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

Goals and techniques

The goal of the paper is to provide an AI approach to the game Go, which has been viewed as, "the most challenging of classic games for artificial intelligence owing to its enormous search space and difficulty of evaluating board positions and moves." To overcome the complexity of the game ($b^d = 250^{150}$) authors of the paper trained multiple neural networks using a pipeline of several machine learning stages. The pipeline included a supervised learning policy network directly from expert human moves, a fast policy that rapidly sampled actions during rollouts, a reinforcement learning policy network, and a value network. The program combined the policy and value networks with Monte Carlo tree search.

The supervised learning for policy networks ended with a softmax layer output with a probability distribution over all legal moves *a* given a state *s* using stochastic gradient descent to maximize the likelihood of the human move *a* selected in state *s*. The paper found that small accuracy improvements, "led to large improvements in playing strength." The next step, improved the policy network by policy gradient reinforcement. This network had identical structure as the previous. The magic happens when games are played between the current policy network and a randomly selected previous iteration of the policy network. This randomization prevents overfitting to the previous network. The second, reinforcement policy network one more than 80% of the games against the first supervised learning network.

Position evaluation is the final step in the pipeline. It uses a value function *v* to predict the outcome of a position of games played by both players. This network again uses similar structure to the previous networks, but outputs a single value instead of a probability distribution. This network had a problem of overfitting due to predicting game outcomes from data that consisted of complete games. To overcome overfitting, the team generated a self-play dataset played between the previous network and the third network until termination. The policy and value networks were combined with a Monte Carlo Search Tree.

Results

Variants of AlphaGO and several other Go programs were run in an internal tournament to evaluate its strength. The results indicated that AlphaGo is many *dan* ranks stronger than other Go programs at close to a 100%-win rate. A distributed version of the program was able to beat its components 100% of the time with a four stone handicap.