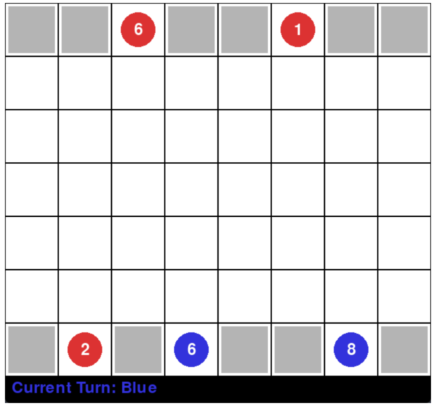
Simplified Jungle Chess (斗兽棋) AI Agents

Game Settings

Board

- 7 × 8 Grid



Pieces

- 8 pieces per side (Elephant, Lion, Tiger, Leopard, Wolf, Dog, Cat, Rat)

Simplified Rules

- **Rank-based Predation**: Higher rank eats lower rank
- **Special Case**: Rat $\leftrightarrow$ Elephant
- **No Traps / No Rivers / No Dens** (optional expansion)

Game End Condition

- Capture all opponent's pieces;
- No captures within a set number of movess (stalemates)

State & Action Space

State Definition

- **Board State**: 7×8 grid with 56 positions
- **Piece Information**: Each position can contain:
  - Empty (None)
  - Piece with attributes: `(player_id, strength, revealed)`
- **Turn Information**: Current player (0 or 1)

State Space Complexity

- Each square has 33 possible states:
  - Empty: 1 state
  - Player 0 pieces: 8 strengths × 2 visibility states = 16 states
  - Player 1 pieces: 8 strengths × 2 visibility states = 16 states
  - Total per square: 1 + 16 + 16 = 33 states
- **Estimated State Space**:
  - $\binom{56}{16} \times 16! \times 2^{16} \times 2 \approx 10^{32}$

Agent Portfolio

- Random
- Greedy
- Minimax
- Tabular Q-Learning
- Deep Q-Network (DQN)

Minimax Evaluation Function

$$\text{Score}_{\text{minimax}}(s) = \sum_{i=1}^{4} \text{Component}_i(s)$$

1. Piece Count Difference Component

$$\text{Component}_1(s) = (|P_{\text{self}}| - |P_{\text{opp}}|)$$

Where:

- $|P_{\text{self}}|$ and $|P_{\text{opp}}|$ represent the total number of pieces for the self and opponent, respectively.
- The weight coefficient is 100, reflecting the importance of piece count.

2. Piece Strength Sum Component

$$\text{Component}_2(s) = \left(\sum_{p \in P_{\text{self}}^{\text{revealed}}} \text{strength}(p) - \sum_{p \in P_{\text{opp}}^{\text{revealed}}} \text{strength}(p) \right) \times 10$$

Where:

- $P_{\text{self}}^{\text{revealed}}$ and $P_{\text{opp}}^{\text{revealed}}$ represent the set of revealed pieces for the self and opponent, respectively.
- $\text{strength}(p) \in \{1, 2, 3, 4, 5, 6, 7, 8\}$ denotes the strength of the piece.
- The weight coefficient is 10.

3. Information Value Component

$$\text{Component}_3(s) = 5 \times |P_{\text{self}}^{\text{unrevealed}}| - 10 \times |P_{\text{opp}}^{\text{unrevealed}}|$$

Where:

- $P_{\text{self}}^{\text{unrevealed}}$ and $P_{\text{opp}}^{\text{unrevealed}}$ represent the set of unrevealed pieces for the self and opponent, respectively.
- Unrevealed pieces for self are considered potential resources (+5 points).
- Unrevealed pieces for the opponent are considered unknown threats (-10 points).

