Research review for alphago

This paper is to introduce a new approach to master the game of go combine with deep neural networks and tree search. It introduce two networks, the ‘value networks’ to evaluate board positions and the ‘policy networks’ to select the moves. Two neural network trained by human expert moves by using supervise learning then use reinforcement learning to self-play to improve the performance. By combining Monte Carlo simulation with value network and policy network, the program archive the state-of-art level and even beat the human professional player in the full-sized game of go.
Lots of game many be solved by recursively computing the optimal value function in the search tree. But for a large game like Go, it is infeasible by exhaustive search. The effective search space can be reduce by two general principles. First, the depth of the search may be reduced by position evaluation truncating the search tree and approximating the v(s) almost equal optimal value v*(s) that predict the outcome from the state. Second, the breadth of the search may be reduce by sampling action from a policy p(a|s) that is the probability distribution over possible moves a in position s. But the prior work been limited to shallow policies or value value functions based on a linear combination of input features due to the complexity. It use the same structure of convolutional neural network to represent the position as a 19 X 19 image. It also use these neural network to reduce the effective depth and breadth of the search tree. It trained the neural network using a pipeline consisting of several stages of machine learning. They begin by training a supervise learning network directly from human expert moves. Then using reinforcement learning to the SL network performance. Finally, train a value networks that predicts the winner of games played by the RL policy network against itself. This efficiently combines the policy and value networks with MCTS.

Supervised learning of policy networks

The SL policy network consist with

- 13 layers of convolutional network
- RELU activation function
- Final layer softmax output the action distribution.

It use the state-action paring data as training set using stochastic gradient ascent to maximize the likelihood of human move a selected n state S. The gradient is:

$$\Delta\sigma \propto \frac{\partial \log p_\sigma(a|s)}{\partial \sigma}$$

It already achieve state-of-art according its report comparing with previous best model.

Reinforcement learning of policy networks

This stage goal is to improving the policy network by policy gradient reinforcement learning.

- The parameter initiate as the same value as previous stage.
- Play game between current network and randomly selected previous iteration of policy network.
- Use r(s) that is zero for all non-terminal time steps t<T

- The final result is +1 for winning and -1 for losing
- Weight updated at each time step t by SGA maximize the outcome

$$\Delta\rho \propto \frac{\partial \log p_\rho(a_t|s_t)}{\partial \rho} z_t$$

After evaluated the performance of RL policy network. It wins 80% over the previous policy network. And beats the strongest open-source Go program, and 2 dan human player.

Reinforcement learning of value networks

The final stage is to evaluate the position value. The difference of this network is output a scalar value. It use the MSE as loss function.

$$\Delta\theta \propto \frac{\partial v_\theta(s)}{\partial \theta}(z - v_\theta(s))$$

This method is 15,000 times less computation then rollouts approach.

Searching with policy and value networks

The alphaGo combines the policy and value networks in an MCTS algorithm that selects actions by lookahead search that is simulation traverses the tree by selecting the edge with maximum action value Q plus a bonus u(P).

$$a_t = \underset{a}{\operatorname{argmax}}(Q(s_t, a) + u(s_t, a))$$

$$u(s, a) \propto \frac{P(s, a)}{1 + N(s, a)}$$

This formula is proportional to the prior probability but decays with repeated visits to encourage exploration.

The leaf node is evaluated in two very different ways: The first part is value network and second is value come from fast rollout policy network. these evaluations are combined, using a mixing parameter lambda as weight.

$$V(s_L) = (1 - \lambda)v_\theta(s_L) + \lambda z_L$$

At the end of the simulation, action values is combine visit count and mean evaluation of all simulation passing through that edge:

$$N(s, a) = \sum_{i=1}^{n} 1(s, a, i)$$

$$Q(s, a) = \frac{1}{N(s, a)} \sum_{i=1}^{n} 1(s, a, i) V(s_L^i)$$

Evaluating the playing strength of AlphaGo

According to the tournament evaluation, Single machine AlphaGo has been achieved the professional 2 dan and more surpass than any open source and commercial Go program. The distributed AlphaGo performance is more better than single machine and achieve about 4.5 dan level. The tournament evaluation also compared the power with different network combination and different threads and hardware.

Discussion

They have developed the Go program, base on tree-search and neural network that achieving one of artificial intelligence's "grad challenges". It introduce a new search algorithm that successfully combines neural network evaluation with Monte Carlo rollouts. Comparing with deep blue, it evaluated thousands of times fewer position. The deep blue relied on a handcrafted evaluation function, but AlphaGo relied on expert human data and reinforcement learning improving. By combining tree search with policy and value networks, AlphaGo has finally reached a professional level in Go, providing top that human-level performance can now be achieved in other seemingly intractable artificial intelligence domains.