Research Review for AlphaGo by DeepMind

Yue Duan

The game of Go has been viewed as one of the most challenging games for AI, mainly because 1) size of the search space (b^d possible moves where b is breadth, the number of legal moves per position and d is depth), and 2) difficulty to accurately evaluate board state and potential moves. Previous commercial and open-source Go programs used Monte Carlo tree search (MCTS) based methods, and these methods have achieved strong amateur level play. Limitations of these methods include shallow policies or using linear combination of input features for value functions, which are not optimal. The goal of this paper is to give an overview of DeepMind's AlphaGo, which achieved professional play and defeated human professional player in the full-size game of Go for the first time.

The work of DeepMind combines supervised learning and reinforcement learning to train a multi-stage deep learning training pipeline. The first stage of the pipeline is a supervised learning (SL) policy network trained on expert human moves. They also trained a fast policy similar to Fuego to rapidly sample actions during rollouts. The second stage is training a reinforcement learning (RL) policy network to improve the policy network by playing with itself and optimizing game outcome. In this stage, the RL policy network adjusts the policy network to optimize for winning the game, rather than the origin SL policy network's goal to maximize prediction accuracy. In the final stage, a value network that evaluates the position and predicts the winner is trained by letting the RL policy network play against itself. The value network is trained by regression on game state-outcome pairs to minimize mean squared error between predicted value and actual outcome using stochastic gradient descent. Once the policy and value networks are trained, they are combined in an MCST algorithm that selects actions based on lookahead search. Each simulation traverses the tree from root state by selecting the edge with maximum action value Q plus bonus u(P) based on stored prior probability P. The node may expand and the policy network will output probabilities for each action, and these probabilities are stored as prior probabilities. When a simulation ends, the leaf node will be evaluated by two methods: 1) the value network v_{θ}, and 2) running a rollout to end of the game using the fast rollout policy and then find out the outcome z_L. The leaf evaluation is formulated as a linear combination of the two: $V=(1-\lambda)\ v_{\theta} + \lambda\ z_L$. It's interesting that value network alone without rollout can provide a good alternative to Monte Carlo evaluation in Go. However, a mixed evaluation with $\lambda=0.5$ gives best performance. The authors think the two evaluation mechanisms compliment each other: the value network approximates game outcome played by strong but slow RL policy network, while the rollouts can precisely evaluate game outcome played by weak by fast rollout policy. There seems to be an interesting trade-off between the two for move selection and position evaluation.

Results of this paper is really exciting: distributed AlphaGo beats all other programs consistently, and defeated human professional play in the full game of Go without handicap for the first in October 2015. And we all know the story after this: AlphaGo defeated Lee Sedol 9-dan late 2016 and then world No.1 player Ke Jie in 2017. Since then AlphaGo has triggered the public's interest in deep learning and more people. I'm curious about what "mission impossible" AI can achieve in the near future.

Reference:

[1] Silver, David; Huang, Aja; Maddison, Chris J.; Guez, Arthur; Sifre, Laurent; Driessche, George van den; Schrittwieser, Julian; Antonoglou, Ioannis; Panneershelvam, Veda (2016). "Mastering the game of Go with deep neural networks and tree search". *Nature*. **529** (7587): 484–489. PMID 26819042. doi:10.1038/nature16961

[2] AlphaGo. Wikipedia. https://en.wikipedia.org/wiki/AlphaGo