4. Threat Opportunity Component

$$\text{Component}_4(s) = \sum_{p_s \in P_{\text{self}}^{\text{revealed}}} \sum_{p_o \in P_{\text{opp}}^{\text{revealed}}} \text{TacticalValue}(p_s, p_o)$$

Where the tactical value function is defined as:

$$\text{TacticalValue}(p_s, p_o) = \begin{cases} +20 & \text{if Adjacent}(p_s, p_o) \wedge \text{CanCapture}(p_s, p_o) \\ -15 & \text{if Adjacent}(p_s, p_o) \wedge \text{CanCapture}(p_o, p_s) \\ 0 & \text{otherwise} \end{cases}$$

Complete Minimax Evaluation Function

$$\boxed{\begin{aligned} H_{\text{minimax}}(s) = 1 \times (|P_{\text{self}}| - |P_{\text{opp}}|) \\ &+10 \times \left(\sum_{p \in P_{\text{self}}^{\text{revealed}}} \text{strength}(p) - \sum_{p \in P_{\text{opp}}^{\text{revealed}}} \text{strength}(p) \right) \\ &+5 \times |P_{\text{self}}^{\text{unrevealed}}| - 10 \times |P_{\text{opp}}^{\text{unrevealed}}| \\ &+\sum_{p_s, p_o} \text{TacticalValue}(p_s, p_o) \end{aligned}}$$

Greedy's Heuristic Function

$$\text{Score}(s) = \sum_{i=1}^{4} \text{Component}_i(s)$$

Where each component is defined as follows:

1. Piece Value Component

$$\text{Component}_1(s) = \sum_{p \in P_{\text{self}}} V(p) - \lambda_1 \sum_{p \in P_{\text{opp}}} V(p)$$

Where:

- P_{self} and P_{opp} represent the sets of pieces for the self and opponent, respectively.
- $V(p)$ is the value function for a piece:

$$V(p) = \begin{cases} \text{piece\_values}[\text{strength}(p)] & \text{if } p.\text{revealed} = \text{True} \\ 30 & \text{if } p.\text{revealed} = \text{False} \end{cases}$$

- $\text{piece\_values} = \{8 : 100, 7 : 90, 6 : 80, 5 : 70, 4 : 60, 3 : 50, 2 : 40, 1 : 10\}$
- $\lambda_1 = 0.8$ (opponent piece weight coefficient)

2. Positional Reward Component

$$\text{Component}_2(s) = \frac{\lambda_3}{10} \sum_{p \in P_{\text{self}}^{\text{revealed}}} [f_{\text{advance}}(p) + f_{\text{center}}(p)]$$

Where:

- Advance reward function:

$$f_{\text{advance}}(p) = \begin{cases} 5 \times \text{row}(p) & \text{if player\_id} = 0 \\ 5 \times (ROWS - 1 - \text{row}(p)) & \text{if player\_id} = 1 \end{cases}$$

- Central control reward function:

$$f_{\text{center}}(p) = \begin{cases} 5 & \text{if col}(p) \in \{\lfloor COLS/2 \rfloor - 1, \lfloor COLS/2 \rfloor\} \\ 0 & \text{otherwise} \end{cases}$$

- $\lambda_3 = 15$ (positional weight coefficient)

3. Mobility Component

$$\text{Component}_3(s) = \lambda_2 \times |A_{\text{possible}}(s)|$$

Where:

- $A_{\text{possible}}(s)$ represents the set of all possible actions in the current state.
- $\lambda_2 = 2$ (mobility weight coefficient)

4. Safety Component

$$\text{Component}_4(s) = w_{\text{safety}} \sum_{p_s \in P_{\text{self}}^{\text{revealed}}} \sum_{p_o \in P_{\text{opp}}^{\text{revealed}}} \text{ThreatValue}(p_s, p_o)$$

Where the threat value function is defined as:

$$\text{ThreatValue}(p_s, p_o) = \begin{cases} -10 & \text{if Adjacent}(p_s, p_o) \land \text{CanCapture}(p_o, p_s) \\ +15 & \text{if Adjacent}(p_s, p_o) \land \text{CanCapture}(p_s, p_o) \\ 0 & \text{otherwise} \end{cases}$$

Capture condition function:

$$\text{CanCapture}(p_1, p_2) = \begin{cases} \text{True} & \text{if } (\text{strength}(p_1) > \text{strength}(p_2)) \land \neg(\text{strength}(p_1) = 8 \land \text{strength}(p_2) = 1) \\ \text{True} & \text{if strength}(p_1) = 1 \land \text{strength}(p_2) = 8 \\ \text{False} & \text{otherwise} \end{cases}$$

