AlphaGo summary

Goals and Background

AlphaGo is an artificial intelligence which can master the game of Go and can defeat the professional human player in a full-sized Go game. This feat was considered previously a decade away.

Go game is hard for the computer to solve because the complexity of the board positions and the associated moves. We usually approximate the complexity with b^{d} (b is the search breadth, d is the search depth in a search tree), in Go $\mathrm{b} \approx 250, \mathrm{d} \approx 150$, which means a computer with gigahertz processing rate needs to exhaust the search space in about 10290 seconds to find the optimal action, which is unacceptable.

In order to reduce the huge space effectively and efficiently, AlphaGo uses deep learning to train the policy network which selects the move from the current position (i.e., state), and value network which estimates the winning player of the current game. As we can see, the policy network can cut the breadth of the search tree while value network can simulate the winning rate of the current node, thus save the search depth.

Design and Techniques

AlphaGo trained the Fast rollout policy p_π and supervised learning policy network p_σ with 30 million positions from the KGS Go Server. p_σ achieved 57% accuracy for predicting expert moves against the 44.4% of the previously state-of-the-art methods. While p_π achieved an accuracy of 24.2% but just using 2 µs against 3 ms by p_σ. Then AlphaGo initialized the RL policy network p_ρ with p_σ to boost the training on p_ρ. To avoid overfitting to the single opponent in the training of p_ρ, AlphaGo sampled the delayed experience randomly from the previous iterations of itself in 1.2 million games. By this way, p_ρ achieved 80% against the traditional SL policies with Pachi (an open source program based on Monte Carlo tree search (MCTS) heuristics ranked at 2nd amateur dan on KGS). v_θ can be trapped by overfitting with KGS dataset as the value network just memorized the game outcomes rather than generalizing to new positions. Therefore, AlphaGo used p_ρ to generate 30 million distinct positions for training v_θ. v_θ is extremely effective as it does not need any look ahead search giving the winning player from the global view. It has the similar accuracy to MCTS rollouts with RL policy but saves 15000 times computations.

To ensemble the above policies, AlphaGo designed a new MCTS way includes selection, expansion, evaluation and backup, which depicted in figure 1[1].

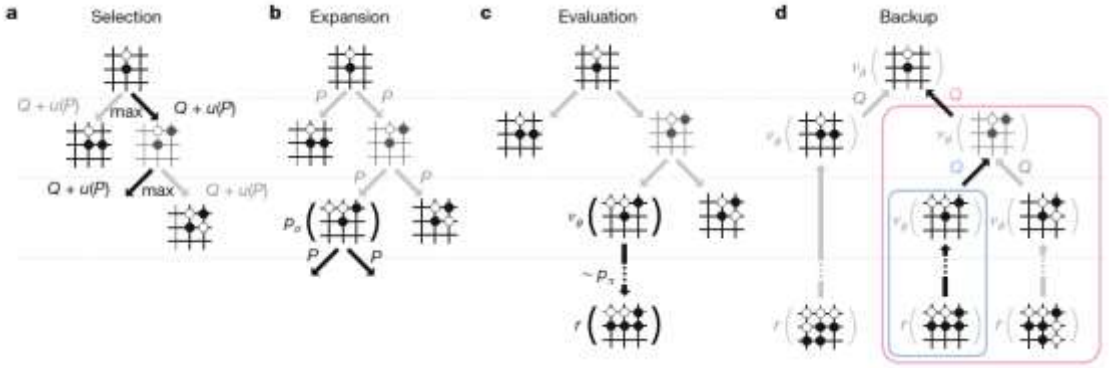

Figure 1 MCTS in AlphaGo

- Selection

  Given the current board position, select the best move with the largest state action value from the active player view.

- Expansion

  If a leaf node in a search tree needs to be extended, AlphaGo uses p_σ to expand the branches with the corresponded probability.

- Evaluation

  AlphaGo simulates the rollouts in many rounds with p_π until game ends.

- Backup

  After simulation, update the state action value for each visited edge in the search tree. Once the search is complete, the algorithm chooses the most visited move from the root position.

Results and Conclusions

AlphaGo won 99.8% matches against other state-of-the-art Go problems (e.g., Crazy Stone, Pachi, Zen). And AlphaGo is also scalable, its distributed version can beat the singleton with 77% winning rate. By the means of deep neural network and tree search, AlphaGo defeated Fan Hui (won the European championships many times) by 5 games to 0.

In a word, AlphaGo mastered the game of Go and performed even better than the professional level of human. This result strongly pushes the development of AI and deep learning.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.