Deep Reinforcement Learning Agent of Gomoku

Ouraman Hajizadeh

June, 2023
ECT* Trento

Overview

- A generalization of Tic-Tac-Toe also known as 5 in a row
- 2 players, with a perfect play the first player wins
- The goal is to be the first player to get an unbroken row of five own symbols horizontally, vertically, diagonally or off-diagonally

Training A Gomoku Agent

- To know what is the best move we can evaluate every posible move at each step
- It can be done by palying out all the possible game paths
- Grows like $N^2, N^2(N^2-1), (N^2(N^2-1)N^2-3), \ldots, N^2!$, N = number of rows/columns
- The number of paths to be evaluated can grow up to $N^2!$ for the last move

AI approaches

- Alpha Beta algorithm: Reduces the search space by pruning non-optimal branches
- Monte Carlo Tree search
- Reinforcement Learning
- Deep Learning

Bellman Equation

- Learning from experience: Play out the game, backpropagate the reward
- V: Value of the state
- m: Move number
- C: Number of state visitis
- R: $R \in \{1, -1, 0\}$ for win, loss and draw
- γ: Decay constant of reward
- M: Number of the last move

$$V_t = \frac{(R\gamma^{(M-m)} - V_{t-1})}{C} + V_{t-1}$$

Move Selection

- Different Modes
  - Explore to learn the value of not yet visited states
  - Exploit the accumulated knowledge to evaluate the performance
- Traditional Approach: $\epsilon - \textit{Greedy}$ Policy
  - Each of the above modes are chosen based on the value of a random number (RN) and exploration rate $\epsilon \in [0, 1]$ at each game step
  - Exploration, $RN < \epsilon$: the player picks a position from a uniform distribution
  - Exploitation, $\epsilon < RN$: $V_{\tilde{s}} = \max\{V_{s_0}, \dots, V_{s_K}\}$, K : number of available states

Present Approach to RL

- Exploration mode, with parameter T: We choose the state, e.g. s_m by random sampling from the following distribution

$$p(s_m) = \frac{\exp(\frac{V_{s_m}}{T})}{\sum_{s_i} \exp(\frac{V_{s_i}}{T})}$$

$$V_{s_i} \in \{V_{s_0}, \ldots, V_{s_K}\} \quad -1 \leq V_{s_i} \leq 1$$

- Exploitation: $V_{\tilde{s}} = \max\{V_{s_0}, \ldots, V_{s_K}\}$, K : number of available states

Selection according to "Temperature"

- Interpolating between the above modes, using single parameter T, $V \in [-1, 1]$
- $T \rightarrow 10$: All states weighted equally
- $T \rightarrow 0$: The highest valued state is chosen
- $T \simeq 1$: All the states, according to their values, have a measurable chance of being chosen.

Search at high Temperature

- The first move of first round is chosen randomly
- High temperature is chosen for initial rounds:
- Not to be misled by the values based on few state visits
- The exploration is faster at higher temperature

Search at medium Temperature

- Middle rounds: There is a confidence on the learned values
- The search is controlled by the values not to waste time on paths that are far from optimal
- all the states have the chance of re-evaluation

Search at low Temperature

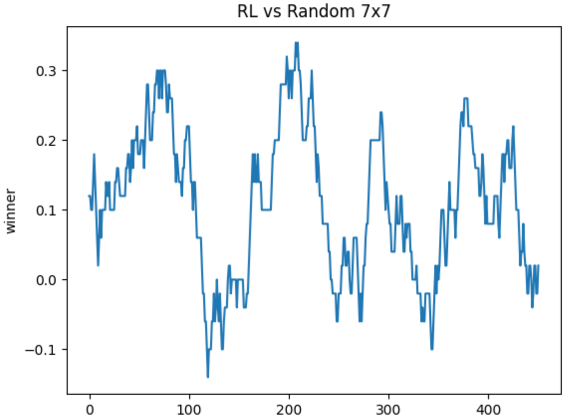
- Mainly highly evaluated states are visited
- At the last set of rounds of training
- To fine tune the set of highest values

RL training setup

- 40000 rounds for each temperature $T \in \{10, 1, 0.1, 0\}, \gamma = 0.9$ for 64 players
- The state-values (state: (value-count)) of all the players are merged into one, averaging over values of shared states weighted by their count of visit
- The final RL agent knows 10^7 states on 7x7 board
- Given the win size $= 5$, The shortest game path is of length 10, number of possible states at move 10th is 10^{12}

Moving average of winner

- The performance of RL agent versus untrained againt that learns while playing.
- Winner of each round $\in \{1, -1\} - >$ average winner $\in [-1, 1]$



