AlphaGo Research Paper Summary

AlphaGo is a program combining tree search and deep neural networks that plays at the strongest human players' level thus achieving one of AI's grand challenges. It is the first to develop effective move selection and position evaluation based on neural networks. The networks are trained based on combination of supervised and reinforcement learning. It introduces a new search algorithm that combines MCTS and neural networks evaluations. During gameplay the program evaluates less positions than Deep Blue did at chess, evaluating its select positions more accurately using the value network, an approach perhaps closer to how human play. Furthermore while Deep Blue relied on handcrafted evaluation function the neural networks of AlphaGo are trained directly from gameplay through general purpose machine learning techniques

AlphaGo further improves on the Monte Carlo Tree Search technique used by previous state of the art Go programs. The focus of MCTS is on the analysis of the most promising moves, expanding the search tree based on random sampling, rollouts, which are long sequences of moves traversing the tree all the way to a terminal state. The final game of each such rollout is then used to weight the nodes of the game tree so that better nodes are more likely to be used in the future. An Upper confidence bound method is used to maintain a balance between exploration and exploitation where with each repeated visit to a node its selection probability degrades.

The networks pipeline architecture starts by training a supervised learning (SL) policy network P_σ from expert human moves. This is a 13 layer CNN which outputs a probability distribution of a win over all legal moves. The input to the network is a representation of the board state as 19X19 pixel grid. The network was trained from randomly sampled state-action pairs from a database of 30 million positions.

Another policy network P_π was trained that can rapidly sample actions during rollouts requiring a smaller set of features than P_σ does.

Next was trained a reinforcement learning (RL) network P_ρ that improves the SL policy network by optimizing the final outcomes of games of self-play. This adjusts the policy towards the correct goal of winning a game rather than predicting its outcome.

Finally a value evaluation network V_θ that predicts the outcome of games the RL policy network played against itself. This network has a similar architecture to the policy network but outputs a single prediction instead of a probability distribution.

Monte Carlo rollouts search maximum depth without branching at all by sampling long sequences of actions for both players from a policy function.

AlphaGo combines the policy and value networks in an MCTS that selects actions by lookahead search, each edge of the search tree (s,a) stores an action value Q(s,a), Visit count N(s,a) and prior probability P(s,a).

The tree is traversed by simulation starting from the root state. At each time step *t* of each simulation an action a_t is selected from state s_t to maximize the action value $Q(s_t,a)$ plus a bonus $u(s_t,a)$. The bonus is proportional to the prior probability but decays with repeated visits to encourage exploration. $a_t = \text{argmax}(Q(s, a)+u(s, a))$

The prior probabilities of each state's actions are gathered only once from the SL network P_σ the first time a leaf is expanded. The leaf node *L* is then evaluated in two different ways: the first by the value network V_θ the second by the outcome Z_L of a random rollout played to a terminal step T using the fast rollout network P_π. These evaluations are combined using a mixing parameter λ into a leaf evaluation: $V(S_L) = (1- \lambda) V_\theta(S_L) + \lambda Z_L$. At the end of a simulation the action values and visit counts of all visited edges are updated. Each edge accumulates the visit count and mean evaluations of all simulations passing through it. Once the search is complete the algorithm chooses the most visited move from the root position.