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

The article presented serves to provide an outline of the methods and logic behind the development of the AlphaGo model conceived by researchers at Google's DeepMind group. AlphaGo was found to significantly outperform all other computational models previously developed which had been applied to the game of Go, and ultimately achieved a near perfect win rate against both computer and human opponents. The paper outlines a pipeline developed by the researchers which incorporated training via a supervised policy network, a reinforcement learning policy network based on self-play of games, and finally a value network used to evaluate moves chosen. Deep convolutional neural networks were used to conduct the training, with the game being passed in as an image and the convolutional layers used to construct the representation, and Monte Carlo Tree Search(MCTS) employed to maximize computational efficiency in move choice. This allowed for a professional level of play to be achieved by AlphaGo while significantly reducing the computational complexity required during search and evaluation.

The first step in the pipeline consisted of a supervised policy network trained on expert human moves. This takes an input representing board state and applies a stochastic gradient ascent to maximize the likelihood of a human move in the selected state. A 13-layer network was trained based on 30 million positions gleaned from the KGS Go Server. Accuracy of this network alone achieved 50.7% using all inputs and 55.7% using board positions and move history.

The Reinforcement Learning state was identical in structuring to the first step in the pipeline, with a random model selected from previous iterations of the network used as the opponent in gameplay to prevent overfitting of the model. Outcomes are limited to terminal rewards of +1 for wins and -1 for losses, with weights updated at each time step via stochastic gradient ascent to maximize outcome. This model achieved a 80% win rate against the model trained in the first step and a 85% win rate against the previous state of the art Go model Pachi (which uses a MCTS method).

The final step in the pipeline was the estimation of a value function to predict the outcome of a game through the evaluation of the player's current position based on previous games played using the policy network and the outcome from the selected position. The architecture used in this step is similar to those previously outlined with the exception of the output being a single prediction instead of a probability distribution.

The policy and value networks outlined are combined in a novel application of Monte Carlo Tree Search which selects action by look-ahead search. Traversal of the tree is done from the root state, with action values being increased for those actions which are most visited so that only those actions are selected.

Implementation of the pipeline was originally done through an asynchronous multi-threaded search executing simulations on CPUs and computing policy and value networks in parallel on GPUs. This was later expanded to a distributed model with over 20 times the computational capacity which significantly outperformed the asynchronous model when matched against it.