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

Go is far more complex than chess in both average breadth and depth of the game tree at average breadth of 300 and depth of 150 vs 35 and 80 for chess respectively. Exhaustive search of game trees are implausible at the Go numbers above. So first search breadth is reduced by sampling from a policy that is a probability distribution of possible moves 'a' in position 's'. Monte Carlo rollouts are typically used in computer Go programs to reduce the breadth by sampling and averaging over these rollouts to provide effective position evaluation. In addition, AlphaGo uses Monte Carlo tree search(MCTS) that employs these rollouts to estimate the value of each state 's' above. More simulations are employed to grow the search tree in order to get more accurate values for each state 's'. The policy is improved over time by selecting children with higher values, to eventually converge to optimal policy. So far other Go programs using MCTS have used shallow policies or value networks based on a linear combination of input features.

AlphaGo uses deep convolutional neural networks to evaluate game positions using a value network and sampling actions using a policy network. First a supervised learning(SL) network is trained from expert human moves. In addition a second fast policy is trained that can rapidly sample actions during rollouts. Next a reinforcement learning(RL) policy network is used to improve the SL policy network by optimizing the final outcomes of games of self-play that are played to the end rather than partly as earlier with other programs to simply maximize predictive accuracy.

SL policy network used was 13-layer from 30 million positions from the KGS Go server, predicting expert moves with accuracy of 57% vs 44.4% for the next best available Go program. The second stage RL is identical to SL but uses a randomized pool of opponents each time by using a randomly selected previous iteration of the policy network to avoid overfitting. The final stage of the training pipeline focuses on position evaluation estimating values that predict the outcomes from position 's ' of games by using policy 'p'. The weights used by the value network are trained by regressing state-outcome pairs to reduce mean squared error(MSE) between predicted value and corresponding outcome. To avoid overfitting a new training data set of 30 million distinct positions are used each sampled from a separate game. Training on this new data set lead to very low MSE of 0.226 and 0.234 vs 0.37 previously. In addition, computation was reduced 15000 times than Monte Carlo rollout without sacrificing accuracy.

Thus AlphaGo combines the policy and value networks in an MCTS algorithm that selects actions by lookahead search, descending the tree to complete games without lookup. Action value is set for each action selected from state 's' with bonus added to values that is proportional to prior probability but decays with repeated visits to encourage exploration. AlphaGo also used parallelism in 48 CPUs for executing simulations and 8 GPUs to compute policy and value networks.

RESULTS:

AlphaGo is many dans stronger than any prior Go program winning 494 out of 495 games against other Go programs and between 77 – 99% of games even with 4 handicap stones against Crazy Stone, Zen and Pachi. AlphaGo was evaluates against Fan Gui, the European Go champion and beat him 5-0, a feat previously believed to be a decade away! One of AIs grand challenges to play at human level was thus achieved by employing deep neural networks trained by a novel combination of supervised and reinforcement learning described above.