AlphaGo Synopsis

An analysis of AlphaGo paper by the DeepMind Team.

Summary

The AlphaGo game playing agent seems to use a technique which involves search tree heuristics in combination with convolutional deep learning techniques. Because exhaustively searching all the nodes of a tree in a Go board would be infeasible due to the sheer size of possibilities a novel way of filtering the “best” tree paths for evaluation needed to be invented. This “filter” is known as “policy networks”. These policy networks are trained using deep convolutional networks to reduce the effective depth and breadth of the search tree.

Neural Network Architecture

The policy networks are trained the same way that traditional images are trained in deep convolutional networks. Just as you would map each pixel in an image to a neural network node as an input - each position on the Go board was mapped onto a neural network node as a 19x19 image.

The initial training of the policy network seems to have begun as supervised training with expert human Go players. Supervised training involves training neural network with the “right” answers to specific scenarios. The more supervised examples the network sees, the better it becomes at predicting the best possible outcomes. The initial training set for this supervised learning stage begins from a data set of 30 million positions from the KGS Go Server. The network was designed as a 13-layer convolutional neural network with a final softmax layer that outputs a probability distribution over all legal moves given the current board state.

The second stage of the network involves reinforcement learning to give the neural network more “sample data” to play with. The reinforcement learning is generated through “self play” where the policy network plays against previous versions of itself and then records its outcome as a new data set. This network is identical in structure to the SL policy network (13-layer convolutional).

Monte Carlo Search + Neural Networks

These policy networks are used in combination with traditional Monte Carlo search trees in order to select actions by lookahead search. The policy networks provide a filtering mechanism

for leaf node expansion during search, thereby reducing the number of nodes that need to be visited.

Performance

Because evaluating policy and value networks requires several orders of magnitude more computation than traditional search heuristics, multiple threads (40) were used in combination with multiple CPUs (48) and GPUs (8). The policy and value networks were evaluated using GPUs since neural network computations perform much better as GPU tasks. Distributed versions of *AlphGo* were also created with much higher CPU and GPU numbers.

AlphaGo was run head to head against several other open source Go playing programs (such as *Pachi* and *Fuego*) with 5 second timeouts per move and was shown to be very effective at defeating it's opponents with ***99.8%*** win rates. *Pachi* and *Fuego* do not use any deep learning and rely solely on optimized Monte Carlo search algorithms.