Research review for AlphaGo program

The game of Go is a most challenging classic game for artificial intelligence because of its complexity. Most of the games can be solved using the game tree search with evaluation function to evaluate the quality of the nodes. But for Go same approach won't work because of high branching factor and difficulties to design good evaluation function. Alternative approach to searching the game tree is to use Monte Carlo Tree Search (MCTS) which run many game simulations, store some values, such as how often each node has been visited, and how often this has led to a win, and use stored values in next simulations. Many Go AIs uses MCTS algorithm, but only the best ones achieve strong amateur play level. AlphaGo introduced new approach.

AlphaGo makes extensive use of machine learning, the art and science of designing computer algorithms that learn from data. In supervised learning (a standard setting in machine learning) an algorithm is repeatedly presented with training examples along with their corresponding labels. A training example could be the game-state of a game of Go, and the training label would be if this state ultimately led to a win or a loss for the current player. The goal is to learn a model which is able to generalize well on previously unseen examples, it should be good at predicting the outcome of unseen Go games.

One class of models in machine learning is artificial neural networks, which represented by number of layers, each of layers contain parameters whose need to be trained. Each layer in an artificial neural network contains artificial neurons. Each neuron receives as input the outputs of neurons in a previous layer. The inputs are then summed together. Convolutional networks are a sub-type of neural network that are especially well adapted for the processing of image data. AlphaGo uses convolutional networks, which takes as input images with game state.

The AlphaGo uses next deep neural networks: one 'value networks' to evaluate board positions and three 'policy networks' to select moves. The value network provides an estimate of the *value* (probability of a win) of the current state of the game. The policy networks provide guidance regarding which action to choose, given the current state of the game. The output is a probability value for each possible legal move. Moves with higher probability values have a higher chance of leading to a win.

First policy network, was trained with 30 million positions, and had accuracy 57% and response time 3 ms. To get result faster was trained second smaller network with accuracy 24.2% but with response time 2 ms. Third policy network was trained with reinforcement learning: current policy network played with random previous iteration of policy network. Value network was trained again with reinforcement learning on 30 million game positions obtained while the policy network played against itself.

AlphaGo also introduced a new search algorithm that combines Monte Carlo simulation with value and policy networks. AlphaGo uses an evaluation function to give a value estimate of a given game state. AlphaGo uses a mixture of the output of the value network and the result of a self-play simulation of the fast policy network: value of a state = value network output + simulation result. The slow policy network is also used to guide the tree search.

As result AlphaGo achieved 99.8% winning rate against other Go programs and defeated the human European Go champion by 5 games to 0. What is very interesting about AlphaGo is that it involves a learning component instead of hand-crafted heuristics. By playing against itself, AlphaGo automatically gets better and better at playing Go. This provides hope that AlphaGo's approach might adapt well to other AI problems.