Udacity Artificial Intelligence Nanodegree
Project – Building a Game-Playing Agent
Research Review

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

Invented in ancient China, Go has been regarded as one of the oldest and hardest game to tackle until today. It is an abstract strategy board game for two players, in which the aim is to surround more territory than the opponent. There has been attempt of using modern artificial intelligence and advance computational power to play the game with the highest possibility of winning ever. In this regard, Google's DeepMind in London developed AlphaGo – an automated computer program to play the board game Go. This article summarizes the novelty and characteristics of AlphaGo in terms of the hardware and software design.

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. A game of Go using a 19x19 board has a branching factor of 250, a state-space complexity of 10^{170}, and a game-tree complexity of 10^{250}. In contrast, chess has much smaller values: branching factor of 35, state-space complexity of 10^{47}, and game tree complexity of 10^{123}. According to [1], Go was previously thought to make the game unconquerable by computers until decades in the future.

Key ingredients of AlphaGo includes the use of convolutional neural network, the use of 'value networks' to evaluate board positions, the use of 'policy networks' to select moves, reinforcement learning, monte carlo tree search and distributed computation.

- Convolutional neural network: The board position is passed as a 19 × 19 image and convolutional layers are used to construct a representation of the position. As a first stage of supervised learning, these neural networks are used to reduce the effective depth and breadth of the search tree: evaluating positions using a value network, and sampling actions using a policy network.
- Reinforcement learning: The second stage of the training pipeline aims at improving the policy network by policy gradient reinforcement learning through the use of value networks and policy networks.
- Value networks: It is a neural network which predicts the next move and is used to narrow the search to consider only the moves most likely to lead to a win.
- Policy networks: It is another neural network which reduces the depth of the search tree, estimating the winner in each position in place of searching all the way to the end of the game.
- Monte carlo tree search: AlphaGo looks ahead by playing out the remainder of the game in its imagination, many times over and uses deep neural networks to guide its search. In order to reduce the enormous search space, it combines a tree search with two deep neural networks (i.e. the value and policy networks mentioned above), each of which contains many layers with millions of neuron-like connections. During each simulated game, the policy network suggests intelligent moves to play, while the value network astutely evaluates the position that is reached. In other words, policy network and value network work hand-in-hand in guiding its search in the enormous game tree.
- Distributed computation: Distributed version of Alphago consists of multiple machines, 40 search threads, 1,202 CPUs and 176 GPUs.

The competition result of AlphaGo is encouraging. In October 2015, AlphaGo became the first Computer Go program to beat a human professional Go player without handicaps on a full-size board. In March 2016, it beat Lee Sedol in a five-game match, the first time a computer Go program has beaten a 9-dan professional without handicaps, giving a final score of 4 games to 1 in favour of AlphaGo.

Reference:

[1] Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." Nature 529.7587 (2016): 484-489.
[2] http://opencuny.org/machinelearning/files/2016/03/AlphaGo_MLRG.pdf
[3] https://research.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html