Summary : Mastering the game of Go with neural networks and Tree Search

AlphaGo has taken a different step on the game board problem by integrating two techniques into an intelligent system, these techniques are:

1. Deep convolutional neural networks
    1.1 Policy network
    1.2 Value network
2. Monte Carlo tree search

Policy networks

The policy networks provide guidance regarding which action to choose , given the current state of game. The input is an image of the game state and output is probability value for each possible legal move (i.e. the output of the network is as large as board). Here a 13 layer policy network is used.

Supervised Training : Two policy networks are trained , one is trained on random rollouts and the other network is trained to predict human expert moves. The network is trained from 30 million positions from the KGS GO server.
Reinforcement learning :The reinforcement learning policy network are same in arhitecture. We play games between the current policy network and a randomly selected (to avoid overfitting) previous iteration of the policy network. We award r(S) to all the terminal nodes from the perspective of the given player : +1 for winning and -1 for losing.

Value networks

Value networks take the current game state as imput and output a single numeric value representing the probability of a win. Value network is similar to policy network in architecture.

Training : The value network is trained on a new self play data set consisting of 30 million distinct position search sampled from a seperate game.Each game is played between RL ploicy networks and itself until the game is terminated.

Monte Carlo Tree Search

The focus of Monte Carlo tree search is on the analysis of the most promising moves, expanding the search tree based on random sampling of the search space. The application of Monte Carlo tree search in games
is based on many playouts. In each playout, the game is played out to the very end by selecting moves at random. The final game result of each playout is then used to weight the nodes in the game tree so that better
nodes are more likely to be chosen in future playouts. The most basic way to use playouts is to apply the same number of play-outs after each legal move of the current player, then choosing the move
which led to the most victories. The efficiency of this method—called Pure Monte Carlo Game Search—often increases with time as more playouts are assigned to the moves that have frequently resulted in the player's victory (in previous playouts). Full Monte Carlo tree search employs this principle recursively on many depths of the game tree. Each round of Monte Carlo tree search consists of four steps:

- Selection: start from root R and select successive childnodes down to a leaf node L. The section below says more about a way of choosing child nodes that lets the game tree expand towards most promising moves, which is the essence of Monte Carlo tree search.

- Expansion: unless L ends the game with a win/loss for either player, either create one or more child nodes or choose from them node C.

- Simulation: play a random playout from node C. This step is sometimes also called playout or rollout.

- Backpropagation: use the result of the playout to update information in the nodes on the path from C to R.