A Research Review of AlphaGo

by Shijie Wang

In the game of Go, the branching factor is 2 to the factor of 250 and the typical number of moves is the factor of 150. Although the effective search algorithm can reduce the search space by the position evaluation and the breath of the search by sampling actions from a policy that is probability distributed over possible moves, the go program which based on Monte Carlo tree search(MCTS) has been limited to shallow policies or value functions based on linear combination of input features.

Goals

Overcome the AI challenge on the game of Go to defeat the human professional players by reducing the search space of possible moves.

Techniques

A combination training of supervised learning from human expert games and reinforcement learning from games of self-play. It is also beyond supervised learning: using a reinforcement learning to enable the system to against other instance users of itself could create and manage a much stronger game-playing agent.

a). System Design:

Alpha go uses **neural networks** combining with **Monte Carlo Tree Search(MCTS)** to make the game agent.

The **deep neural networks** consist:

- One **"value networks"** – evaluate board positions in search tree and identify the most promising move by looking at the images of the and assigned values to each position. It estimates the possibility on which player will have a higher chance to win. In order to make the value network generally fit the games, AlphaGo team trained the games collectively about 30M human games and 1.5 billion self-play games.
- Three **"policy networks"** – select moves. These networks are designed to decide which move to investigate and which ones to set

b). Training Pipeline:

Stage one - **Supervised learning(SL)** to predict the expert moves in the Game of Go. The neural network design and training is a supervised learning policy network which consists of 13-layer deep CNN training on 30 million Go game positions. When a game position is given, the network will predict the "most likely" moves.

- A reinforcement learning policy network is initialized to the SL policy network.
- policy gradient learning to maximize the outcome. (that is wining more games)
- a regression is used to train a value network to predict the expected outcome

Stage two – improving the policy network by policy gradient **reinforcement learning(RL**) which is initialized to the SL policy network. It maximizes the outcome which means winning more games, rather than only predict the next move. It has the same structure of SL network.

Stage three – **Fast Fallout(FR)** policy network. It is also the network to predict the next move but evaluate in a much faster speed, although may not as accurate as SL network. It is used for position evaluation, estimating a value function that predicts outcome from position s of games played by using policy p for both players, outputs a single prediction instead of a probability distribution.

c). Searching:
To sum up the search algorithm which has been implemented in AlphaGo, it uses Monte Carlo tree search, Combining MCTS with deep neural networks using asynchronous multi-thread search that executes simulations on CPU, and computes policy and value networks in parallels on GPU.

The search starts from the current position as the top of the tree, going down to the edges to possible moves which holds a value Q that indicates how good this potential move is. There are four phrase within the searching process.

1. The agent chooses with the edges which holds the highest value Q, and explore with that branch.
2. When it reaches the leaf node of that branch, it runs the SL policy to produce the “most likely” move.
3. Then goes to the evaluation phrase. It runs the value network once to evaluate the new position and use FR network to simulate a play game from that move with as many games to run as possible
4. After phrase three, the search algorithm will propagate the information which are collected previously and bubble it up to the top of search tree, which is the current position.

Result

- 99.8% winning rate against over other Go program.
- The RL policy network won more than 80% of games against the SL policy network, 85% against Pachi.
- Won 77%, 86%, 99% against Crazy Stone, Zen and Pachi.
- Distributed version won 77% against single-machine AlphaGo
- Won all others 100%
- Win 5-0 to the Europe Go human championship, Fan Hui.

Reference

[1] Mastering the game of Go with deep neural networks and tree search, by David Silver et al @ https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf