RESEARCH REVIEW: ALPHA GO

This paper introduces the world's first game playing agent capable of playing the game of Go at a level exceeding human expertise. This milestone has long been considered by AI experts to be decades away from being reached. The challenge of conquering the game of Go stems from the fact that the game's branching factor (b) and depth (d) are large (approximately 250 and 150 respectively). Recall that a game with a branching factor b and depth d has $O(b^d)$ nodes to explore, which in the case of the game of Go excludes the use of exhaustive search techniques.

To master the game of Go, AlphaGo uniquely combines the techniques of Monte Carlo Tree Search (MCTS) and deep neural networks. In MCTS, one determines how to navigate a game not by exhaustively exploring the tree but by running many simulations of how the game evolves starting from the current state and ending at the game's end. By keeping track of how tree nodes across the simulated games correlate with the game's outcome, the MCTS algorithm enables an agent to choose good moves. But how does AlphaGo perform the game simulation that is the cornerstone of the MCTS algorithm?

In conventional MCTS, game simulation is performed by sampling actions for the agent and the opponent using hand-crafted rules. This has limited previous algorithms that use MCTS for the game of Go to play at an amateur level. In contrast, AlphaGo relies on two types of deep neural networks to evolve the game. One neural network, the value network, determines how likely a game tree node will ultimately result in a win and thus determines the value of a node. The second neural network, the policy network, determines which action to take at each of the game tree nodes. But what makes these neural networks so much better than hand-crafted rules? How were they trained and used in AlphaGo?

To train these neural networks, the AlphaGo team relied on a combination of supervised learning and reinforcement learning. To train the policy network, the team first used supervised learning to teach the network how to predict an expert player's next move using a large annotated dataset. Next, the team employed reinforcement learning and self-play (i.e. two neural networks playing against each other) to ensure that the final policy network selects moves that lead to the game being won (a long-term goal) as opposed to only selecting moves that mimic an expert's next move (a short-term goal). To train the value network, the team once again turned to a self-play data. More specifically, they used self-play data to teach a network to assess the value of a game state with respect to the long-term goal of winning the game.

The AlphaGo agent is a clever mix of classical and modern artificial intelligence and machine-learning techniques. What I find most fascinating about AlphaGo is how the algorithm reached (and surpassed) expert level play through self-play. In other words, by pitting the AlphaGo agent against itself countless times (more times than any human can ever play the game of Go) the agent steadily improved.