REVIEW: Mastering the game of Go[1]

By Alessandro Bersia

The objective of this article is to explain how Artificial Intelligent has achieved to master the game of Go, with neural networks and tree search.

PROBLEM TO SOLVE (Huge search tree)

When it comes to large games (such as Go or Chess), recursively computing the optimal value function in a search tree which is extremely huge ($b^d = 250^{150}$, in the Go case) would be impossible within a reasonable time frame. Two general principles can reduce the search space:

1. The depth search can be reduced by truncating the search tree at state *s* and replace the subtree below by an approximate value function that predicts the state *s*
2. The breadth of the search can be reduced by sampling actions form a policy probability distribution over possible moves in each position

The application of these principles will be discussed further.

ADVANCEMENT IN DEEP CONVOLUTIONAL NEURAL NETWORKS

Thanks to the advancement in the performances of deep convolutional neural networks, it has been possible to reduce the effective depth and breadth of the search tree. The method used consists in passing the board position as 19x19 image and therefore using convolutional layers to construct a representation of the position. The neural networks reduce depth and breadth by evaluating the positions with a value network and sampling actions with a policy network.

ALPHAGO IMPLEMENTATION

AlphaGo uses neural network and Monte Carlo Tree Search. The following policy neural networks are used:

- **Supervised Learning (SL) policy networks**: SL is trained directly from experts' human moves. In addition, it trains a **fast policy** to rapidly sample actions during rollouts
- **Reinforcement Learning (RL) policy networks**: RL is trained by playing against itself and it kept the network weights of the winner, thus dramatically improving itself
- **Value Network**: the value network predicts if the current position will result in winning or losing the game for the current player. It is trained on the games played by the RL policy network against itself.

SEARCHING PHASES

The AI agent searches the best move across the tree across four phases.

1. **Selection Phase:** the agent chooses the edge with maximus value Q plus a bonus u(P) that depends on a stored prior probability P for that edge
1. **Expansion Phase**: the agent creates a down brunch and identifies a possible best move
2. **Evaluation Phase**: The leaf note is evaluated using the value network and by running a rollout to the end of the game with the fast rollout policy
3. **Backup Phase**: Q values are updated accordingly

Ultimately, the AI agent can choose the move that results in the highest Q value.

CONCLUSION

The algorithm used successfully combines neural network evaluations with Monte Carlo rollouts.

The results are impressive and represent a milestone in the Artificial Intelligence field: AlphaGo surpassed any human player and other Go programs based on high-performance MCTS algorithms.

[1] https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf