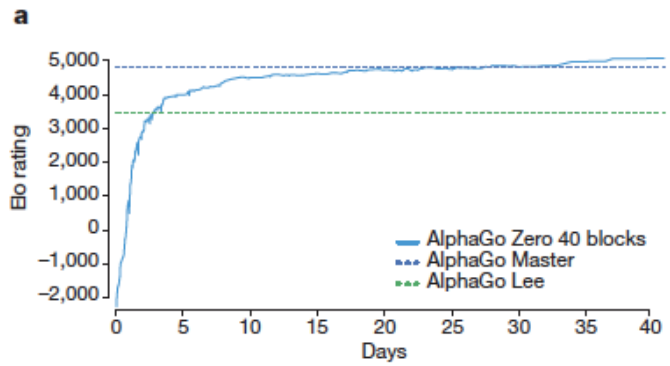
Student Name: Nikhil Doifode
Student ID#: 112714121

Review of "Mastering the game of Go without human knowledge"

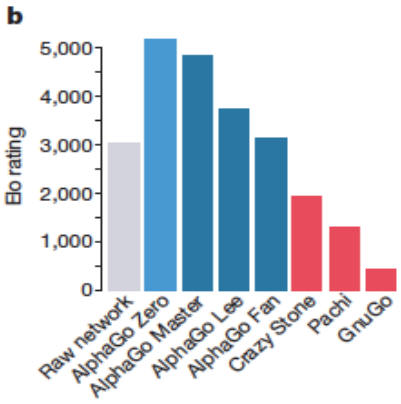
Paper published by DeepMind about the Mastering the game of Go without human knowledge is about the new version of AlphaGo called AlphaGo Zero. AlphaGo Zero was released to showcase that Artificial Intelligence can be used to solve more complex problems solely by self-learning reinforcement learning without any supervision or use of human data. This paper showcases the advancement made on the original AlphaGo in Computational Requirements, time taken to achieve the level of expertise and the most important to not be bounded by the input data from human experts.

All the supervised learning systems developed in the field of Artificial Intelligence which rely on the training to replicate decisions of human experts are dependent on the human experts datasets which sometimes is unavailable, expensive or unreliable and are also bounded by the level of performance they can achieve. That's why, the sole reinforcement learning method used in AlphaGo Zero makes it better than AlphaGo in terms of performance. AlphaGo was trained using the data collected by the human experts to train the Convolution Neural Networks and then use reinforcement learning to train itself based on that data. This process took months for AlphaGo to reach the level of expertise. However, AlphaGo Zero doesn't require any such kind of data. It only requires the board sketch, the rules of the game and the two types game pieces used i.e. Black and White.

In the performance part, Learning curve for AlphaGo Zero using a larger 40-block residual network over 40 days. The following plot shows the performance of each player from each iteration of the reinforcement learning algorithm. Clearly the learning curvy of AlphaGo Zero start from zero but overtakes the AlphaGo Lee on Day 4 and overtake AlphaGo Master on Day 30 and continue to become better.



AlphaGo Zero was trained for 40 days using a 40-block residual neural network. The following plot shows the results of a tournament between AlphaGo Zero, AlphaGo Master, AlphaGo Lee, AlphaGo Fan, as well as previous Go programs Crazy Stone, Pachi and GnuGo.



Each program was given 5s of thinking time per move. AlphaGo Zero and AlphaGo Master played on a single machine on the Google Cloud; AlphaGo Fan and AlphaGo Lee were distributed over many machines. The raw neural network from AlphaGo Zero is also included, which directly selects the move a with maximum probability p_a, without using MCTS. A 200-point gap corresponds to a 75% probability of winning therefore In a 100-game match with 2-h time controls. AlphaGo Zero won by 89 games to 11.

Strengths:

1. AlphaGo Zero uses no human knowledge and learns only from self-play.
2. AlphaGo Zero uses only single neural network with ResNets structure which serves the purpose to decide next move and evaluate chances of winning
3. AlphaGo Zero has simpler Monte Carlo Tree Search (MCTS)
4. It doesn't require any handcrafted features, only requires raw board representations and its history, plus some basic game rules as neural network input
5. Improved computation efficiency by using only single machine of google cloud with 4 TPU's

Weakness:

- AlphaGo Zero is a very narrow application of AI and it relies on many properties of Go. The paper misses the opportunity to identify other broad applications of the algorithm so that it can be generalized

Questions:

1. What is the difference between AlphaGo Zero and AlphaGo Lee?

Ans: Differences between AlphaGo Zero and AlphaGo Lee (generally called as AlphaGo only) are as follows:

a. Alpha Go Zero is trained solely by self-play reinforcement learning, starting from random play, without any supervision or use of human data, whereas AlphaGo Lee required large data-sets of expert games
b. AlphaGo zero uses the black and white stones from the board and rules of games as input features, whereas AlphaGo Lee uses a few handcrafted features for it's working.

c. AlphaGo Zero uses a single neural network which requires less computation power, whereas AlphaGo Lee has separate policy and value neural networks which requires more computation power.
d. AlphaGo Zero uses a simpler tree search that relies upon the above mentioned single neural network to evaluate positions and sample moves, without performing any Monte Carlo rollouts. However, in AlphaGo Lee has complex Monte Carlo Tree Search to combine policy networks output and value network output to select actions lookahead search.

