Game playing with deep neural networks and tree search

These techniques are based on the research presented by the Google DeepMind team on the game playing AI AlphaGo.[1] Resources on topics discussed are in references section.

-The goal of the AlphaGo team was to create a new approach to computer Go that would achieve superhuman performance. To accomplish this, they used a combination of Reinforcement Learning, Supervised Learning, and Search Trees.

-The pipeline used to create the agent and process moves is outlined by the following:

1) Construct a Convolutional Neural Net(CNN) with the current board state.
2) Train a Policy Network with Supervised Learning(SL) using prior knowledge of expert moves in Go.
3) Train a second Policy Network using Reinforcement Learning(RL) that utilizes self-play to optimize outcomes.
4) Train a Value Network using RL to predict outcomes.
5) Utilize Monte Carlo Tree Search(MCTS) to combine the 3 networks to select actions.

-The first policy network utilizes SL with 30 million positions in the KGS Go Server[2] to evaluate the likelihood of a typical expert move, using stochastic gradient ascent and CNNs. The result is a policy gradient that holds the best common moves from the current state. This policy gradient is then sent to the next stage of the pipeline, the RL policy network.

-The second policy network then uses RL by taking the gradient and running games of self-play to predict optimal positions from optimal outcomes. When selecting opponents during RL, it randomizes previous iterations of the RL policy network in order to prevent overfitting.

-The value network is then constructed from the optimal positions output of the second policy network. This is accomplished by training the weights using stochastic gradient descent. The network then uses RL to do self-play again using the newly minimized values in order to further reduce error.

-Using this value with the minimal error, an MCTS algorithm can be used to predict moves in the future. The algorithm does this by descending the entire tree, and at each time step, selecting the max current value in the tree, and keeping a count of times a node is visited. This value slowly decays as the tree gets deeper, to encourage it to select values deeper in the tree. At the end of the tree search, each node in the tree has it’s value and count saved, and another iteration begins.This allows the algorithm to adaptively choose a better value. The final value chosen is that which has been visited the most times.

-In order to initially evaluate the ability of AlphaGo, the DeepMind team held an internal tournament, pitting AlphaGo against previous iterations of itself, as well as other commercial programs and open source Go bots. The result of the tournament showed that AlphaGo achievied a 99.8% win rate. This was further proved by handicap matches, in which it averaged an 87% win rate. The true test of their goal came when they place AlphaGo against Fan Hui, the winner of the 2013-2015 European Go Championships. AlphaGo won the match 5 games to 0, achieving their goal of being the first computer to defeat a human professional in Go.

References

[1] Mastering the game of Go with deep neural networks and tree search (https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf)
[2] KGS Go Server (https://www.gokgs.com/)

Convolutional Neural Nets - http://www.deeplearningbook.org/contents/convnets.html

Reinforcement Learning - http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching_files/intro_RL.pdf

Monte Carlo Search Trees - https://stanford.edu/~rezab/classes/cme323/S15/projects/montecarlo_search_tree_report.pdf