[toc]

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

A review of the article published in Nature[link] by the Google DeepMind[link] team in which they describe how they built the famous AlphaGo.
This review is written in partial fulfillment of the Nanodegree in Artificial Intelligence (AIND) by Udacity[link to AIND].

Goals

The goals laid out by the authors of this paper were two-fold. Firstly, they wanted to showcase the exciting possibilities of *deep learning* by improving on the current state of the art in GO game playing agents to such a level previously thought impossible. Secondly, they wanted to move beyond the realm of conventional game playing agents and implement the concept of *self playing* through *reinforcement learning*.

Design Implementation

The authors set out to achieve their goals through the introduction two new types of board evaluation methods: *value networks* and *policy networks*. These methods were inspired by the recent advancements of convolutional neural networks (CNNs) in the domain of computer vision, amongst others.

The basic premise was to train a *value network* to assign values (or weights) to each board position, indicating which positions provide a positional advantage. Then, make a decision about which move to make next query the *policy networks*.

Two *policy networks* were trained:

- ρ_{σ}: Trained on annotated expert moves using Supervised Learning (SL)
- ρ_{ρ}: Improving ρ_{σ} through Reinforcement Learning (RL)

And one *value network* was trained:

- v^ρ: Trained using RL through predicting the game outcome from a certain position, by playing the game using ρ_{ρ} for both players.

The SL policy network (and a smaller version of it called ρ_{π}) is used to evaluate each possible move while searching the game tree. The RL policy network ρ_{ρ} is used to estimate the *value function* for the *value network*.

Finally, the *value network* and *policy networks* were combined using Monte Carlo Tree Search (MCTS). A leaf position is evaluated using ρ_{σ}, and the output probabilities are used as the prior probabilities for all the possible actions from that leaf. The leaf nodes are then evaluated by mixing the outcomes of the value network $v^\rho(s)$ and the outcome of ρ_{π} for that node.

This whole process is repeated for multiple simulations, and after each simulation, the values of each leaf node are updated. After completing all the simulations, the game agent will choose the action with the highest value, plus some bonus term which encourages the agent to explore.

An interesting aspect of the design implementation is how the authors overcame overfitting when training v^ρ. The issue that they encountered was that while predicting the game outcomes using ρ_{ρ}, successive board positions are highly correlated, but the regression target z - the final outcome of the game, was shared for the entire game. Thus, the value network started memorizing entire games, rather than generalizing to new moves.

To overcome this issue, the authors generated a new data set from the self play of ρ_{ρ}. This is the beauty of reinforcement learning - the ability to generate data while learning, and to then learn from that data.

Results

The results presented in this paper are unprecedented and caused a global upset, with experts predicting these results would only be achievable within the next decade at best. Through the combination of the novel 'value networks' and 'policy networks' to evaluate and choose new board positions, AlphaGo defeated Fan Hui, the European Go champion by 5 games to 0, and boasts a 99.8% winning rate against other Go playing programs.