AlphaGo Un-packed

This paper presents a summery of the AlphaGo algorithm, that was developed by Google Deepmind, and was used by the first computer to defeat a human professional Go-player.

The material presented in this paper is based on the seminal paper "Mastering the game of Go with deep neural networks and tree search" that provides a detailed description of how the AlphaGo algorithm is structured, how it was trained, and how all and parts of it stack up against current state-of-the-art Go-playing algorithms.

Introduction

The goal of the AlphaGo algorithm is very simple: *To master the game of Go*.

The game of Go is a classic example of a game of perfect knowledge, where both players have full, or perfect, knowledge about the game at any given board position, *state*. Other games of perfect knowledge include chess, checkers, and othello.

The purpose of an algorithm for playing a game of perfect knowledge is to provide the next move, *action*, a player should take given the current board position. By evaluating all possible games from a current board position, it is theoretically possible to find, the best move.

The process of finding this move is typically referred to as *searching* for the next move, and the challenge for a game playing algorithm is, that the *search space* in a game like Go like is so great, that it's impossible to perform a brute force search of the entire search space.

Consequently, a game playing algorithm must use what it learns while searching for the next move, to reduce the size of the search space.

Techniques

The AlphaGo consists of two deep neural networks:

- A *policy network*, Po to find the next action, given a state. This is output as a probability distribution over all possible actions.
- A *value network*, Vo, that takes a state and outputs the expected outcome of this state.

To search for the next action, AlphaGo does the following:

1. Select the *action* with the most value
2. Expand the selected *action* using the policy network, Po, to get the probability distributions for the actions
3. Evaluate the expansion by (1) playing the game to end (rollout) using a fast policy network Pp trained on "small pattern features", and computing the winner, and (2) using the value network, Vo. The results of the two evaluations is combined using a mixing parameter, lambda.

AlphaGo performs this search as a Monte Carlo tree search, which is actually the same technique employed by the strongest open source program, Pachi.

At the end of the simulation, the most visited action is chosen. As the selected *action* in a simulation step is the *action* with the most value, selecting the most visited action is the same as selecting expected *action* from the Monte Carlo tree search.

The policy network, Po is trained by first using supervised learning to train a convolutional neural network using 30 million (human) moves from recorded games. A reinforcement learning network is then initialized from this network, and is then improved using policy gradient learning.

A new data set is generated by letting the reinforcement learning network self-play, and the value network is trained using regression to predict the winner from board positions in the self play dataset.

Results

The resulting AlphaGo algorithm proved to be very capable.

Running on a single machine, AlphaGo was able to win 494 of 495 games played against other statet of the art Go programs. A distributed version of AlphaGo was able to defeat a single-machine AlphaGo opponent 77% of games, and 100% games against other Go programs.

The article does not specify whether the other Go-programs were capable of running distributed, but even if they are, the results at least suggest that AlphaGo is able to more efficiently utilize the additional compute resources in the distributed configuration.

Concluding Remarks

The media hype that surrounded the tournament in which AlphaGo defeated a professional human player all but predicted the looming threat of AI.

When un-packing the AlphaGo algorithm, it becomes clear, that at least for now it is little more than a very clever way to search for the best move amongst many, in the game of Go, and until we're able to transfer that knowledge to something else, the AlphaGo algorithm does only that.

What makes AlphaGo more interesting to me, than "simpler" search algorithms such as MiniMax, is that it includes a "trained" component, in the form of deep neural networks.

It would have been very interesting in the authors had also tried to analyse what meta-features the layers, in the network represent in the same way it can be shown for CNN's trained on image data.