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

Esteban Rodríguez Betancourt

April 22, 2018

Go is a game with a huge legal move space: 250^{150}. With such number of legal moves common approaches like Minimax, Monte Carlo Tree Search or throwing more hardware to the problem isn't enough.

DeepMind team was able to make a breakthrough in Go[1] by combining the following techniques:

- Monte Carlo Tree Search
- Using Deep Convolutional Neural Networks for evaluating board positions and selecting moves
- Using supervised learning for bootstraping 'value networks' and 'policy networks'
- Using reinforcement learning to improve on both previous networks
- Make the system play against itself to improve further the learned behaviors
- Distributed training

Monte Carlo Tree Search

Monte Carlo Tree Search is a process where the moves tree is explored in the following order:

1. A move is selected
2. The branch is expanded
3. Random playout is performed from this point
4. The game result is used to update information in the selected branch

The process is repeated until the time allocated is spent, so it selects the move with the best "win/plays" ratio.

Deep Convolutional Neural Network

The Go board provides perfect information to any viewer. As such, DeepMind considers it to be a good candidate to apply a Deep Convolutional Neural Network to evaluate the board.

A Convolutional Neural Network just sees a window of the input at a time, so it is able to generalize faster and learn features from the input set. That is why it is useful for image classification tasks, as it tolerates better image translations and rotations.

In the case of Go, a CNN may be able to identify situations, regardless of position of the board where they happens.

Supervised and Reinforcement Learning

They used Supervised Learning to train the networks, based on previous games databases. This makes a good start and provides good results, but the network tends to just memorizes games.

To force the network to learn new situations they used Reinforcement Learning, and made the system play against itself. Reinforcement Learning gives rewards to the player that wins a game, so it allows to do unsupervised learning.

When training a neural network one of the most difficult steps is getting good training sets, so being able to use unsupervised training (via reinforcement learning) on a problem is achieving a big milestone in its own right.

Hardware and Distributed Training

AlphaGo was implemented in a distributed version and a single-node version. the final version uses 40 search threads, 48 CPUs and 8 GPUs, while the distributed version uses 40 search threads, 1202 CPUs and 176 GPUs.

The computations are splited between search threads and GPUs. As MCTS requires simulating plays that task is handled by the CPUs. The computation of the policy and value neural networks was handled by GPUs, as they are better suit for performing lots of matrices calculations.

Regarding the hardware there are two highlights: AlphaGo performed less computations than DeepBlue agains Kasparov, by using a smarter strategy for prunning the moves tree. Also the distributed AlphaGo was able to beat the single-node version most of the times.

References

[1] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.