Review: Training AlphaGo with deep neural networks and tree search

For our project, we built a simple game-playing agent to play the game Isolation, while using techniques such as alpha-beta pruning and iterative deepening to evaluate branches of our game tree with custom value heuristics. In our case, the branching factor and depth of the game tree were fairly small, so we could explore it deeply. However, for a game like Go, where the branching factor and depth (~250 and ~150) make exhaustive search infeasible, better techniques are required when searching the game space.

AlphaGo combines deep neural networks and Monte Carlo tree search (MCTS) to evaluate board positions and select moves. With this new approach, AlphaGo achieved a 99.8% win rate against other Go AI's, and defeated the human European Go champion in a 5-0 series. This article explores the techniques that made this possible.

Neural Network Training

The Supervised Learning (SL) policy network is a 13-layer deep convolutional neural network trained from 30 million positions from the KGS Go Server. It uses stochastic gradient descent to predict likely human expert moves given a data set of positions.

The Reinforcement Learning (RL) policy network aims to improve the policy network by simulating games between the current policy network and a randomly selected previous iteration. This randomness stabilizes training by preventing overfitting to the current policy.

Next, the Reinforcement Learning of value networks focuses on position evaluation and estimates a value function by using policy p for both players. The structure of this network is similar to the policy network, but outputs a single prediction using the strongest policy instead of a probability distribution. The weights of the value network are trained by regression on state-outcome pairs using stochastic gradient descent. To prevent overfitting to the KGS data game outcomes, a new self-play data set was generated from games played between the RL policy network and itself.

Search and Evaluation with Policy and Value Networks

AlphaGo combines the policy and value networks in a Monte Carlo Tree Search (MCTS) algorithm that selects actions by lookahead search. Each edge of the search tree stores an action value, visit count, prior probability. It traverses the search tree by simulation, selecting an action that maximizes the action value plus a bonus proportional to prior probability but decays with repeated visits in order to encourage exploration

When an edge is selected, its leaf node is expanded and this new node is processed through the SL policy network. Then, it is evaluated by the value network and also by a fast rollout policy which is played out until a time limit is reached. At the end of the simulation, these evaluations are combined and the action values and visit counts of traversed edges are updated, and the algorithm chooses the best move from the root.

Results

The final version of AlphaGo used 40 search threads, 48 CPUs and 8 GPUs! They also implemented a distributed version that ran on 1202 CPUs and 176 GPUs. Both versions were significantly stronger than existing Go programs and the European champion player. Even when it only used value networks, AlphaGo performed better than other programs. This article showed that by combining tree search with policy and value networks, AlphaGo was able to reach a professional level in Go, a feat that was thought to be many years away.

[1] David Silver et al., Mastering the game of Go with deep neural networks and tree search
https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf