Research review of Mastering Alpha GO

By: Davis Martin

The main goal of this paper is to explain how the Alpha GO program made by Google was able to defeat professional GO player 5 times in 5 games. This is groundbreaking for game playing agents the number of sequences one has to consider grows rapidly `$breadth ^ {depth}`. The Tree for GO is particularly huge (b=250, d=150) because of enormous search space in the game making it computationally impossible to search the entire tree.

Normally the search space is reduced using 2 principles:

1. **Depth -** can be reduced by position evaluation or truncating the tree as a particular state `s` and replace the entire subtree below with a value function predicting the outcome from the state below. This is normally enough for games like chess with b=35 d = 80.
2. **Breadth -** can be reduced by sampling actions from a policy `p(a|s)` probability distribution over possible moves `a` in position `s`. This is really good for games like backgammon and scrabble.

Google went further by utilizing Monte Carlo Tree Search (MCTS) and Neural Nets to break it down even more as well as quickly pick the best mover against their opponent.

MCTS: Use Monte Carlo rollouts to estimate the value of each tree and w/ enough rollouts the value converges to optimal play. This allowed them to narrow the search beam of high probability actions shrinking the tree further.

Supervised Learning (SL) Neural Net: Trained on expert human moves provides fast efficient learning with updates so that they could predict expert moves.

Reinforcement Learning (RL) Neural Net: Plays games between the current policy network and randomly selected previous iterations of the policy network. This randomizes the pool of opponents and plays itself to update the SL net from its self play.

Finally, Their Alpha GO program efficiently combines the policy and value networks as well as the MCTS so be able to defeat professional GO players.