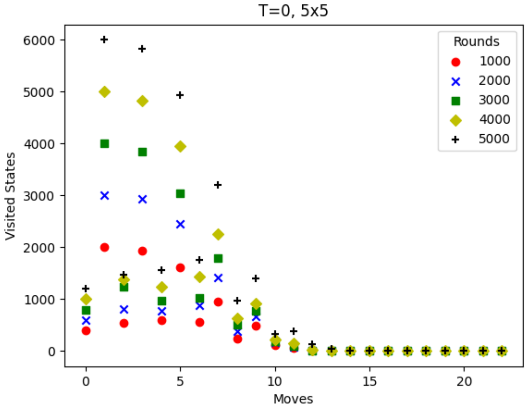
RL Limits

- Number of possible states at step m for board of size N^2:

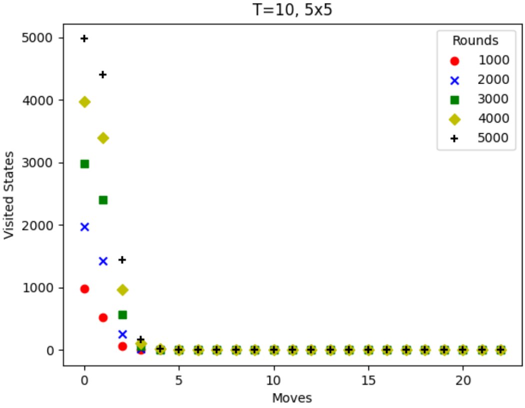
$$\binom{N^2}{\frac{m}{2}}\binom{N^2-\frac{m}{2}}{\frac{m}{2}} = \frac{N^2(N^2-1)\ldots(N^2-m+1)}{(\frac{m}{2}!)^2}, \qquad m \mod 2 = 0$$

- Maximum number of visited states at each round of game: N^2
- Maximum number different visited states after R rounds, (in case of minimum overlap): RN^2

- At T=0, The first player tries very small set of states compared to the second player
- The number of known states drops reaching the 10th move

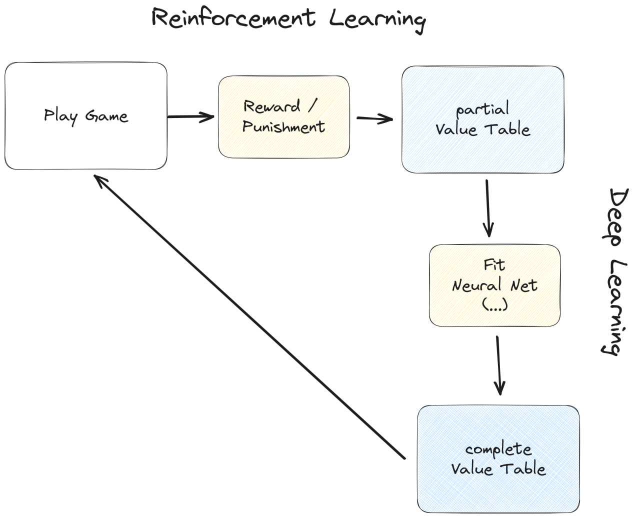


- At T=10, both players explore same size of the space
- No known states after the third moves mainly due to random wins, win size = 3

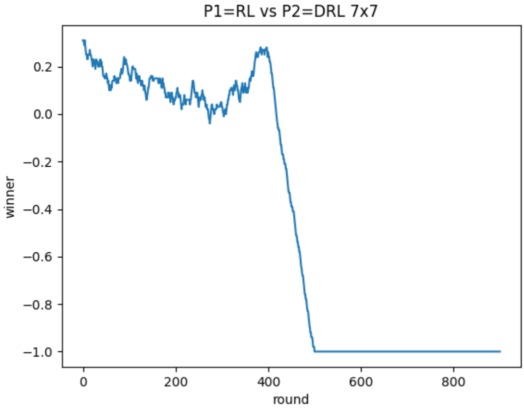


Why Deep Learning

- Reinforcement Learning alone infeasible on larger boards
- Agent needs to generalize, eg. learn patterns of symbols that indicate a win or a loss
- The structure of data suggests using Convolutional Neural Networks


Reinforcement Learning
Play Game
Reward / Punishment
partial Value Table
Fit Neural Net (...)
complete Value Table
Deep Learning

