AI Agents for Ultimate Tic-Tac-Toe

Phil Chen, Edward Xu, and Jesse Doan

Overview

- Ultimate Tic-Tac-Toe is a highly structured game involving 9 regular Tic-Tac-Toe boards
- Evaluated three different game-playing algorithms:
  - Minimax
  - Monte Carlo Tree Search
  - Deep Q-learning Network
- Best agent was Minimax with a simple evaluation function
- Agents performed well in games that matched their assumptions

Problem

- 2-player game with nine regular 3 by 3 Tic Tac Toe boards arranged in larger 3 by 3 grid
- At each move, players are restricted to moves in the smaller board corresponding to the same square that the opponent moved in (Figure 1)
- Player wins the game by winning three individual smaller boards that connect in a line (Figure 2)

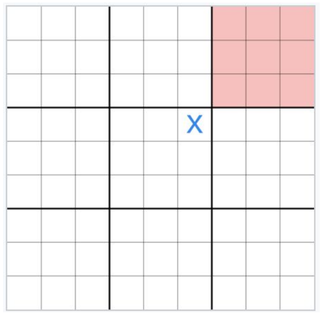


Figure 1: Squares shaded red indicate valid moves after 'X' is placed.

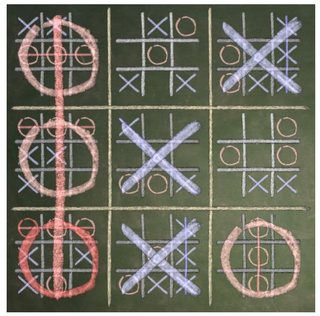


Figure 2: Winning state: 'O' wins the game

Models

Minimax

- Implemented a minimax strategy with alpha-beta pruning
- Simple evaluation function: total number of current miniBoards won

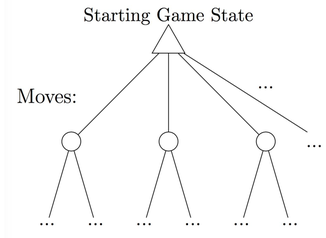


Figure 3: The game is modeled as a search tree, which our Minimax agent traversed to find optimal moves

Monte Carlo Tree Search

- Used Upper Confidence Bound MCTS Algorithm (Figure 4)

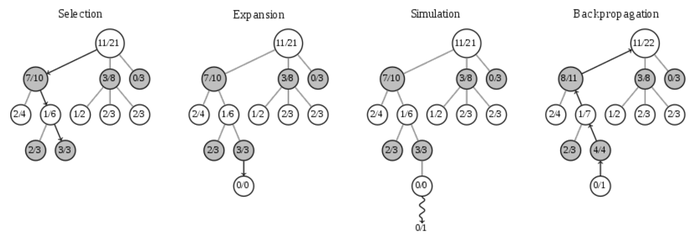


Figure 4: The generic Monte Carlo Tree Search algorithm involves a cycle of four steps. At the end of the search, the agent selects the node with the most visits.

Deep Q-Learning Network

- Architecture: Board > Conv2D (3,3) > Dense > Output
- Output is probability of winning from a given board

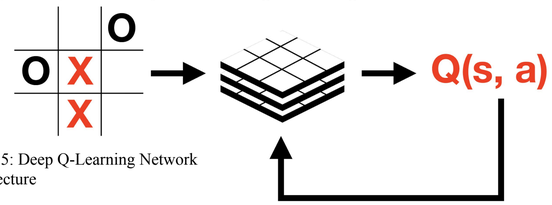
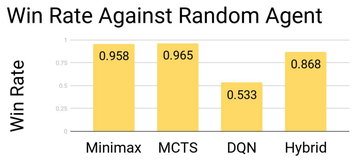
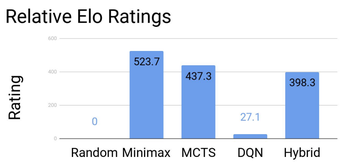


Figure 5: Deep Q-Learning Network Architecture

Results





Overall Win Rates

Player 2

Player 1	Random	Minimax	MCTS	DQN	Hybrid
Random		0.055	0.03	0.515	0.12
Minimax	0.97		0.72	0.98	0.63
MCTS	0.96	0.345		0.965	0.575
DQN	0.58	0.05	0.055		0.16
Hybrid	0.855	0.44	0.495	0.83	

- MCTS is more effective than minimax against random agent, probably because MCTS incorporates randomness into game tree
- Minimax wins against MCTS probably because minimax expects "worst case" while MCTS expects random play
- MCTS seems better at beginning while minimax seems better at end, but hybrid does not perform better
- Architecture of DQN was probably not suited for how structured yet small the board space was

Summary and Future Work

- MCTS is most effective against random agent, but minimax is most effective against "intelligent" agents
- Future: use Asynchronous Advantage Actor-Critic (A3C) combined with MCTS - modeled after AlphaZero network

Poster ideas
Overview

Problem
Ultimate Tic-Tac-Toe
Board demonstrating possible moves
Board demonstrating victory

Models
Minimax with evaluation function
Monte-Carlo Tree Search
Deep Q-learning Network

Results
Minimax, MCTS, DQN against random
Minimax vs MCTS vs DQN (maybe display table)
Relative Elo scores of Minimax, MCTS, DQN

Discussion
Deep reinforcement learning not as feasible for problem with such intricate structure and small action space

Summary/Future work
Combine MCTS + A3C