- $w_{\text{safety}} = 10$ (safety weight coefficient)

Complete Heuristic Function

$$\boxed{\begin{aligned} H(s) = \sum_{p \in P_{\text{self}}} V(p) - 0.8 \sum_{p \in P_{\text{opp}}} V(p) \\ +1.5 \sum_{p \in P_{\text{self}}^{\text{revealed}}} [f_{\text{advance}}(p) + f_{\text{center}}(p)] \\ +2 \times |A_{\text{possible}}(s)| \\ +10 \sum_{p_s, p_o} \text{ThreatValue}(p_s, p_o) \end{aligned}}$$

Q-Learning (Tabular)

1. Basic State Definition

The state tensor $\mathbf{S} \in \mathbb{R}^{7\times8\times4}$ is defined where each position (r, c) has a feature vector:

$$\mathbf{s}_{r,c} = \begin{bmatrix} p_{r,c} \\ \frac{\text{strength}_{r,c}}{8} \\ \mathbb{I}_{\text{revealed}}(r, c) \\ \mathbb{I}_{\text{occupied}}(r, c) \end{bmatrix} \in \mathbb{R}^4$$

Where:

$$p_{r,c} = \begin{cases} 0 & \text{if piece at } (r,c) \text{ belongs to player 0} \\ 1 & \text{if piece at } (r,c) \text{ belongs to player 1} \\ 0 & \text{if position } (r,c) \text{ is empty} \end{cases}$$

$$\frac{\text{strength}_{r,c}}{8} = \begin{cases} \frac{k}{8} & \text{if piece at } (r,c) \text{ has strength } k \in \{1,2,\ldots,8\} \\ 0 & \text{if position } (r,c) \text{ is empty} \end{cases}$$

$$\mathbb{I}_{\text{revealed}}(r,c) = \begin{cases} 1 & \text{if piece at } (r,c) \text{ is revealed} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbb{I}_{\text{occupied}}(r,c) = \begin{cases} 1 & \text{if position } (r,c) \text{ contains a piece} \\ 0 & \text{if position } (r,c) \text{ is empty} \end{cases}$$

Action Space

- **Reveal Action**: `("reveal", (row, col))` for unrevealed own pieces
- **Move Action**: `("move", (start_row, start_col), (end_row, end_col))`

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

Hyperparameters

- $\alpha = 0.1 \quad \gamma = 0.95 \quad \varepsilon$-greedy $\downarrow$

$$r_{t+1} = \text{reward}$$

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)$$

$$Q_{\text{new}}(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot \delta_t$$

reward function

$$r_{t+1} = R_{\text{action}}(s_t, a_t) + R_{\text{terminal}}(s_{t+1})$$

1. Action Reward $R_{\text{action}}(s_t, a_t)$

$$R_{\text{action}}(s_t, a_t) = \begin{cases} w_{\text{reveal}} \times V(\text{piece}) & \text{if } a_t = \text{reveal} \wedge \text{success} \\ w_{\text{capture}} \times V(\text{captured}) & \text{if } a_t = \text{move} \wedge \text{capture\_success} \\ w_{\text{be\_captured}} \times V(\text{lost}) & \text{if } a_t = \text{move} \wedge \text{be\_captured} \\ w_{\text{mutual}} & \text{if } a_t = \text{move} \wedge \text{mutual\_destruction} \\ w_{\text{invalid}} & \text{if } a_t \notin \mathcal{A}_{\text{valid}} \\ R_{\text{tactical}}(s_t, a_t) & \text{otherwise} \end{cases}$$

2. Weight Parameters

```
self.weights = {
    'win_game': 100.0,
    'lose_game': -100.0,
    'draw_game': 0.0,
    'capture_piece': 10.0,
    'be_captured': -8.0,
    'mutual_destruction': -0.5,
    'reveal_piece': 1.0,
    'survival_penalty': -0.1
}
```

