Udacity Artificial Intelligence Nanodegree

Research Review: Mastering Go with Deep Neural Networks and Tree Search

> Author: Ke Zhang
>
> Submission Date: 2017-06-23 (Revision 1)

Introduction

The research paper [1] is about a new approach to master the traditional board game Go. In the field of artificial intelligence, the challenge was thought to be unconquerable some years ago because of the degree of massiveness of the game. A game of Go using the default 19*19 board has a branching factor of 50, a state-space complexity of 10^{170} and a game tree complexity of over 10^{250}. [2] Chess in comparison has multiple magnitudes smaller values of 35, 10^{47} and 10^{123}. According to the paper this modern-art of game-playing AI named AlphaGo, developed by Alphabet Inc.'s Google DeepMind in London In October 2015, is currently the strongest Go player and the first program to defeat a Go world champion.[3]

Design Goals and Techniques

Go is an example for fully-observed and fully deterministic adversarial game. The paper introduces a new way to solve the game combining the existing techniques like deep neural networks (DNN), reinforcement learning (RL) and Monte Carlo Tree Search (MCTS).

The model is constructed in four stages:

- Stage 1: SL policy network to clone behavior of human professional players

  > The board positions of over 30 million Go games by human experts are treated like a 19*19 images and are used as input to train a 13-layer deep convolutional neural network (CNN) followed by rectified linear unit (ReLU) to build a policy network. Given a game position, this network can predict the probability of the next expert moves with a 57% accuracy. A faster but less accurate rollout policy is also generated which uses a linear softmax function which needs only one-tenth of the time but its accuracy decreases to 24%. The policy network helps to reduce the breadth of the search to predict the next best moves.

- Stage 2: RL policy network from deep learning of self-plays

  > RL policy network uses the result of SL policy network as the starting point and gets perfected with self-play against earlier incarnations of itself over 1.2 Mio times using a policy gradient RL algorithm. During the play the less accurate, but faster SL rollout policy network is used to play out the rest of the game and to evaluate the most likely outcome of the next move. When the two SL and RL policy networks were played head-to-head against each other, the RL policy network won over 80% of the games.

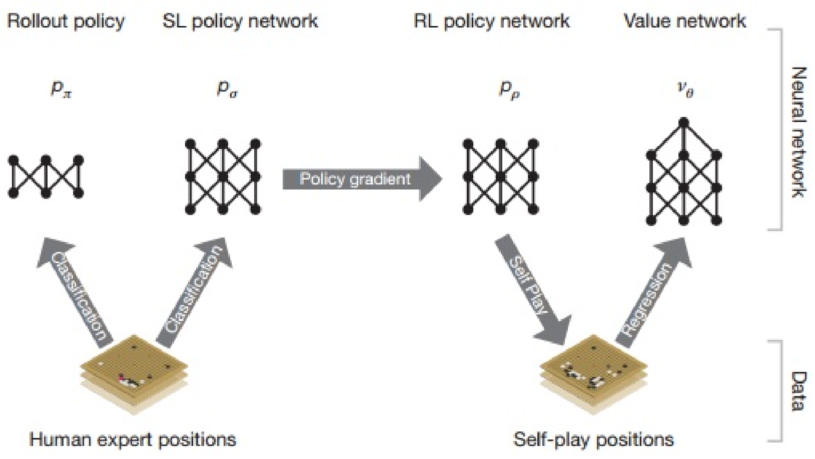
- Stage 3: RL to train the value network (cf. Fig.)

  > While the task of a policy network is to propose the next best moves, the value network is to predict the winner probability of each selected board position. The value network is derived from the same RL policy network by playing self-games over 1.5 billion times until the game terminated. Again to prevent overfitting, a previous iteration of the RL network is randomly selected as opponent to stabilize the discrepancies between the training and validation results. The value network helps to prune poor performing moves and reduces the depth of the search.

- Stage 4: MCTS with policy and value networks to play Go game

  > AlphaGo is a game playing tree search agent using pre-trained SL policy and RL value networks. (According to the authors on the question why SL policy is selected instead of the RL policy, they argued that the SL policy performed better probably because humans tend to select a diverse beam of promising moves, whereas RL optimizes for the single best moves only.) In the allocated time, AlphaGo tries to calculate the next potential move following these four steps:
  >
  > 1. Selection: AlphaGo performs look-ahead search and selects the edge with the highest score.
  > 2. Expansion: AlphaGo explores that branch by performing the slow SL policy network on the leaf nodes.
  > 3. Evaluation: the simulation of the policy and value networks is separated into asynchronous multi-threaded search tasks and ran on computation nodes (or CPUs) in parallel until a predefined time limit is reached.

> 4. Backup: the results are collected and bubbled up the search tree to update the score of that branch. The move with the highest score so far is selected as final result.



Results and Further Thoughts

In the paper, the performance of a local and a distributed variant of AlphaGo are evaluated. Both variants significantly outperform the strongest existing Go AIs with an astonishing over 99.8% winning rate. By allocating more hardware resources in the distributed version, alphaGo performed even better. When playing against the best professional european player, AlphaGo defeated him in five games by 5 to 0.

It's to be mentioned that there were no new algorithmics introduced by the paper. It's the way how AlphaGo utilizes existing techniques to make fundamental breakthroughs in AI, but it's still only an example of a narrow AI[4].

References

- [1] Mastering the game of Go with deep neural networks and tree search, Nature
- [2] Game complexity, Wikipedia
- [3] Deepmind: The story of AlphaGo so far
- [4] Medium: AlphaGo, in context