

Project Report
Artificial Intelligence
[CSE 3705]

Breakthrough Game using Q-Learning

By
Godavarthi Sai Nikhil
210214
Anish Borkar
210220



Department of Computer Science and Engineering
School of Engineering and Technology
BML Munjal University
November 2023

Declaration by Candidates

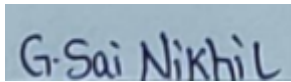
We hereby declare that the project entitled “Breakthrough Game using Q-Learning” has been carried out to fulfill the partial requirements for completion of the course Artificial Intelligence offered in the 5th Semester of the Bachelor of Technology (B.Tech) program in the Department of Computer Science and Engineering during AY-2023-24 (odd semester).

This experimental work has been carried out by us and submitted to the course instructor Dr. Soharab Hossain Shaikh. Due acknowledgments have been made in the text of the project to all other materials used.

This project has been prepared in full compliance with the requirements and constraints of the prescribed curriculum.

Name: Godavarthi Sai Nikhil

Signature of Student-1:



Name: Anish Borkar

Signature of Student-2:



Place: BML Munjal University

Date: 18th November 2023

Contents

	Page No.
1. Introduction and Problem Statement	4-6
2. Methodology	7-10
2.1 Table Driven Approach	
2.2 Intelligent Agent	
3. Implementation and Technology Stack	11-12
4. Conclusions	13
5. References	13
6. Appendix	14

Introduction and Problem Statement

Breakthrough is a two-player strategy board game combining chess and traditional war games. The game is played on an 8x8 square board, like a chessboard. Each player has 2 rows of pawns, 'White' for the player and 'Black' for the AI, and the objective is to advance a pawn to the opponent's home row while capturing or blocking the opponent's pawns. The game involves strategic moves, capturing opponents, special moves like en passant, and planning to defeat the opponent.

Gameplay Mechanics:

- **Starting Position:** Breakthrough begins with an 8x8 game board, with each player starting with 16 pawns placed on the two rows closest to them. The pawns are represented by 'black' for the AI player and 'white' for the human player.

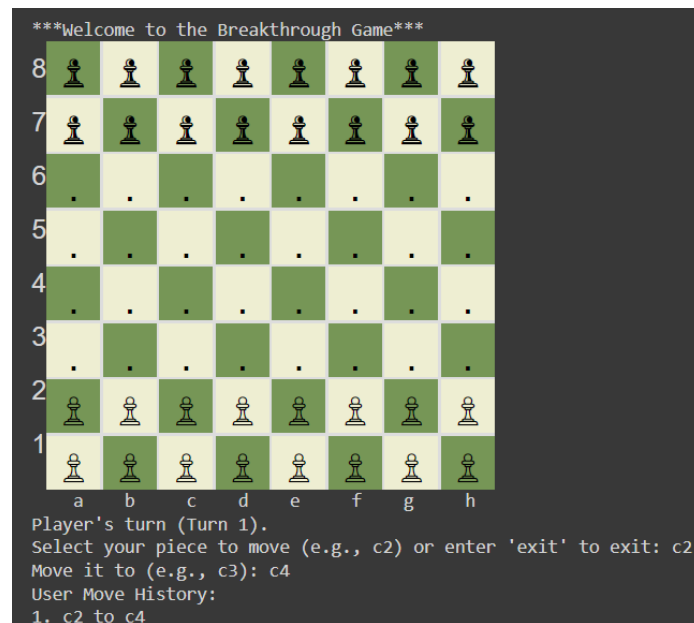


Fig 1.1: Initial of the Breakthrough Game Problem.

- **Player Turns:** Players take turns making moves. The player moves first, followed by the AI player. On each turn, a player can select one of their pawns and make a legal move based on the specific rules of movement for pawns in Breakthrough.
- **Legal Moves:** Pawns can move forward one square, and on their first move, they have the option to move forward two squares. Pawns capture opponents diagonally. En-passant captures are also possible when a pawn moves two squares forward, and an opponent's pawn could have captured it had it moved only one square.
- **AI Strategy:** The AI player employs a strategy based on predefined rules, including prioritizing moving closer to the player's home row, capturing the player's pawns, blocking the player's progress, utilizing the two-square first move option, considering en passant moves, and focusing on reaching the other end of the board for a quick win in the endgame.

- **Winning the Game:** The game concludes when one of the player's pawns reaches the opposite end of the board. The player whose pawn reaches the last row (for the player) or the first row (for the AI) wins the game. Additionally, the game may end if a player cannot make a valid move, and the opponent is declared the winner.
- **En Passant:** If a pawn moves two squares forward from its starting position and lands adjacent to an opponent's pawn, the opponent has the option to capture the moving pawn as if it had only moved one square.

