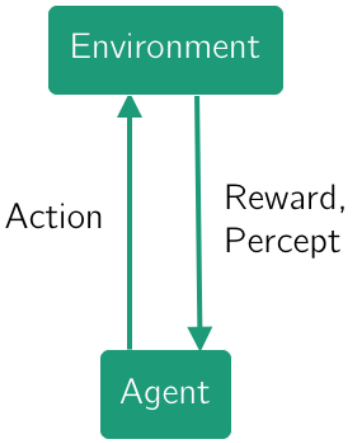
Deep Reinforcement Learning for Chess

David Hui

AI research is at a crossroads

- Symbolic vs Connectionist
- Is combination effective?
- Chess is a game with perfect information
- Win 95 % of 80 ply games against a random agent (Actual: 60%)



The main goal of AI is to implement an agent as shown. After receiving a percept and perhaps a reward from the environment, it hopefully returns a rational action.
Since the dawn of AI research since 1950s, the field has been split into two main subfields trying to achieve this.
Symbolic computation, specialising in algorithms which manipulate precise variables and classes and connectionist cybernetics, a field which enables agents to self-organise and discover their own representations.
A symbolic agent is Deep Blue, which exhaustively searches and generates trees for moves and a connectional agent is KnightCap, which chooses optimal representations, having learnt it by reinforcement learning.
Recent research has been able to combine the two approaches together, producing results such as AlphaGo.
To investigate this hybrid architecture, I decided to train an agent within the game of Chess, chosen because it has discrete time, space and perfect information – that is, it is possible to evaluate the favourability of the board purely by examining its current configuration.

Three Components of a Chess Engine

- Search algorithm
  - Minimax
- Training Algorithm
  - Reinforcement Learning
- Evaluation Function
  - Deep Network

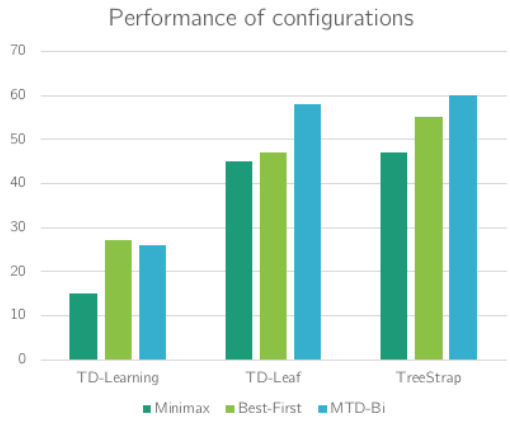
From classical symbolic AI comes a search algorithm. The class of algorithms which perform best in Chess are Minimax. Minimax algorithms simulate the future evolution of a game and selects a move which maximises what the opponent can minimise of a favourable board condition.

Favourability can be learned through Deep Reinforcement Learning. By playing against oneself, an agent can determine which chessboard states are conducive towards winning.
This is achieved through considering the sequence of boards played in the game. Reinforcement Learning increments the score of the board if it leads to victory and vice versa.

Scores are given by a Deep Neural Network. The network is used as a function approximator for board favourability.

TreeStrap and MTD-Bi strongest

- SUBJECT TO CHANGE
- Feedforward network used
- After 5000 self-play games
- From board position only



Overall, the best deep network structure found was a feedforward network, which was the easiest to train and the fastest to run.
Three searching algorithms and learning algorithms were implemented.

The best searching algorithm was MTD-Bi, which was better than Best-First and Minimax.

The best learning algorithm was TreeStrap, which was better than Temporal Difference Learning or TD-Leaf because the learning algorithm was able to search more nodes.

All the data came from self-play without the need for pre-training supervised learning.

DATA ON THE SLIDE SUBJECT TO CHANGE IN FUTURE EXPERIMENTATION

Performance can still be improved

- Implemented all Chess Engine components
- Wins 60 % of 80 ply games against a random agent (Target: 95 %)
- Investigate search algorithm
- Use adaptive network