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

Silver et al (2016) *Nature 529*, 484-489.

In this paper, Silver et al from Google DeepMind team introduce a novel approach of developing a Go playing AI, called AlphaGo. Many other perfect information games have been mastered by AI already, such as chess and checkers. However, Go still remained elusive because of the great complexity; the breadth and depth of Go's possible game tree is approximately 250 and 150, respectively, compared to 35 and 80 for chess (that would mean approximately 250^150 nodes!). While other Go programs were playing at beginner or amateur level, AlphaGo was able to play at professional level. AlphaGo defeated other Go programs with 99.8% winning rate, and defeated the professional Go champion Fan Hui 5-0 (and would go on to defeat Sedol Lee in March 2016), previously thought to be unachievable for few more decades.

To achieve this historical feat, AlphaGo combined three techniques: policy network, value network, and rollouts. First, a policy network was developed by training the AI to predict a professional action (placing of a stone) given a board state in a dataset from the KGS Go Server. This part of the training was done using supervised learning with convolutional neural networks. Then this policy network was enhanced by reinforcement learning, playing against a random choice from previous versions of the policies. Rewards of +1 for winning and -1 for losing were used, and weights were backpropagated using stochastic gradient ascent of expected outcome.

Second component is the value network, which estimates the value function for an action taken at a given board state---in simple terms, how likely the action will lead to a win or a loss. Because of the complexity of Go, approximation is used. Reinforcement learning was used by playing against the policy network, backpropogating the weights using gradient descent of the mean squared error between predicted value and the outcome.

Lastly, rollout involves a depth search where a complete game is played out without backup, a version of Monte Carlo tree search, technique previously developed into latest Go programs. Several of the paths are taken, using the action value with discount for number of visits, so that exploration is encouraged. After a given amount of time, most visited node is chosen as the action.

Although these individual parts performed already well, at a level comparable to or better than other Go programs, it performed much better with all three components. Policy network worked like knowing general good moves for a given state, whereas value network evaluated those moves more accurately, with rollout combining the two quickly. Because of the complex architecture of AlphaGo, it required orders of magnitude more computational power, with final version of AlphaGo having 40 search threads, 48 CPUs, and 8 GPUs. There is also a distributed version of AlphaGo, which improves the performance even further. This demonstration has given hope that more domains can be mastered by AI, or fear for Terminator fanatics.