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

Main Objective

DeepMind's objective for this project was to develop a game agent for Go to beat any previously developed Go agent or human player. The paper mentions how previous attempts used the Monte Carlo Tree Search (MCTS) to accomplish this. Due to the enormous search size of the game, the previous attempts all had limitations regarding speed/efficiency and accuracy. AlphaGo is an algorithm that is designed for games that have a big breadth and depth space by incorporating deep neural networks to enhance the searching algorithm.

Techniques Used

Go has an insanely large amount search space. The possible sequences of moves for a game is b^d (breath to the power of the depth). In the case of chess, the breath is approximately 35 and the depth is 80. For the case of Go, the breath is 250 and the depth is 150. Since the game is significantly more complex, exhaustive searching on a set that is 250^150 is not feasible. The AlphaGo algorithm incorporates supervised learning from human played expert games and unsupervised learning from self play in conjunction with search to obtain efficiency. The algorithm is composed of three components: The policy network, the value network, and the MCTS algorithm. The policy network is the component utilizes supervised learning by sifting through 30 million positions form KGS Go servers. Reinforcement learning is then applied to the policy network to increase accuracy. The value network is used for positional evaluation. This part utilizes the policy network to generate data for unsupervised learning. This is designed to reduce over fitting. Lastly, the policy function and value functions are used by the MCTS algorithm to perform searching. The search space is much smaller than original and less computations are required. The algorithm looks ahead at each viable branch and counts the edge that is most traversed in the selection process. This combination of neural networks and search proves to have significantly promising results.

Results

Pachi was the leading Go playing agent before AlphaGo. The old algorithm utilized MCTS and could play at a 2 dan amateur level. AlphaGo, with only the supervised and reinforcement learning (with no search), won against Pachi 85% of the time. The value network function when compared to the Monte Carlo fast rollout policy, proved to be far more accurate while requiring 15,000 times less computation. With search, the AlphaGo algorithm is far superior than any electronic or human opponent. Given 5 second limit on each move, AlphaGo was able to win 99.8% of the time against other agents. Furthermore, AlphaGo beat Fan Hui, arguably the best human Go player (and winner of 2013, 2014, and 2015 European Go Championship) 5 games to 0. Along with reducing the search size (policy network) and the computation required to evaluate boards (value network), the final version of AlphaGo utilizes asynchronous multi-threaded search. The program was able to distribute it the searching on 40 threads on 1,202 CPUs and 176 GPUs. Ultimately this project foreshadows a strong link between machine learning and artificial intelligence. AlphaGo was the first algorithm able to win against a human with no handicap. This feat was predicted to be impossible for this decade. Although the algorithm is coupled tightly to just playing Go, there are a plethora of opportunities to solve using this neural-network/search relationship.