3. Piece Value Function $V(\text{piece})$

$$V(k) = \begin{cases} 3.0 & \text{if } k = 1 \text{ (Rat)} \\ 1.0 & \text{if } k = 2 \\ 1.5 & \text{if } k = 3 \\ 2.0 & \text{if } k = 4 \\ 2.5 & \text{if } k = 5 \\ 3.0 & \text{if } k = 6 \\ 3.5 & \text{if } k = 7 \\ 4.0 & \text{if } k = 8 \text{ (Elephant)} \end{cases}$$

4. Evaluate Position Reward $R_{Pos}(s_t, a_t)$

$$R_{Pos}(s_t, a_t) = \frac{1}{n} \sum_i P(i) \times V(i) \times (\Delta(Opp) - \Delta(Threat))$$

Where the $P(i)$ is the probability of the i-th unrevealed piece to be at the selected position.

$$P(i) = \frac{1}{\text{num of unrevealed pieces}}, Opp = \sum_i Opp(i), Threat = \sum_i Threat(i)$$

Where the $\Delta(Opp)$ and $\Delta(Threat)$ are the changes in the opponent's and your threat levels, respectively, after the action a_t is taken.

if nearest enemy of piece i can be captured by i:

$$Opp(i) = \frac{3}{1 + \text{distance}(i, \text{nearest enemy}(i))}, \quad Threat(i) = 0$$

if nearest enemy of piece i can captured i:

5. Terminal Reward $R_{\text{terminal}}(s_{t+1})$

$$R_{\text{terminal}}(s_{t+1}) = \begin{cases} +100.0 & \text{if } \text{Win}(s_{t+1}, \text{current\_player}) \\ -100.0 & \text{if } \text{Lose}(s_{t+1}, \text{current\_player}) \\ 0.0 & \text{if } \text{Draw}(s_{t+1}) \lor \neg\text{Terminal}(s_{t+1}) \end{cases}$$

6. Complete Reward Function

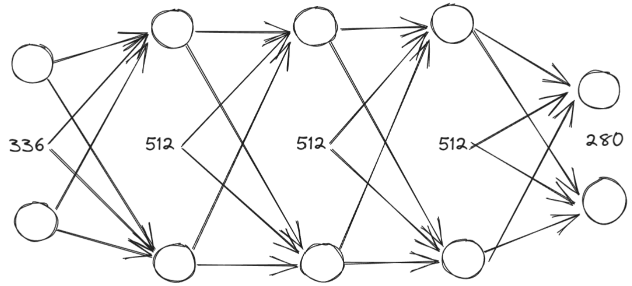
$$
r_{t+1} = \begin{cases} -0.1 + 1.0 \times \text{Expectation}(\Delta(Opp) - \Delta(Threat)) + R_{terminal} & \text{if reveal} \\ -0.1 + 10.0 \times V(capture) + \Delta(Opp) - \Delta(Threat) + R_{terminal} & \text{if capture} \\ -0.1 - 8.0 \times V(lost) + \Delta(Opp) - \Delta(Threat) + R_{terminal} & \text{if lost} \\ -0.1 - 0.5 + \Delta(Opp) - \Delta(Threat) + R_{terminal} & \text{if mutual destruction} \\ -0.1 + \Delta(Opp) - \Delta(Threat) + R_{terminal} & \text{others} \end{cases}
$$

Deep Q-Network

1. Model Architecture

1.1 Neural Network Structure

The DQN employs a 4-layer fully connected neural network architecture:
$S : 7 \times 8 \times 6 = 336$



- $h_1 = \mathrm{ReLU}(W_1 \cdot s + b_1)$
- $h_k = \mathrm{Dropout}(\mathrm{ReLU}(W_k \cdot h_{k-1} + b_k))$
- $Q(s, a) = W_4 \cdot h_3 + b_4$

1.2 State Space Representation

The state space uses a 6-channel representation, with dimensions $7 \times 8 \times 6 = 336$:

Channel	Meaning	Value Range
0	Player 0 piece position	{0, 1}
1	Player 1 piece position	{0, 1}
2	Piece strength (normalized)	[0, 1]
3	Piece revealed status	{0, 1}
4	Current player's pieces	{0, 1}
5	Opponent's pieces	{0, 1}

