RESEARCH REVIEW

Mastering the Game of Go with Deep Neural Networks and Tree Search

Sai Tai

Goal

This research introduces an approach to create a game agent which is capable to play a classic Japanese board game Go. Go has long been viewed as the most challenging of classic games for Artificial Intelligence owing to its massive search space. It contains approximately b^d possible sequences of moves, which ($b\approx250$, $d\approx150$) while other chess ($b\approx35$, $d\approx80$).[1] We needs better techniques when deciding which branch of the game tree to select to defeat a professional Go player.

The authors of this research used 2 type of networks, '**value networks**' to evaluate board positions and '**policy networks**' to select moves. They invented an innovative approach that leverages three neural neural networks in the following order:

Step 1: Supervised Learning policy network - Trains with human expert moves.
This network is a 13-layer deep CNN trained on randomly sampled state-action pairs from 30 million positions from the KGS Go Server. The neural network takes input features from the board position and outputs the probability of each move on the board being the actual next move.

Step 2: Reinforcement Learning policy network - Evaluates self-play outcomes of the current state
This second network produces a probability distribution of moves, and passes this into a randomly selected previous iteration of the policy network which performs a regression to establish the winner of the self-play for 1.2 million times to build 'a stronger itself'.

Step 3: Reinforcement Learning for Value Network - Predicts the winner of games with RL policy network
The final stage of the training pipeline focuses on position evaluation, estimating a value function that predicts the outcome from a position of games played by using policy for both players. This neural network had a similar architecture to the policy network but outputted a single prediction instead of a probability distribution.

Step 4: Search with Policy and Value Network
AlphaGo combined the policy and value networks in an MCTS algorithm which utilizes random sampling of the tree with evaluation of each game tree branch. At each visit evaluations are stored so as to quantify a "bonus" for each branch that effectively determines the most "promising" moves. The value for each explored node is the mean of an evaluation function over the number of visits on that node.

Results

By using the combination of policy and value networks with tree search, the game agent AlphaGo achieved a 99.8% winning rate against other Go programs and defeated the human European Go champion by 5 games to 0. The progress of defeating a human professional player in the full-sized game of Go is ahead by at least a decade.

[1] Game complexity, Wikipedia. https://en.wikipedia.org/wiki/Game_complexity#State-space_complexity