Applying Deep Reinforcement Learning to finite state single player games
Donald Stephens (dsteph@stanford.edu)

Task Definition
Design an engine capable of solving any valid constructed Solitaire Chess Puzzle. The engine must be capable of performing better than a brute force approach by exploring less paths (sequence of states-actions).

Game Summary
Solitaire Chess is a single-player logic game utilizing the same rules of classical chess, but presented in a simplified form posed as mini Chess problems rather than full length opponent based strategy games.

Considerations
- All valid puzzles are designed to have a distinct winning piece. However, the sequence of actions need not be unique.
- A valid puzzle can contain multiple instances of any chess piece except for a King.

Evaluation Metrics
- The accuracy of identifying the intended final board piece and its final position. Must be 100% to be considered a viable model.
- The number of paths explored needed to identify a valid solution. Must be less than the average needed by baseline model.

Challenges
- Popular Chess solving strategies offer very little to no advantage to Solitaire Chess:
  - There are no opposing agents.
  - Every action must result in a capture.
- Increasing the number of Chess pieces of a puzzle increases the size of the solution search space.

Rules
- A single-player game.
- All of the Chess pieces are the same color.
- Pawns can be placed anywhere on the board.
- Pawns are not promoted and captured pieces cannot be restored.
- Kings cannot be placed into a state of check and should never be captured.
- Every move must result in a piece capture.

Benchmark Models
- Baseline: Backtracking algorithm iterating through entire search space of all possible actions.
- Oracle: Human generated solutions created by The author of Solitaire Chess.

Our Model
Applied a Deep Reinforcement Learning (DRL) Model based on Deep Q-Learning (DQN) to choose the best action for a given state. The estimated Q-Value is updated using Experience Replay. A multi-layer perceptron network was implemented using Google TensorFlow.
- Which action to take in a given state.
- Estimated value based on the current state.

Results
- Correctly identified 100% of final pieces.
- On average, explored less paths than baseline.

Q-Learning + Exploration

\[ Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha \left( r + \gamma \max_{a'} [E(Q(s', a'), N(s', a'))] \right) \]
Video

https://youtu.be/r9RTdeM09wM