Mastering the game of Go with deep neural networks and tree search: a brief summary

The article presents AlphaGo, a very successful AI for playing Go. Go is a traditional Chinese game where the players take turns placing stones on the game board. The game's goal is to surround a larger total area of the board than the opponent.

Despite rules are quite simple, Go is a very complex game where the wide–ranging number of movements available makes it more difficult than chess. Due to its complexity, previous AI agents were not able to get above the amateur level and experts in the field thought it would take at least a decade of work to get to a professional level.

However, in 2015 AlphaGo surprised the world after defeating Fan Hui, a professional player who had won the last three European Go championships, in a formal five-game match. AlphaGo won the match five games to zero.

Traditional methods, like Monte Carlo tree search (MCTS), depth-first minimax search and alpha-beta pruning, which achieved superhuman performance in other games like chess or checkers, were not effective in Go.

To overcome this problem, engineers designed a solution that used value networks to evaluate board positions and policy networks to select moves. Combining these elements with Monte Carlo simulations, AlphaGo achieved a 99.8% winning rate against others Go programs.

First, they used a supervised learning (SL) policy. This 13-layer convolutional neural network was trained with a dataset of 30 million positions to predict expert moves. With this policy, a 57% of accuracy was obtained, while previous methods achieved only about a 44%.

The second stage was a reinforcement learning (RL) policy. Starting the same way as the SL policy, the RL was trained by playing against a randomly selected, previous iteration of the policy network. The RL policy network won more than 80% of the games against the SL policy network.

Then the team focused on position evaluation. They used another neural network that could assign a score to any position on the board. The architecture of this value network is similar to the RL policy network, but the last layer is just a single node that outputs the probability of winning the game from the given board position. To train this network, the engineers generated a new 30 million dataset positions and had the network playing against itself.

Finally, they combined the policy and the value network in an MCTS algorithm. In order to apply these techniques efficiently, AlphaGo uses an asynchronous multi-thread search that executes simulations on CPUs while policy and value network run on GPUs. In the test, they used two different implementations: a single machine version and a distributed version, having the latter a higher percentage of victories.

These techniques allowed AlphaGo not only to win the games, but also doing it evaluating thousands of times fewer positions than Deep Blue did in it is chess match against Kasparov. Moreover, while Deep Blue relied on handcrafted heuristics, the neural networks of AlphaGo were trained directly from gameplay, suggesting that this approach could be used to solve problems, which are currently too hard for traditional AI methods.

JESÚS MARÍA HERRUZO LUQUE