Research review: Mastering the game of Go with deep neural networks and tree search

Nicholas Chen

Introduction

AlphaGo, a new approach to computer Go, made history by having a 99.8% winning rate against other Go programs and defeating Fan Hui, the human European Go champion, by 5-0. This is the first time a computer program has defeated a human professional player, a feat thought to be at least a decade away.

The main innovation in AlphaGo is the combination of Monte Carlo tree search (MCTS) with value networks (which evaluates board positions) and policy networks (which selects moves). These value and policy networks are deep neural networks which take in a board and output probabilities and scalar values, trained by a combination of supervised learning on human expert games and reinforcement learning from games of self-play.

Goals

Their main goal is to search through moves efficiently and effectively – Go has a branching factor of $b \approx 250$ and depth of $d \approx 150$, making exhaustive search infeasible. Even minimax with alphabeta pruning is ineffective. In comparison, for chess, $b \approx 35$ and depth of $d \approx 80$. It is the extremely large search space b^d of Go which led experts to believe that this game is one that will not be solved too soon.

Techniques introduced

Their main contribution is the usage of value and policy networks for evaluation and move selection, and combining it with MCTS. Let us look at the various components of AlphaGo:

Monte Carlo tree search (MCTS)

Monte Carlo tree search (MCTS) is the main search algorithm for most state-of-the-art Go programs out there. While it is not a new technique, let us recap this idea so that we can understand how AlphaGo works. MCTS consists these steps:

- Selection, where successive child nodes of a root node are selected till a leaf node is selected.
- Expansion, where new child node(s) are created and one of them is selected.
- Simulation, where a search is conducted to the end without branching by sampling actions for both players from a policy (a.k.a Monte Carlo rollout).
- Backpropagation, where the result of the rollout is used to update values from the child node upwards till the root node.

Value and policy networks

Previous programs used shallow policies or handcrafted value functions (often based on a linear combination of features). Key innovations in AlphaGo are using value networks for evaluating positions instead of handcrafted value functions, and policy networks to sample actions in place of shallow policies. Using neural networks allow the effective breadth and depth of the search tree to be reduced. These networks are of the convolutional neural network type, used commonly in image analysis, and the input to the networks is the board represented as a 19 X 19 image.

Training of neural networks

- The policy network is first trained via supervised learning, where the inputs are expert human moves from the KGS Go Server and the output is the probability distribution over legal moves. Also, they trained a fast policy network which is less accurate but returns the output faster (by a factor of ~1000).
- Next, they initialized a reinforcement learning policy network with the supervised learning policy network from above, and trained the reinforcement learning policy network via self-play and optimizing for the final outcome .
- Lastly, they train a value network that predicts the winner of games played by the reinforcement learning policy network against itself.

Combining MCTS with policy and value networks

AlphaGo selects actions by lookahead search. The action selected for a state is one which maximizes the action value plus a bonus. The action value is calculated from the leaf evaluations normalized by the number of visits. This leaf evaluation is calculated from a linear combination of the output from the value network and the outcome after a random rollout played using the fast policy network. The bonus is proportional to the prior probability (output from the policy network) but inversely proportional to the number of visits to encourage exploration.

Results

While AlphaGo is the combination of MCTS and neural networks, the standalone neural networks are already breaking new ground:

- The supervised learning policy network predicted expert moves with 57.0% accuracy compared to the state-of-the-art accuracy of 44.4%.
- The reinforcement learning policy network won 85% of games against Pachi, the strongest open-source Go program, without any search.

Run on a single-machine, AlphaGo won 494 out of 495 games (win rate of 99.8%) against other Go programs. The distributed computing version of AlphaGo is even stronger – it won 100% of the games against other Go programs and 77% of games against single-machine AlphaGo.

Using only the value and policy networks (without rollouts), AlphaGo exceeded the performance of all other Go programs in terms of Elo rating, showing that value networks are a viable alternative to Monte Carlo evaluation in Go. The best combination, however, is when the value network and rollout are used together for evaluation, winning over 95% of games against other variants, suggesting that these two evaluation techniques complement each other.

The distributed version of AlphaGo won 5-0 against Fan Hui, the European Go champion for 2013-2015. This is the first time a computer Go program beat a human professional in the full game of Go, a feat previously thought to be at least a decade away. In this match, AlphaGo evaluated thousands of times fewer positions compared to Deep Blue in the chess game against Kasparov, despite the fact that the search space for chess is much smaller than that for Go: AlphaGo had to select positions more intelligently and evaluate them more precisely, closer to how humans play.