involved) during which two opponents take turns to put stones on a board. Once a stone is put on a board it stays there till the very end. However stones can be captured by completely surrounding them. That creates a vast search tree with average branching factor of 250 and average depth of 150. Game's ultimate objective is to control more than 50% of a board.

The paper presents a new approach to building AI agent of Go. Proposed solution achieved 99.8% win rate with other AI opponents and 5-0 victory with the human European Go champion. Researchers combined Monte Carlo tree search that chooses optimal move and convolutional deep neural networks which reduce search space.

There are two types of neural networks involved. Value network which reduces depth of game tree and policy network which reduces its breadth. Policy neural network has 13 layers and was trained from 30 million game positions from professional Go server. Its training was also reinforced via playing against itself (against randomly selected previous network iterations). Value network is used to estimate a value function of a given position. It outputs a single value prediction instead (unlike policy network which outputs probability distribution).

Minimax depth-first search with alphabeta pruning turns out to be insufficient in the game of Go. Alternative approach used in AlphaGo is double approximation. First uses Monte Carlo simulations to estimate value function and second uses regular minimax optimal actions.

Features used during neural networks training are 19 x 19 planes. They originate directly from game rules and contain raw representation of typical game moves such as stone colors, liberties or captures. The training is similar to the one used in image recognition. Inputs to both networks are 19 x 19 x 48 image stacks however value network also receives additional binary feature describing which color to play.

To prevent networks from overfitting a new distinct dataset of 30 million distinct positions was created. Each position was sampled from a different game thus mitigating the problem of simply memorizing a move by neural network rather than generalizing to new positions.

To further optimize searching AlphaGo uses asynchronous multithreading. Simulations are performed on 48 CPUs while policy and value neural networks are run on 8 GPUs. Researchers also implemented a distributed version of AlphaGo with 1202 CPUs and 176 GPUs.

Effectiveness of proposed approach was evaluated by playing tournaments with other available AI agents such as Crazy Stone, Zen, Pachi or Fuego and measuring Elo rating. AlphaGo won 494 out of 495 games. Results were also satisfactory when opponent was given a small handicap. Time constraints for formal games were 1 hour main time plus 30 seconds of *byoyomi*, which is additional time after regular time expires.

Researchers developed a novel way that combines supervised and reinforced learning and introduced a new search algorithm using Monte Carlo rollouts. Combining those three components created a Go agent that plays game at the level of best human players.