Research Review

Table of Contents

Research Review: Game Tree Searching by Min / Max Approximation

Research Review: Deep Blue

Research Review: Mastering the game of Go with deep neural networks and tree search

Research Review: Human-level control through deep reinforcement learning

Author: Alexander Ponamarev alex.ponamryov@gmail.com

Research Review: Game Tree Searching by Min / Max Approximation

Goals

Existing techniques help to reduce computational burden of exploring a game tree. However, do not intentionally expand the node that is expected to have the largest effect. Therefore, this paper is introducing a method to solve this problem.

Technique introduced

Paper introduces “min / max approximation” method intended to focus the computer’s attention on the important lines of play. The key idea of the method is to:

1) Approximate min and max methods with a single generalized mean-value operator that have continues derivative.
2) Use derivative of min / max approximation to choose which node to expand.

Method Description (Python):

```
def M(A, p=2):
    A = list(map(lambda a: a**p, A))
    Mp = sum(A)/len(A)
    Mp = Mp**(1/p) if not p==0 else Mp
    return Mp
```

For large positive or negative values of p, Mp(a) is a good approximation to min and max functions respectively.

The derivative of M(A, p) will show changes of utility function as nodes of the state expand. Therefore, the nodes that demonstrate the higher improvement in utility are the best candidates to expand. However, it is not clear how to optimally choose the p parameter. Choosing high p value corresponds to having a high degree of confidence in the accuracy of utility function. Therefore, high p values lead to very deep but narrow trees. Whereas, p=1 grows the tree in a breadth-first search manner.

Results

Experimental results from almost 1,000 games of Connect-Four suggest that Min/Max Approximation is superior to minimax search with alpha-beta pruning. However, this method has higher overhead. Therefore, when CPU time is a limited resource minimax search with alpha-beta pruning seems to play better.

Author: Alexander Ponamarev alex.ponamryov@gmail.com

Research Review: Deep Blue

Goals

This paper describes the Deep Blue computer chess system. Deep Blue was developed in 1990's and it had several versions. The most successful versions competed against Garry Kasparov in 1996 and 1997. Deep Blue I was presented in 1996, and it lost to Garry Kasparov.

After this loss IBM team made additional changes to the system:
- Redesigned evaluation function, going from around 6400 features to over 8000
- Increased search speed from 1.6-2 million positions to 2-2.5 million positions per second for a single chess chip
- Double the number of chess chips

The search algorithm included the following techniques:
- Quiescence search
- Iterative deepening
- Transposition tables
- NegaScout

Technique introduced

This research paper provides a comparison of Parallel Search Performance. IBM Deep Blue team compared the performance of 24 chip system on a variety of positions against a single chip system. The results varied widely depending on the depth of the position searched. For positions with many deep forcing sequences speedups averaged about 7, for an observed efficiency of about 30%. For quieter positions, speedups averaged 18, for an observed efficiency of 75%.

Results

The success of Deep Blue in the 1997 match was not the result of any one factor. The large searching capability, non-uniform search, and complex evaluation function were all critical. However other factors also played a role, e.g., endgame databases, the extended book, and evaluation function tuning.

Author: Alexander Ponamarev alex.ponamryov@gmail.com

Research Review: Mastering the game of Go with deep neural networks and tree search

Goals

This paper introduces a new approach to computer Go that uses ‘value networks’ to evaluate board positions and ‘policy networks’ to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Using this search algorithm, program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0.

Technique introduced

Supervised Learning Policy Network (SL) - is a recurrent convolutional neural network that takes as an input a pair of board states and human expert actions and outputs soft-max probability distribution for all legal moves. The network is trained on randomly sampled state-action pairs using SGD to maximize the prediction of human move in a selected state. SL consists out of 13-layers. The network is trained on 30 million positions from KGS Go Server.

Reinforcement Learning Policy Network (RL) - is identical in structure to SL and it is initialized with the same weights. RL uses reward function that assigns reward (z) +1 for winning and -1 for losing the game. RL weights then updated at each time step by SGD in the direction that maximizes the reward.

Value Networks (VN) - is a network with a similar to SL and RL architecture that outputs a single value - of the reward (z). The network is trained using SGD to minimize MSE between the predicted value and the corresponding z reward.

Results

SL predicted expert moves on a held out test set with an accuracy of 57.0% using all input features, and 55.7% using only raw board position and move history as inputs, compared to the state-of-the-art from other research groups of 44.4% at date of submission.

RL policy network won more than 80% of games against the SL policy network. RL was also tested against Pachi (the strongest open-source Go program) and won 85% of the time.

VN prediction approach Monte Carlo rollouts accuracy using 15,000 less computation.

Author: Alexander Ponamarev alex.ponamryov@gmail.com

Research Review: Human-level control through deep reinforcement learning

Goals

This paper introduces a novel artificial agent, deep Q-network (DQN), that can learn policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning. DQN combines reinforcement learning with a class of artificial neural network known as deep neural networks.

Technique introduced

The goal of the agent is to select actions in a fashion that maximizes cumulative future reward. DQN uses a deep convolutional neural network to approximate the optimal action-value function after making observations and taking an action.

DQN training is based on experience replay technique and mini batch update. Experience replay randomizes over the data thereby removing correlations in the observation sequence and changes the data distribution. To perform experience replay, the agent stores experiences at each time-step in a data set.

Mini-batch updates assume that the weights are updated every C steps, while all experience being stored for the future experience replay.

DNQ implementation is available in Torch:
https://sites.google.com/a/deepmind.com/dqn/home/Human_Level_Control_through_Deep_Reinforcement_Learning.zip?attredirects=0&d=1

Results

To evaluate our DQN agent, we took advantage of the Atari 2600 platform. We compared DQN with the best performing methods from the reinforcement learning literature on the 49 games where results were available. In addition to the learned agents we also report scores for a professional human games tester playing under controlled conditions. DQN agent performed at a level that was comparable to that of a professional human games tester across the set of 49 games, achieving more than 75% of the human score on more than half of the games. Nevertheless, games demanding more temporally extended planning strategies still constitute a major challenge for all existing agents including DQN.

Author: Alexander Ponamarev alex.ponamryov@gmail.com