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

The DeepMind team attempted to create superhuman levels of play in the game of Go with the advent of AlphaGo. Previous agents developed to tackle this game made significant progress but fell short when playing against professionals without handicaps. The approach makes use of Monte Carlo Tree Search; a heuristic that randomly sample leaves recursively until an outcome is achieved to estimate the value of the root. The information of the outcome is propagated back to the root to represent the historical win ratio for each player of all simulations. This approach has often been used in the past with pre-defined value functions but have gained little traction in Go, due to its larger branching factor and depth in comparison to Chess. The DeepMind team augment this search strategy with neural networks to learn functions that make up AlphaGo. A network is constructed for predicting expert moves, learning desired outcomes from these predictions through repetitive self-play, and to learn a value function to accurately represent the utility from the simulated outcomes.

The supervised network policy uses convolutional neural networks with nonlinear rectifier units which produces a distribution of all possible moves from a current state. Based on a historical dataset of games on the KGS Go server, a position's frequency of next moves is used to create a resulting probability distribution from the maximum likelihood estimate of observed subsequent moves. The weights are tuned using stochastic gradient ascent and achieved an accuracy of 57% when predicting what moves a professional would play. A less accurate rollout policy was trained in conjunction that predicts moves at less than half the accuracy, but is 1,500 times quicker at selecting moves.

The network of predicting expert moves is further built upon through repetitive self-play against multiple generations of the supervised model to avoid overfitting to the most up to date policy. This training occurs under the framework of reinforcement learning; where weights are tuned in response to a reward function associated to moves taken. Post training, the RL network won 80% of games when playing against the most recent supervised policy, and also achieved similar win rates against the strongest pre-existing bot at the time.

Finally, the value of each position on the board is learned through a neural network that is based on the RL policy network. The architecture is similar to the policy networks but is designed to output a predicted value to represent how good the move is, trained using regression between the states as variables and the subsequent outcomes (+1 for win / -1 for loss). Initial attempts to learn the value resulted in the network memorizing the outcomes, creating a disparity between the error rate between train and test, strongly suggesting overfitting. This issue was resolved by creating additional game datasets from self-play, retraining led to achieving lower error on the test set and an almost equivalent training error.

These components are combined to store at each node a learned value, a distribution of future moves, and a visit count. The Monte Carlo Tree Search heuristic takes these inputs and selects moves based on a weighted average of the value function and an outcome from sampling a terminal outcome from the node, some noise to allow room for exploring less common moves. The value of the action and the visit counts are updated post-simulation, and the most frequently selected move from the root is chosen. In a tournament against the strongest bots, the final iteration of Alpha Go achieved over 99%-win rate. The bot went on to subsequently win against European champion Fan Hui 5-0 without a handicap in 2015, again in 2016 versus the second highest ranked player in the world Lee Sedol 4-1, and finally the highest ranked player Ke Jie 3-0 in 2017.