RESEARCH REVIEW

Mastering the game of Go without human knowledge

Submitted by
Matt Toledo
As part of the AIND Program, Udacity

The paper, Mastering the game of Go without human knowledge, introduces a deep neural network that is able to achieve superhuman proficiency in the challenging domain that is the game of Go. While this has been previously achieved by AlphaGo, what this paper introduces is the way to achieve this while starting *tabula rasa* and without human data.

This miraculous feat was achieved by scraping the supervised learning part of the training pipeline and keeping only the reinforcement learning components. Previously as part of the training pipeline AlphaGo was trained to identify the move a human professional would take. In order to remove the limits imposed by human data, AlphaGo Zero, was trained strictly by self-play. Using self-play reinforcement learning, ensures that the agent being trained always has an opponent of equal skill to better itself against. As games are played, agents are evaluated to find a better performing agent. This agent is then used to update an underlying deep neural network which then creates stronger agents.

To help the agents select optimal moves, a deep neural network is used to evaluate the state of the board. The network, which is comprised of convolutional, batch normalization and ReLu layers, is 79 parameterized layers deep with an additional two layers for the policy head and three layers for the value head. As inputs the network takes a representation of the board and the move history. The network then calculates both the probabilities of all available moves at that position as well as the probability that the current player wins the game from the current board state. This information is then used to guide a Monte Carlo Tree Search without using any rollouts. To improve the network, the best performing agent is used to generate self-play data. This data is then used to minimize the difference between the predicted outputs and the actual observed results.

A simpler variant of the Asynchronous Policy and Value Monte Carlo Tree Search algorithm used in previous versions of AlphaGo was used. Each node in the tree contains an edge for each legal move at that step. Multiple simulations are then ran on separate search threads that terminate when they reach a leaf node. At the start, on its way to a leaf node, the simulation selects the move with the highest prior probability and low visit count at each node to start. Over time, during training the simulations start to prioritise edges with a higher action value. When a leaf node is reached it is then added to a queue to be evaluated by the neural network. Once evaluated the leaf node is expanded and all new edges are initialized. Then all visited nodes are updated via a backwards pass. Finally at the end of the search, a move is selected with some probability which controls the level of exploration. The new state becomes the new root node and the sections of the tree that can't be visited are discarded. This process then repeats again with the agent forfeiting should the value of its root node and best child node fall bellow some preset threshold.