AlphaGo Research Review

By Jordan L. Carson

Research Review

In the past few decades, there have been many attempts of solving classic multi-player games of perfect information using artificial intelligence. Go and Chess are both classic games of perfect information, meaning each player is perfectly informed of all events that have previously occurred. "All games of perfect information have an optimal value function, v*(s), which determines the outcome of the game, from every board position or state (s), under perfect play by all players." To solve games of perfect information, agents must evaluate each possible move through recursive computing the optimal value function in a search tree containing approximately $\mathbf{b^d}$ possible moves – where b is the number of possible legal moves and d is a game length. In particular, the game of Go has been viewed as the most challenging due to its enormous search space and the difficulty of evaluating board positions. For chess, the approximate numbers are (b~35, d~80 – leading to $\mathbf{10^{123}}$) and for Go, (b~250, d~150 – leading to $\mathbf{10^{359}}$ different games). Trying to evaluate all possible moves in Go would exceed the number of atoms in the Universe. It's not possible.

Go has been an inspiration for mathematical research and in recent years an ambitious goal within the computer science circle for computational complexity. Because of this, a UK based Deep Learning Company, DeepMind, built an artificial intelligent deep neural network titled AlphaGo in an attempt to push the envelope further in this domain.

AlphaGo uses a neural network consisting of action sampling (policy network) and effective position evaluation (value network).

Policy & Value Networks

Policy Network

Similarly to the policy model we implemented in the Machine Learning Smartcab Image Classification project, AlphaGo estimates the value of each game state in a tree search – however they did so through Monte Carlo tree search (MCTS), using Monte Carlo rollouts which becomes more accurate as the search tree expands. The first stage to train this policy network, using supervised learning, is used to narrow the search to a layer of high-probability actions, and to sample actions during rollouts. They essentially create a convolution neural network representing a 19x19 image of the board, and use convolutional layers to construct a representation of the position. What they implement is very clever – they use these neural networks to reduce the effective depth and breadth of the search tree: using a value network to evaluate positions, and a policy network to sample actions.

The trained network was able to predict expert moves with an accuracy of 57%, 12.6% higher than the highest value from other research teams (at 44%).

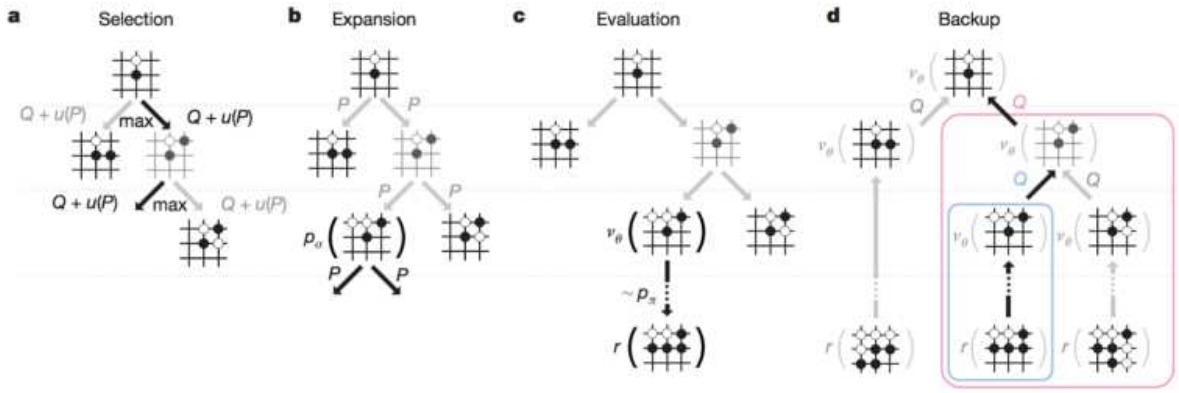
These two networks use supervised learning and reinforcement learning, respectively. The policy network is trained from 30 million positions from the KGS Go Server, consisting of expert human moves. They also train a fast rollout policy that can rapidly sample actions during rollouts, which is used to run rollouts to the end of the game when evaluating nodes in the MCTS.

In the second stage, they train a reinforcement learning policy network to improve the supervised learning policy network by optimizing the final outcome of games of self-play. During this stage, AlphaGo played roughly 1.2 million games with randomly selecting previous iterations against itself. They do this, to correctly adjust the policy toward leaning to the correct goal of winning games rather than maximizing predictive accuracy. Reinforcement learning was able to win more than 80% of games against the supervised learning network. They also tested their network against the strongest open-source Go program, Pachi (another sophisticated Monte Carlo search program heuristic) – using no search at all, the reinforcement learning policy network won 85% of games against Pachi. To compare, previous state-of-the-art networks only won 11% of games against Pachi and 12% against a slightly weaker program, Fuego.

Value Network

In the final stage of training, AlphaGo is focused on evaluating each position, estimating the probability of each move and selecting the move with the maximum action value plus a bonus that is proportional to the prior probability, but decays with repeated visits to encourage exploration. They play games between the current policy network and a randomly selected previous iteration of the policy network. Doing this, randomizing their opponents from a pool stabilizes their training by preventing overfitting to the current policy.

Monte Carlo Tree Search (MCTS)



The above diagram explains the Monte Carlo Tree Search.

a) In the selection stage, each simulation passes through the tree by selecting edges with the maximum value – Q - plus a bonus - u(P) - that is based on a prior probability P.

b) In the expansion stage, if any node is expanded, it is processed by the supervised learning policy network and the output probabilities are stored as prior probabilities for each action.
c) In the evaluation stage, each node is evaluated by both the value network and the fast rollout policy.
d) In the backup stage, action values are updated to track a collection of the mean value of all evaluations during the previous stage.

After all the stages have been processed and the iterative game has reached the end, it chooses the maximum value of Q.

Conclusion

AlphaGo has attracted attention from not only researchers, but the media and individuals outside the computer science circle. To evaluate the performance of AlphaGo, its creators ran internal tournaments among other variants of AlphaGo, other Go programs (including Crazy Stone and Zen, Pachi, and Fuego), and the European champion Fan Hui. All programs were allowed a computation time of 5s per move. The results indicated that AlphaGo ranked several dan ranks stronger than any other computer-based Go program, even with a handicap. AlphaGo was also evaluated against Fan Hui, a professional 2 dan, and winner of the 2013, 2014, and 2015 European Go championships. During 2015, they competed in a formal five-game match – AlphaGo won the match 5 games to 0, leading to the first time a computer program has ever beat a human professional player without a handicap in the full game of Go.

The results of AlphaGo were believed to be at least a decade away, but through the successful introduction of MCTS, led to dominance in this domain. This introduction led to corresponding advances in many other domains, including general game-playing, classical planning, partially observed planning, scheduling, and constraint satisfaction. The results of AlphaGo were astonishing, and will hopefully lead others to pursue similar acts of achievement – to push the envelope one step further.