Research Review

By Nimish Sanghi

DeepMind team have a AlphaGo at game of Go

Game of Go with Neural Networks and tree search[1]

Background

Most classical games such as Chess, Go, backgammon, Scrabble etc can be programmed in computer to play against humans with varying level of strength. Theoretically all such games involve searching the state space of game board and a brute force search can solve all these games to perfection. However, depending on the branching factor at each move and depth of games make the search tree grow exponentially making it impossible unless some ways to curtail the depth and breadth are deployed.

Game of Go is particularly tough. A game of Go with 19x19 board has a branching factor 'b' in range of 250 and depth 'd' in range 150. This makes the search space to the order of $b^d = 250^{150}$. Compared to this 8x8 standard chess has a search space of about 35^{80} nodes.

The purpose of this paper is to explain the approach DeepMind took to design an intelligent agent which could defeat the world's best Go human players using some a combination of neural networks, reinforcement learning approach to feed a Monte Carlo tree search (MCTS).

Approach

Traditional approaches in reducing search space involves a) curtailing the depth by using position evaluation wherein evaluation functions evaluate an intermediate board position as an estimate of end game result, and b) reducing the breadth by sampling actions from a probability distribution of all actions at a given position ~ p(a|s).

Monte Carlo Search Tree (MCTS)[2] is a popular technique which expands search tree using random sampling in search space. Most earlier approaches in solving GO used MCTS enhanced with policies that are trained to predict human expert moves. However, the policies and value functions used till recently were based on a linear combination of input features.

DeepMind team used neural networks to reduce the effective depth and breadth of search tree wherein positions were evaluated using a value network and sampling actions were chosen using a policy network. These policy and value networks were combined with MCTS.

Policy Network: First a Supervised learning(SL) policy network was built using convolutional layers with rectifier non-linearities (RELU) and was trained using human expert positions. The SL policy network was further trained into reinforcement learning (RL) policy network using policy gradient approach. RL policy network was also similar to SL policy network in terms of architecture i.e. using convolutional network with rectifier non-linearities.

The self play positions generated by this RL policy network was used to train a value network using regression. Value network also consisted of multiple convolutional layers+RELUs.

---

[1] AlphaGo by the DeepMind Team - https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf

[2] Monte Carlo Tree Search - https://en.wikipedia.org/wiki/Monte_Carlo_tree_search

AlphaGo combined policy and value networks in MCTS algorithm to select actions by look ahead search. These evaluations of policy and value networks required several orders of magnitude more computation that traditional search heuristics. AlphaGo team used an asynchronous mutli-threaded search to achieve effectiveness of combining MCTS with deep neural networks[3].

Results

By combining MCTS with deep neural networks, AlphaGo team achieved a 99.8% winning rate against other Go programs and defeated the human European Go champion by 5 games to 0. This was the first time a computer program defeated a human professional player in the full-sized game of Go. AlphaGo team hopes that similar approaches are used in other seemingly intractable artificial intelligence domains.

[3] Deep Learning - https://en.wikipedia.org/wiki/Deep_learning