Research Review

Mastering the game of Go with deep neural networks and tree search.

Gabriela Chiribau

Goals and techniques

Artificially Intelligent (AI) game agents have a difficult time solving the game of Go because of its large set of possible moves. Classic decision tree search with carefully crafted heuristics only achieved amateur level game play. Pachi, a sophisticated Monte Carlo search program, barely ranked at 2 amateur dan on the KGS Go Server.

A decision tree search, habitually used in AI game agents, involves two major steps: calculating a utility value associated with a specific node and deciding on the next action, possibly considering the utility value. Algorithms like MiniMax make two major assumptions: always use the utility value and chose either the max or the min value depending whose player turn it is. It assumes the agent player would always attempt to maximize its value and that the opponent would always try to minimize the player's outcomes. These assumptions are often wrong against human players using, what Danny Kahnemann would call, "expert intuition".

Fighting fire with fire, the article shows how using "intuition" largely improves an AI player's outcome. The state of the art of modelling it is the universal function approximator known as neural network.

Expert intuition

The first step is to capture "expert intuition" via a supervised learning (SL) network trained with 30 million of expert moves from the KGS Go Server. The agent would "look" at the game board and chose the most likely expert move based on the learned model. The AI agent achieved a 57% accuracy when it used all input features and 55.7% using only raw board position and move history. The difference does not seem major, however small variances in prediction accuracy have large impact in the playing strength.

A good expert move prediction only shows the neural network learned its moves well, but it is not necessarily a winning choice. The SL sometimes loses sight of the goal, which is winning and not the prediction accuracy.

The next step in the pipeline is a reinforcement learning (RL) step, where the AI agent learns to select the winning moves (policy) out of the learned expert moves. It achieves this by playing against random instances of its younger self. Randomizing helps it to not obsess, I mean overfit, to a given past policy. The RL network won 80% of the games when playing against the SL former self. It also won 85 of the games against Pachi using no search at all. Not bad for just "guess work".

"Am I winning?"

A value network would guide next the AI agent in predicting whether the move it considers is a winning move. It uses the RL policy to predict the moves for both the player and its opponent. It shares the same architecture with the policy network, however it produces a single value or win/loose prediction instead of a probability distribution for the legal next moves.

Monte Carlo Tree Search (MCTS)

A Monte Carlo simulation plays hundreds of simulations against itself. It could be considered a function approximator, which "learns" through random plays from a given position in the search tree (look-ahead search) and backpropagating the result to the upstream nodes. AlphaGo combined the MCTS algorithm with the policy and value networks. Time is of essence and AlphaGo used an asynchronous, CPU based, multi-threaded search to perform simulations. Alpha Go evaluated the policy and value networks in parallel, using GPUs.

Thinking: fast and slow

A play from a given position in the search tree is called a rollout. AlphaGo used the trained neural networks to select the next moves and evaluate their value towards winning. Larger networks perform better, however are slower to evaluate. In balancing the winning rates, highly dependent on the prediction accuracy, with the evaluation times, AlphaGo used a faster albeit less accurate policy. For example, the high performing rollout policy took 3 ms to select a move while a network with 24.2% accuracy took only 2 micro seconds to select a move.

"Reality is a cloud of possibilities, not a point."

Said Amos Tversky, one of the most famous economists most people have never heard of. Although less accurate, the SL rollout policy, performed better in the overall MCTS AlphaGo algorithm than the stronger RL network. Humans consider a set of promising

moves, and use a combination of intuition and reasoning to select the next move. The RL algorithm relies on learned intuition alone to select a single best move.

A lesson learned

Well trained intuition could lift one's performance from amateur to professional. However, intuition alone, however strong might be, is not enough to compete with the balanced act of gut feeling and reason.