Summary of AlphaGo by DeepMind Team

AlphaGo is a artificial intelligence program that was designed and trained with the intent of beating top ranked human players in the game of Go. The game of Go is of particular importance in the AI game playing field due to its enormous branching and depth factor. With simple games, such as tic tac toe, a computer can search all possible moves easily. Approximately 20 years ago AI researchers were able to beat chess champion Gary Kasparov using Deep Blue. However their solution included handcrafted heuristics which aided the program in evaluating game states.

The creators of Alpha Go by comparison were able to train a multilayer artificial intelligence network without using any hand crafted heuristics at all. The team was able to do this by combining various known AI techniques, combining prediction with neural networks, with reinforcement neural networks, with finally a neural network that summarizes a game state into a single value. Previously the state of the art AIs for Go used a technique called Monte Carlo Rolling search. Essentially what this means is that an AI would play a game randomly at every move and then see whether it won or lost. It would then update the branches of the path it took with that information. The AI would then repeat this many more times, weighing and reweighing as it goes. A fundamental decision in this technique is the exploitation vs exploration tradeoff in which the program weights going to a branch it know had a win, versus trying a new branch.

Alpha Go instead trained itself through many layers as mentioned above. AlphaGo first step was to train itself to predict human moves with as much accuracy as possible. This Policy network was trained using 30 million moves from the KGS go server. This neural network was good at predicting the next human move, but not good at predicting whether it would win a game or not. The next step was to train another neural network that would be optimized towards winning games. The Reinforcement Neural Network as labeled by the paper, was initialized as the previous single policy network, but played games against randomized iterations of the policy network. Each time it won or lost the Reinforcement Networks weights would be readjusted.after wins and losses. From this a final network was then trained that took a board position and using the outputs from the Reinforcement Learning Neural network, would output a value of the board game state.

During actual play the AlphaGo engine performs a tree search but ends up combining the results of the SL and RL policy networks. The authors of the paper mention that the SL network performs worse by itself, but performs more exploration, versus the RL network picks the single best move. Independently the moves chosen are not the best but combined produce the moves the AlphaGo engine ends up choosing.