2. How does AlphaGo Zero learn by playing with itself?

Ans: AlphaGo Zero learns by playing with itself using reinforcement learning. It uses a deep neural network f_θ. This neural network combines the roles of both policy network and value network into a single architecture as done in AlphaGo. This neural network takes as an input the raw board representation s of the position and its history, and outputs both move probabilities and a value.

$$(p, v) = f_\theta(s)$$

The p represents the probability of selecting each move. The value v is estimation of the probability of the current player winning from position.

After this, in each position s, an Monte Carlo Tree Search is executed, guided by the neural network. The MCTS search outputs probabilities π of playing each move. These search probabilities usually select much stronger moves than the raw move probabilities p of the neural network.

The reinforcement learning algorithm uses these search operators repeatedly in a policy iteration procedure. The neural network's parameters are updated to make the move probabilities and value more closely match the improved search probabilities and self-play winner (π, z). These new parameters are used in the next iteration of self-play to make the search even stronger. Following Figure illustrates the self-play reinforcement learning in AlphaGo Zero.

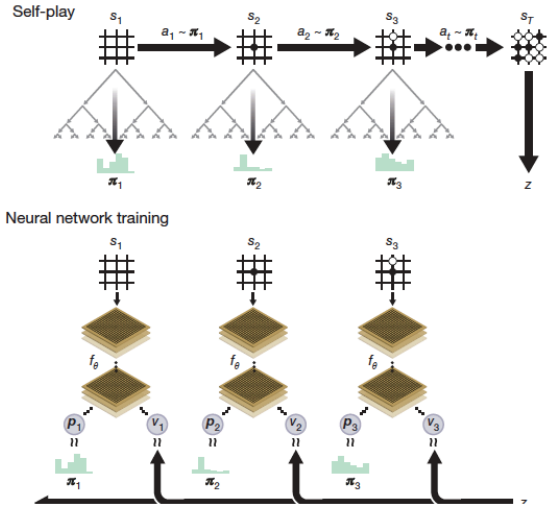


Fig: Self-play reinforcement learning in AlphaGo Zero

3. How does the Monte-Carlo tree search work in AlphaGo Zero?

Ans: In each position of s, a Monte-Carlo Tree Search (MCTS) is executed, which outputs probabilities Π for each move. These probabilities usually select much stronger moves than the raw move probabilities 'p' of the policy head on its own.

MCTS may be viewed as a powerful policy improvement operator. Self-play with search using the improved MCTS based policy to select each move, then using the game winner 'z' as a sample of the value may be viewed as a powerful policy evaluation operator. The main idea of the reinforcement learning algorithm is to use these search operators repeatedly in a policy iteration procedure, the neural network's parameters are updated to make the move probabilities and value

$$(p, v) = f_\theta(s)$$

more closely match the improved search probabilities and self-play winner (Π, z); these new parameters are used in the next iteration of self-play to make the search even stronger.

The entire network is initialized to random weights, and then in each subsequent iteration self-play games are generated. At each time step an MCTS search is executed using the previous iteration of the network, and a move is played by sampling the search probabilities Π_t. A game terminates when both players pass, when the search value drops below a resignation threshold, or when the game exceeds a maximum length. The game is then scored to give a final reward.

$$(p, v) = f_\theta(s)$$

The neural network is adjusted to minimize the error between the predicted value v and the self-play winner z, and to maximize the similarity of the neural network move probabilities p to the search probabilities Π. Specifically, the parameters θ are adjusted by gradient descent on a loss function l that sums over mean-squared error and cross-entropy losses:

$$l = (z - v)^2 - \Pi^t \log(p) + c\,||\theta||^2$$

Following Figure illustrates the Monte Carlo Tree Search in AlphaGo Zero.

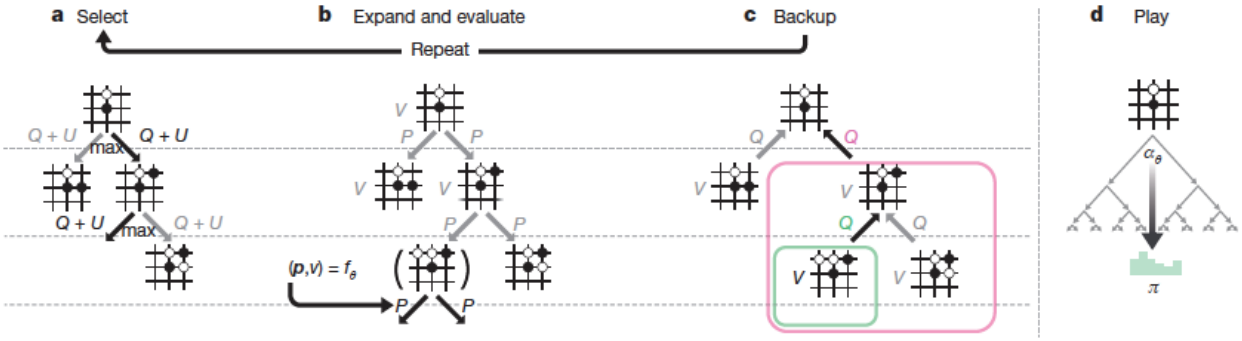


Fig: MCTS is AlphaGo Zero

References:

Mastering the game of Go without human knowledge – David Silver,…