AlphaGo Paper Overview

AlphaGo introduces a mix of DeepLearning (supervised + reinforcement learning) to speed up game tree search and game evaluation with some traditional approaches.

Given that Go is much bigger than chess (branching factor of 250 and depth ~ 150, would not be feasible to just minimax the whole tree without a depth limit straight to the end of the game, same as Chess.

What works for tic tac toe and checkers won't work for Go. What works for Chess won't work for Go either: Deepmind did Minimax + AB pruning + Pruning optimization + Opening moves dictionary + Pattern recognition / regression + endgame recognition and that worked. For Go, it still would not compute in time, the field is too big.

The biggest challenge of Minimax-like algorithms for Go is that you have to evaluate which branches of game tree are more likely to play out and therefore which branches to investigate further.

AlphaGo build up on the recent success in feedforward neural nets: they've used 13-layered net with 3m boards (game states) and given next moves by humans to train a network (Supervised Learning) that gives a probability destribution of the next turn (policy network).

This network is further improved with reinforced learning on self play games of the SL-based alg with random sampling of boards to prevent overfitting of the network.

Surprisingly, AlphaGo's network only gives them 57% probability accuracy with the test set, but that's enough to outperform all other Go agents at the moment.

Policy Network is basically a tool that says AlphaGo which nodes of the game tree to calculate, along with a traditional MonteCarlo tree (MCTS).

To simulate the nodes faster, AlphaGo uses another RL network to approximate the value function (that's what custom heuristics are doing in our little projects with isolation). They've only had to feed 30m of distinct game boards generated by each different self played game to prevent overfitting on the existing dataset (very similar boards can lead to very different results, hence the network is prone to overfitting).

That combined gives AlphaGo two neural nets + (policy + MCTS and value network) that can traverse and approximate values of the game tree superfast: they win 99.8% of games with the best existing Go agents.

Outcome:

1. Our previous approach to the same problem (minimax + optimisations like AB-pruning, searching for similar or equal game states by rotating problem space / boards) can not

perform that fast, because a trained neural net does just a single ff pass of the net to get the value, and we have to do a ton of calculations, which is impossible for relatively big branching factors (>30?)

2. This technique can be used in all sorts of adversarial search situations, not just games, but I don't know if there are any good papers on this. What should I read next?