Alphago Paper Review

Ihab Sultan

March 4, 2017

Paper goals
Although Go is a zero-sum 2 player game with perfect information, it is considered a very complex game for AI due its huge search space. Go branching level is averaged at 250, and has an average depth of 150 steps, making exhaustive search intractable This paper explains methods into which the team at Deepmind built a Go playing agent and tested it against state-of-the-art go implementations, and the human European go champion.

Techniques To achieve the goal, the following models were built:

1. **SL policy network** $[p_\sigma(a|s)]$:
   CNN model trained to maximize the likelihood of human moves
   - **input [s]:** board state represented by 48 features per location (Table 2).
   - **output [a]:** softmax layer produces a probability distribution over all legal moves.
   - **model size:** 13 layers (13 ms per evaluation)
   - **training:** Supervised learning on 30 million positions from KGS Go Server.
   - **model accuracy:** 57%
2. **Rollout policy network** $[p_\pi(a|s)]$:
   Trained to maximize the likelihood of human moves
   - **input [s]:** 6 binary features per location representing different Go patterns (Tablel 4).
   - **output [a]:** softmax layer produces a probability distribution over all legal moves.
   - **model size:** 1 layer (2 μs per evaluation)
   - **training:** Supervised learning on 30 million positions from KGS Go Server.
   - **model accuracy:** 24.2%

3. **RL policy network** [$p_\rho(a|s)$]:
   Trained to maximize the probability of actions that lead to a win

   - **input [s]:** Same as SL policy network.
   - **output [a]:** softmax layer produces a probability distribution over all legal moves.
   - **model size:** Same as SL policy network.
   - **training:** Model is initialized with $\rho = \sigma$ followed by reinforcement learning using policy gradient on games between current network and randomly selected previous iterations.
   - **model performance:** Won 85% of the games against Pachi.

4. **Value network** [$v_\theta(s)$]:
   Trained to accurately estimate the value of the game

   - **input [s]:** Same as SL policy network.
   - **output [v]:** regression head produces a value of the game.
   - **model size:** Similar to SL policy network.
   - **training:** Regression on state-outcome pairs (s,z) on games between current RL policy network and itself.
   - **model performance:** MSE of 0.234

   Policy and value networks are combined with MCTS algorithm as follows:

   (a) **Selection**: Start from the root of the current search tree, and follow actions $a_t = \text{argmax}_a(Q(s_t, a) + u(s_t, a))$ until a leaf is reached at level L.

   (b) **Evaluation**: Calculate $v_\theta(s_L)$ using value network. In addition, get a sample game outcome by following rollout policy ($p_\pi(.|s_t)$) till the end of the game.

   (c) **Backup**: For each visited edge at $t \leq L$, update the number of rollouts, and the rollout value according to game outcome. Update the same edges by adding $v_\theta(s_L)$ to the total action value.

   (d) **Expansion**: Add a new node to search tree once visit count of its preceding edge exceeds a threshold.

Results
AlphaGo competed against state of the art computer programs (Crazy Stone, Zen and Pachi), results were as follows:

1. Single machine AlphaGo won 494 of 495 games
2. AlphaGo won 77%, 86%, and 99% of handicap games (by 4 pieces) against Crazy Stone, Zen and Pachi, respectively.

In addition, Distributed version of AlphaGo played against Fan Hui, the winner of 2013, 2014 and 2015 European Go championships. AlphaGo won the match 5 games to 0.