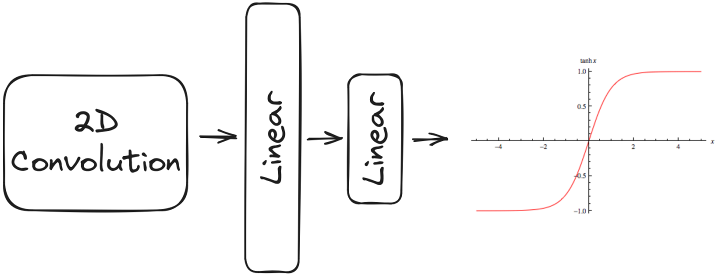
RL vs DRL



T=0 for P_1 and P_2, Board = 7x7

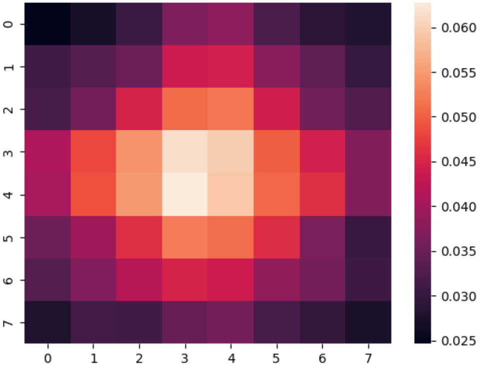
Neural Networks Architecture

- Kernel size = win number = 5
- Roughly 31000 parameters for 7x7 board

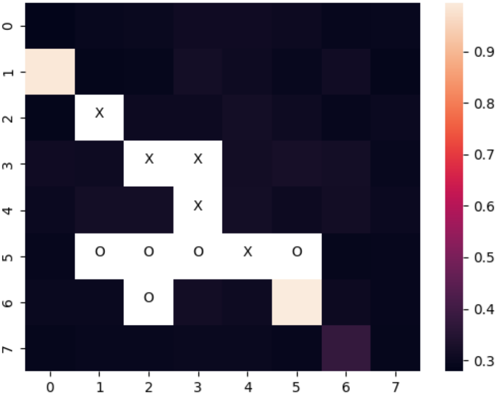


Opening

The strongest opening move is the center of the board. Which is achieved by same neural networks architecture on both 8x8 and 7x7 boards

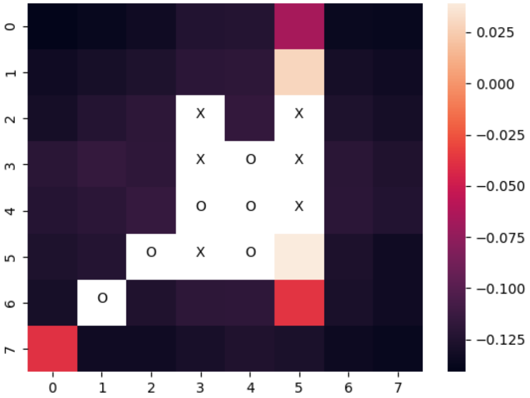


End Game

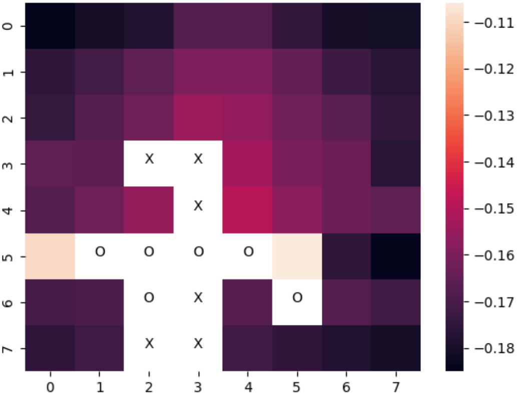


Defence Move

The reward for a defence is smaller and depend on the future of game

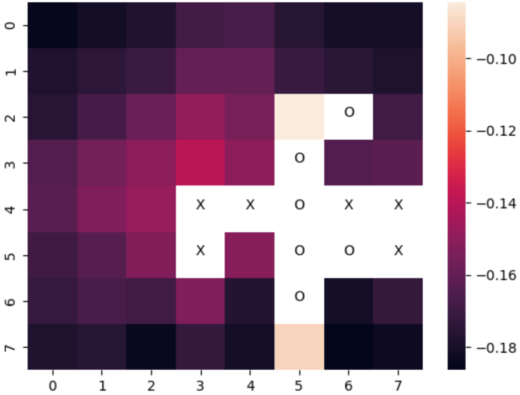


Defence Move

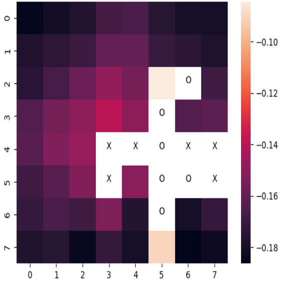


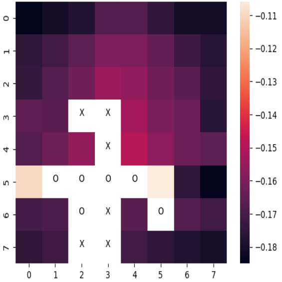
Rotation Symmetry

The value function seems to approximately preserve the rotational symmetry



Rotation Symmetry





Summary

- A DRL agent is trained on the 7x7 and 8x8, boards to play Gomoku
- The agent on both boards shows satisfactory opening and end game
- The defence strategy needs improvement

Outlook

- Merging state-values of first and second player into one: e.g. Reward for the second ≡ punishment for the first
- DRL vs DRL to update RL state-values, general improvement
- A signal of survival from the middle of the game could improve the defence strategy

Outlook

- Varying T for different players and game moves in same round; Let cold plays against hot
- Transfer learning on smaller board to larger one
- Formulating a Hamiltonian with a set of couplings that encode optimal moves to get insight to the neural nets behavior
- Non-zero couplings inside a window of kernel size, sliding over the board

$$p(s) \sim \exp(-H(s)) \quad H \sim generalized \quad Potts$$