A Brief Summary of AlphaGo

As we've seen in our exploration of the very simple game of Isolation on a 7x7 board, using a complete search of the entire search space, roughly 10^43 nodes, to determine the optimal action for a given state, whether employing minimax or apply thing alpha-beta pruning optimization, would take too much time for this to be an acceptable strategy. Expand that complexity to the game of Go on a 19x19 board and the problem falls well into the land of the intractable, 250^150 nodes[1]. Before AlphaGo, the most successful attempts to build an AI agent for the game of Go used a Monte Carol Tree Search (MCTS) to narrow the search space using a beam of high-probability actions. Using shallow policies for action selection and and value functions based on the linear combination of features, these agents were able to achieve strong amateur play.

Researchers at DeepMind, recognizing that deep neural networks were having significant success in complex search spaces in other subfields of artificial intelligence, e.g. computer visualization, began exploring replacing the shallow policies and valued functions commonly used in Go AI agents. They first train a supervised learning policy network (SL policy network) based on expert moves from 30 million positions on the KGS Go Server. This network used stochastic gradient ascent to maximize the likelihood a human would make a particular move, given a board position. The output for this network was a probability distribution of all possible moves. This network was able to accurately predict expert moves 57% of the time, compared with prior state-of-the-art prediction accuracies of 44.4% from other research groups.

While the above policy network was strong, it was still slow, so they also created a simpler, and less accurate fast policy network that could be used to sample actions during Monte Carlo rollouts.

Next, the researchers utilized reinforcement learning to improve on the supervised policy network, creating a new network called the RL policy network. The RL policy network was trained by playing games between the RL policy network and the SL policy network. At this stage the team had a very strong policy network (RL) that even when not using search at all was able to win 80% of games played against the SL network and 85% of games played against the strongest open source Go program, Pachi. The goals wasn't to beat other AI's, though, professional Go champions were!

The research then trained a value network that output a single value, representing outcomes, instead of a probability distribution of likely moves.

Finally, in order to compete against human professional Go players, the DeepMind team brought back the Monte Carol search algorithm, combining their value and policy networks

to choose the best action. Each edge in the search tree stores a Q-value for the action, a count of how many times the action has been visited in simulation, and prior probability. The tree is then traversed by choosing actions that maximize the action value plus a bonus that decays the more times it has been visited (to encourage exploration). When the search reaches a leaf, it is evaluated using the value network trained on the SL policy network as well as the outcome of a random rollout using the fast policy network. These values are then combined into a leaf evaluation value. Finally, after updating all of the action values and visit counts, the agent chooses the action that was most visited during the simulation.

Using this technique—and experimenting with different combinations of networks—the team conducted a tournament against the strongest AI Go programs, winning 99.8% of games. They also played five-game match against Fan Hui, a 2 dan professional, and wind all 5 games —an amazing achievement.

After reading this paper in depth, I'm excited to experiment applying neural networks to my little game of isolation and see if I can improve performance even further.

References

1. D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. Nature, 529 (7587) (2016), pp. 484-489.