RESEARCH REVIEW

MASTERING THE GAME OF GO WITH DEEP NEURAL NETWORKS AND TREE SEARCH

In all games of perfect information, it is possible to derive a value function which determines the outcome of the game from each board position assuming perfect play by each player. However, deriving this optimal score can be computationally intractable for games like Go or chess.

One way to reduce the search tree is to cut off the search at a certain depth, returning an approximation of the value of a move. Even with techniques like alpha-beta pruning to further limit the number of branches to search, the complexity of Go still made it unfeasible to progress beyond weak amateur play.

If the search tree cannot be fully evaluated, another option is to limit the breadth of the search through sampling. Monte Carlo tree search is a popular technique. Long sequences of game play are simulated by sampling from available moves. The accuracy of the tree is expanded as more and more simulations are run. Over many simulations, the estimates of moves become more accurate. The previous top Go programs attempted to optimize the search by selecting moves to sample from policies designed to predict the moves of expert players.

AlphaGo's major innovation was the introduction of two deep neural networks to generate the policy and value functions used to run the Monte Carlo tree search.

The first network uses supervised learning to create a policy network to select moves. The network is a convolutional network that takes as input the board positions and returns a softmax probability of legal moves. This network was trained with games from expert players. This gives move selection a more human characteristic.

The second network is similar to the first but instead of returning a probability distribution, it returns a value for the move. This value is calculated using reinforcement learning by playing against previous iterations of the network. This provides a more accurate valuation of a move by measuring how likely it was to lead to a win.

With the policy and valuation functions in place, a Monte Carlo tree search is then used to evaluate and select possible moves. Because evaluating a neural network is considerably more computationally expensive than traditional search heuristics, this consists of running many simulations in parallel until time runs out (after about five seconds).

Once a move is selected, the selected move becomes the root of the next simulation and the rest of the tree is discarded. In this way, This allows for stronger subsequent moves by leveraging previously evaluated simulations.

AlphaGo's victory over top ranked human players was impressive. Even more impressively, however, it was able to achieve these victories with fewer computations than Deep Blue used in its chess matches and without specifically encoding the rules of Go in its selection or evaluation process.