Research review

Mastering the game of Go with deep neural networks and tree search

by David Silver et al. , Nature, VOL 529, 28 January 2016

Go is a deterministic game. From one state of the game it is possible to derive all the possible preceding moves and states up to the end of the game. A brute force technique to solve a deterministic game is to build recursively a search tree. In the case of Go, this exhaustive search is infeasible, because the size of the search space is approximately 250^{150} (where 250 is the number of legal moves per state, and 150 is approximation for game length from a state). Search can be optimised in many ways, for example using probabilistic Monte Carlo tree search (MCTS) mechanism. These methods have so far achieved levels of strong amateur human players.

This paper present a novel architecture of combination of deep neural networks and tree search. The program implemented can play on the level of best professional players.

Techniques introduced

The solution uses ‘policy networks’ to evaluate moves and ‘value networks’ to evaluate the board position (outcome of the play from certain position). Training pipeline of these networks has three stages:

1. Supervised learning (SL) of policy networks by human expert moves. Network has 13-layers of alternating convolutional layers (with weights) and rectifier non-linearities. The network is trained with 30 million positions from KGS Go Server. These are randomly sampled state-action pairs. The output of this phase is the probability distribution over all legal moves of a state.
2. Reinforcement learning (RL) of policy networks with structure identical to the previous stage. Network weights are initialised with the values from previous step and the network is trained by playing games between the current network and some earlier state of the network (self-play). The reward function gives zero for all non-terminal steps, +1 for winning terminal step and -1 for loosing terminal step. The weights of the network are updated on the basis of the outcome of the reward function in order to win more game.
3. At the final stage of training, RL policy networks are used for position evaluation. Value function gives the outcome from a position when certain game policy is followed by both players. Finding the optimal value function is infeasible. Instead, value networks aim to give best approximation of value function using RL policy networks. The output of value network is a single prediction value. Value networks are trained with generated self-play data in order to avoid overfitting by training with strongly correlated KGS recordings.

Finally, AlphaGo combines policy and value networks in an MCTS algorithm. The tree has game positions as nodes, and the edges store an action value, prior probability and visit count. When traveling down, simulation selects the edge with maximum action value and bonus (which depends on the probability). When a leaf node is reached, it is expanded using SL policy network giving new state nodes and prior probabilities for new edges. Leaf nodes are evaluated using the value network and fast rollout of T steps ahead, giving new action values for the edges. At the end of the simulation, action values and visit counts are updated upwards. From the root position, the edge with largest visit count is selected as next move.

Major results

The RL policy network only version (No value networks or search tree) was able to win 80% of games against strongest open source Go programs. This is much better than earlier CNN based solution has achieved. The complete AlphaGo solution was able to win 99.8% of games against other Go playing programs, including the best commercial programs. These programs used MCTS methods.

The absolutely most impressive result was that the solution was able to beat a human 2 dan player and multiple European champion in a tournament by 5 matches to 0. This was the first time a computer Go program defeated a human professional player. This clearly marked a new milestone in the history of AI.