Stuart Minshull

AI Nanodegree, Udacity

July 1st, 2017

AlphaGo: A Unique Blend of Speed & Accuracy

On September 10th, 2015, AlphaGo[1], an artificial intelligence (AI) Go-playing program developed by Google DeepMind, defeated the human European champion in a formal match 5 games to 0. Due to Go's large board size and complex strategies, the game has long been considered the 'final-frontier' for AI game-playing programs. Through a novel machine learning pipeline of Monte Carlo rollouts, supervised- and reinforcement learning policy networks, and value networks, the DeepMind team developed an AI program that achieved a 99.8% win-rate against other computer Go players, and finally defeated a human champion in a formal match.

The learning pipeline of AlphaGo begins by using supervised learning to train two separate policy networks to predict a human expert's likely moves from enriched datasets of board positions.

The first policy network, a Monte Carlo 'rollout' policy network, uses Monte Carlo algorithms to predict which move a human expert would select by quickly searching to a game-tree's maximum depth. Monte Carlo algorithms randomly sample each player's moves from a pool of likely-good choices, then updates the weights that control which moves are considered likely-good as terminal states are discovered. Rollout policies, while less accurate than deeply trained policies, rely on the law of large numbers to converge towards the optimal move. The rollout policy allows AlphaGo to quickly select a move when running out of time, or when a board-state has not been seen before by the pre-trained value networks.

The second policy network, named the 'SL Policy' network by the authors, uses deep neural networks (192 filters in the tournament version) to create a probability distribution predicting which move a human expert would select for a given board position. This network is trained by randomly sampling possible moves and move-histories from a 30 million position data set, which is pre-enriched by the authors to indicate if the human expert selected that move. The authors achieved an accuracy of 55.7% using only current board positions and move histories, and an overall accuracy of 57.0% with all features considered.

After the supervised learning policy networks have been trained, a reinforcement learning (RL) policy network is trained by playing matches between the current version of the SL policy network, and a randomly chosen previous version of the SL network. This helps prevent

[1] This paper is a summary write up of the original paper published Nature, 'Mastering the Game of Go with deep neural networks and tree search', *doi:10.1038/nature16961*, *D. Hassabis et. al.*

overfitting by preferring policy networks that win, rather than SL policy networks that are very good at predicting human moves (but not at winning). A reward function that provides +1 reward for winning and -1 reward for losing, with no other outcomes, also helps tune the move-predicting SL network into a game-winning RL policy network. When the final RL network was played against the best SL network, the reinforcement network won over 80% of the time.

In the final stage of the pipeline, the authors convert the RL network of trees-of-moves to a value network containing a single value per move indicating relative desirability. The value network is trained using stochastic gradient descent, a technique that tries to minimize the mean-square error of the RL network by allowing a small amount of randomness to affect the move selection. This small, controlled randomness helps avoid local minima and find the true global minima of error. As more game-states are seen, the value network prediction approaches the optimal choice of move, similarly to how the Monte Carlo algorithms rely on the law of large numbers.

When the program is executed at runtime it uses look-ahead search, the same type of search used in Minimax, on both the Monte Carlo rollout network and the value network. The outputs of these policy network searches are combined in an argmax algorithm that selects the move with the maximum combined weighted action value. In addition, this algorithm includes a component that is inversely proportional to the times a state has been seen, encouraging the algorithm to explore newly-seen states that may be good choices many moves in the future. However, this two-policy heuristic is quite computationally expensive: in their tournament design, the AlphaGo code was executed in an asynchronous, distributed manner across many 40 threads, 1202 CPUs, and 176 GPUs.

Through a novel combination of sophisticated machine learning techniques, creative heuristics, and a decision to use probabilistic policy networks instead of intrinsic state evaluation, the DeepMind team has achieved an incredible breakthrough for artificial intelligence. Go, a game long considered out of reach of computer players, has finally been conquered. The techniques learned from AlphaGo offer promising opportunities for artificial intelligence agents of all domains.