Research Review

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

Summary of the Goals and Techniques Introduced

The goal of this research was to create a game-playing agent that would be competitive against skilled Go players. Go is an intractable problem compared to Chess because of the larger search space. Grandmaster-level Chess players can be defeated by an agent that searches a curated section of the search space utilizing a custom evaluation function. Go programs that utilize similar techniques are nowhere near as successful.

Previous Go game-playing agents had used MCTS (Monte Carlo Tree Search). The researchers' novel techniques were using deep neural networks that were trained with vast amount of data from the KGS Go Server. First, a Policy Network was created to evaluate what moves to take. This Policy Network was trained via Supervised Learning using the data set. This step created a probability distribution of likely moves that were to be explored. Next, the Policy Network was improved using Reinforcement Learning. The Policy Network was made to play against randomly sampled previous iterations of itself.

A separate Value Network was also created to evaluate the score for a player of the current board state. The Value Network also utilized deep neural networks. The researchers sought to avoid overfitting because there was a tendency for the Value Network to focus on the end outcome and not similar situations. To avoid this problem, they made the Value Network train on the games created from Reinforcement Learning instead of the KGS Go data. This network was then made to play against previous iterations of itself that were randomly sampled.

These networks were then used by the MCTS to determine which moves to take. When a particular branch was searched often, that particular branch was downgraded so as to encourage the exploration of other branches.

Summary of the Results

AlphaGo was able to defeat one of the top Go players in the world 5 to 0. Additionally, AlphaGo defeated other Go programs in 99.8% of matches. This result was achieved with a much more abstracted neural network model as opposed to Deep Blue which used tailored evaluation functions. The authors specified that the neural network or the MCTS alone performed worse than the combination of the two.

One negative cited was the computational power required to achieve the results. AlphaGo used 48 CPUs and 8 GPUs and the distributed version of AlphaGo used 1202 CPUs and 176 GPUs. However, despite the massive computational power required, the approach used in which AlphaGo learned to play based on playing games instead of using tailored evaluations functions is a promising development for future AI agents.