1.3 Action Space Encoding

Action space size: $|\mathcal{A}| = 280$

- **Reveal Action**: $a_{reveal} \in [0, 55]$
    - Encoding formula: $index = r \times 8 + c$
- **Move Action**: $a_{move} \in [56, 279]$
    - Encoding formula: $index = 56 + r_1 \times 32 + c_1 \times 4 + \text{direction}$
    - Direction mapping: $\{(-1, 0) : 0, (1, 0) : 1, (0, -1) : 2, (0, 1) : 3\}$

2. Training Parameters

2.1 Core Hyperparameters

Parameter	Symbol	Default Value	Description
Learning Rate	α	10^{-4}	Adam optimizer learning rate
Discount Factor	γ	0.99	Future reward discount factor
Exploration Rate	ϵ	0.1	ϵ-greedy policy exploration probability
Exploration Decay	ϵ_{decay}	0.995	Per-episode exploration rate decay factor
Minimum Expl. Rate	ϵ_{min}	0.01	Lower bound for exploration rate
Batch Size	B	64	Experience replay batch size

2.2 Exploration Strategy

A decaying ϵ-greedy strategy is used:

$$\epsilon_t = \max(\epsilon_{min}, \epsilon_0 \times \epsilon_{decay}^t)$$

Action selection probability:

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{|\mathcal{A}_{valid}|} + (1 - \epsilon) & \text{if } a = \arg\max_{a'} Q(s, a') \\ \frac{\epsilon}{|\mathcal{A}_{valid}|} & \text{otherwise} \end{cases}$$

3. Loss Function and Optimization

3.1 Q-Learning Target

Target Q-value calculation:

$$y_i = r_i + \gamma \max_{a'} Q_{target}(s'_i, a')$$

3.2 Loss Function

Mean Squared Error (MSE) loss is used:

$$\mathcal{L}(\theta) = \frac{1}{B}\sum_{i=1}^{B}[Q(s_i, a_i; \theta) - y_i]^2$$

3.3 Optimizer

Adam optimizer is used:

- Learning rate: $\alpha = 10^{-4}$
- $\beta_1 = 0.9$, $\beta_2 = 0.999$
- $\epsilon = 10^{-8}$

4. Reward Function Design

4.1 Immediate Rewards

Action Type	Reward Value	Condition
Reveal Piece	+0.1	Successfully revealed own piece
Move Piece	+0.05	Normal move
Capture Piece	$+0.2 \times \text{strength}$	Successfully captured opponent's piece
Be Captured	-0.3	Own piece was captured
Invalid Action	-1.0	Executed an invalid action

4.2 Terminal Rewards

$$R_{terminal} = \begin{cases} +10 & \text{Victory} \\ -10 & \text{Defeat} \\ 0 & \text{Draw} \end{cases}$$

$$
r_i = \begin{cases} 0.1 + R_{\text{terminal}} & \text{if successfully revealing own piece} \\ 0.05 + R_{\text{terminal}} & \text{if making a normal move} \\ 0.2 \times k + R_{\text{terminal}} & \text{if successfully capturing opponent's piece of strength } k \\ -0.3 + R_{\text{terminal}} & \text{if own piece is captured} \\ -1.0 & \text{if executing an invalid action} \\ R_{\text{terminal}} & \text{otherwise} \end{cases}
$$

5. Network Update Mechanism

5.1 Experience Replay

Randomly sample a batch of experiences from the experience buffer D:

$$\{(s_i, a_i, r_i, s'_i, done_i)\}_{i=1}^{B} \sim \text{Uniform}(D)$$

5.2 Target Network Update

The target network is updated every 100 episodes:

$$\theta_{target} \leftarrow \theta_{main}$$

5.3 Gradient Update

Backpropagation is used to update the main network parameters:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$$

Experimental Results

Agent	vs Random	vs Greedy
Random	50	25
Greedy	75	–
Minimax d=3	92	68
Q-Learning	78	52
DQN	85	61

Thank You!

Q & A Welcome 🎉