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

By Bryan Travis Smith

David Silver et. al. discuss the high level process of how they developed AlphaGo in their Nature article, *Mastering the game of Go with deep neural networks and tree search*. They describe the use using a novel combination of Deep Neural Network, Reinforcement Learning, and Regression to develop policy networks and value network used in Monte Carlo Tree Search to estimate the value of a given state in the tree. This work represents the use and power of combining deep neural networks, reinforcement learning, and tree search. The result of this work was the (5, 0) defeat of a professional Go player.

The AlphaGo team started by building a 13 layer neural network that produced a probability distribution over all legal moves using data from a dataset of 30 million positions form the KGS Go Server. This supervised learning model was then used to produce a policy network of optimal actions for a given state. The team then performed policy update through self game play on this policy network using stochastic gradient decent. As the policy improved, it would play against randomized pools of previous versions of the policy network to avoid overfitting against itself. Finally, a value function was generated by performing a 13 layer neural network regression to build a fast value function. This required the team to build a dataset of game-states randomly sampled from distinct games in self-play to avoid overfitting.

The value network and policy network were used in the Monte Carlo Tree search by using them in a value function that took a weighed sum of the predicted value of a state from the value network and the terminal state by playing the current state to end game with both players playing the policy network strategy. This value is used to calculate the action value Q(s,a) that produces the value of an action a for a given state s.

The Monte Carlos process explore the tree by explore a node by following the action with the largest value given from: $Q(s,a) + P(s,a)/(1 + N(s,a))$. $P(s,a)$ is the prior on the state, action pair, and $N(s,a)$ is the number of times this state-action pair has been visited. The second term is used to promote exploration of new actions.

This new methodology lead to the surprising result of AlphaGo far exceeded any other Go AI Agent to date. In a tournament against the leading proprietary and open source agents, the AlphaGo program won 99.8% of the time. A distributed version of the program, with greater computational capacity, won 100% of the time. This also lead to the startling first of a Go AI agent beating a professional Go player.