Paper Summary for Project 2, AIND

Hossein Seyedmehdi

The game of Go is one the most challenging games for artificial intelligence (AI) due to the intractability large search space. In [1], a novel method is proposed by combining neural networks and decision search trees that enables the AI agent (AlphaGo) to win against human experts with a 5 to 0 winning rate and outperform pervious AI agents.

Due to the large search space (which is approximated to be in the order of 250^{150}), exhaustive search for the game of Go is infeasible. There are two approaches to reduce the size of search space: firstly, truncating the search tree at state s and replacing the tree below with a value function $v(s)$; secondly, pruning the breadth of the search by sampling actions from a policy set $p(a|s)$ which is the probability of an action a given the state s. In this paper, authors have achieved these goals by training a deep neural network for value functions (value network) and another deep neural network for policies (policy network).

The policy network is trained via supervised learning (SL) in the first stage and then reinforcement learning (RL) in the second stage. The SL policy network is comprised of a convolutional neural network with weights σ and non-linear rectifiers. In a randomly selected state s, a stochastic gradient descent is used to maximize the likelihood of human expert move a, $p_\sigma(a|s)$. Doing so, a 13-layer deep neural network is trained using available KGS Go database. In the second stage of the training, a RL is used to further tune the policy network. The weights ρ for the RL policy network are first initialized with the weights from the SL network, σ, and then, policy network p_ρ is played against a randomly selected previous iteration of the policy network. In this paper, the RL policy network is shown to win more than 80% of the games against the SL network.

The value network is trained by reinforcement learning in this work. For the value network (with weights θ) a similar convolution structure to that of the policy network is used. This value network $v_\theta(s)$ is aimed to predict the outcome of the game at state s. To train the weights of this network, the mean squared error (MSE) between the predicted value, $v_\theta(s)$, and the outcome z is minimized using the stochastic gradient descent. It was first observed that only using the available data set, the network tends to be overfitted (which basically means that the network memorizes the outcomes of each state rather than generalizing to new states). To alleviate the overfitting problem, the authors generated a large number of random game states and trained the value networks by using the results from the RL policy networks.

The policy and value networks are combined in a Monte Carlo tree search in AlphaGo. In this search tree, the value of each leaf node s_L is calculated by mixing the value network $v_\theta(s_L)$ and the outcome z_L of a fast rollout play using the policy network. The objective function to select the action a_t decays with the number of visits to a particular state to encourage exploration. The authors have mentioned that evaluating policy and value networks requires significantly more search power than conventional methods. AlphaGo uses 40 search threads, 48 CPUs and 8 GPUs in the centralized version and 1,202 CPUs and 176 GPUs in the distributed version.

When AlphaGo was played against other Go programs and human experts, it showed superior performance. The single machine AlphaGo has a winning rate of 99.8% when played against other programs. Further evaluations using only value network or only the fast rollout policy network in the decision tree shows that these mechanisms are complementary to each other.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.