Research Review

This research review summarises the paper "Mastering the game of Go with deep neural networks and tree search". This paper presents a new way (AlphaGo) that enables computer to play the game of Go at professional human player level by combining deep neural networks and Monte Carlo tree search (MCTS).

Before this paper, MCTS, together with policies trained to predict human expert moves so as to reduce search space, have been used to create Go programs that could achieve strong amateur play. However, the prior work was limited to shallow policies or valuation based on a linear combination of input features.

The authors of this page used deep convolutional neural network to represent positions. A value network was used to evaluate positions and a policy network was used to sample actions. The training of the neural networks were implemented using a pipeline of several stages: supervised learning of policy networks, reinforcement learning of policy networks, and reinforcement learning of value networks.

At the stage of supervised learning (SL) of policy networks, a policy network $p_\sigma(a|s)$ was trained to present the probabilities of all legal move a when at game state s. Randomly sampled state-action pairs (s,a) and stochastic gradient ascent were used to train this network to maximize the likelihood of move a selected at state s by human. A less accurate rollout policy $p_\pi(a|s)$ was also trained using small pattern features. It presented a lower accuracy but spent only 2μs selecting an action instead of 3ms by the policy network.

During reinforcement learning of policy networks, policy gradient reinforcement learning (RL) was used to improve the policy network. An RL policy network p_ρ with the identical structure as the SL policy network trained at last stage was used. It was initialized with the same weights as SL policy network. Games were played between policy network p_ρ and a randomly selected previous iteration of the policy network. +1 is the reward for winning and -1 is for losing. At the end of the games, weights were updated at each time step by stochastic gradient ascent to maximize expected outcome.

The stage of reinforcement learning of value networks focused on position evaluation. Assuming that the game was played by both players using policy *p*, it tried to estimate a value function $v^p(s)$ which predicts the outcome of position *s.* The value function was approximated using a value network $v_\theta(s)$ which is weighted by θ and has a similar architecture to the policy network p_ρ. The value network was trained using a self-play data set that each position was sampled from a separate game so as to mitigate the overfitting problem. During the training, the weights were trained by regression on state-outcome pairs (s,z). Stochastic gradient descent was used to minimize the mean squared error between predicted value $v_\theta(s)$ and the corresponding outcome z.

After the neural networks were trained, the authors of this page combined policy and value networks in an MCTS algorithm. The tree was traversed by simulation. In each simulation, the edge with maximum action value Q plus a bonus $u(P)$ was selected. The bonus value was proportional to the prior probability but decayed by edge visit count. When the traversal reached a leaf node, the leaf node may be expanded and processed by policy network p_σ. The output probabilities from p_σ for each legal action $p_\sigma(a|s)$ are stored as prior probabilities. The leaf node was evaluated by the mix of value network $v_\theta(s)$ and the outcome of a random rollout played out until terminal step using rollout policy p_π. When the simulation ended, all traversed edges were updated with action values and visit counts.

By combining deep neural networks and Monte Carlo tree search, AlphaGo outperformed other Go programs, including Crazy Stone, Zen, Pachi and Fuego, by winning 494 out of 495 games. In the games with four handicap stones, AlphaGo won 77%, 86% and 99% of games against Crazy Stone, Zen and Pachi. The distributed version of AlphaGo was also used to play games against European Go champion and won the matches 5 games to 0.