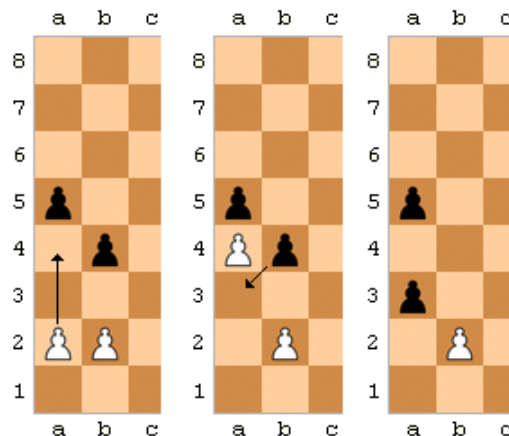


Fig: 1.2 En-passant Move.

- **Endgame:** The endgame strategy involves a focus on quickly reaching the opponent's end of the board, especially when the number of moves is limited.

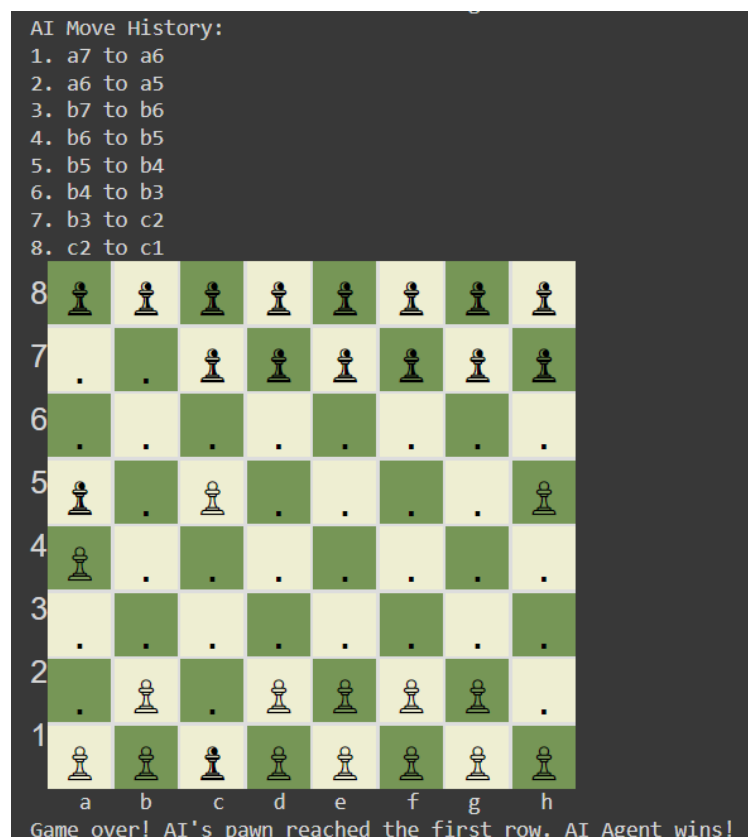


Fig: 1.3 Goal of the Breakthrough Game Problem (Reaching the other end).

Problem Description:

Making an intelligent AI player for Breakthrough is the challenging part. The AI must move across the game board, prioritizing pawn captures, strategic advances, and strong defense. The range of possible actions like two-square initial moves, diagonal captures, and a complex en passant move is what leads to the complexity. The main challenge is creating an AI that can use reinforcement learning to gain knowledge and modify its approach.

Objectives:

1. The main objective is to implement an AI player using reinforcement learning that learns to play Breakthrough effectively.
2. Train the AI to understand effective strategies such as moving closer to the player's home row, capturing player pawns, and blocking the player's progress.
3. Teach the AI to utilize special moves like en passant, enhancing its ability to defeat the player.

End Goal:

By the end of the project, we aim to have an AI Breakthrough player that demonstrates intelligent decision-making based on learned strategies. The AI should be capable of competing against a human player, making informed moves, and adapting its strategy during gameplay. The game ends when either a player's or AI's pawn reaches the other end.

Methodology

2.1 Table Driven Approach

In the Table-Driven approach for the Breakthrough Game AI, we utilize a set of predefined rules and strategies mapped in tables. The agent observes specific percepts and, based on these percepts, selects appropriate actions from the predefined set.

Utility of Table-Driven Approach

1. Structured Decision Making:

By mapping percepts to specific actions, the AI agent can make decisions in a more structured and organized manner.

2. Scalability and Modification:

It is easier to scale or modify the AI's behavior by updating the tables rather than changing the core logic.

3. Clear Separation of Logic:

This approach provides a clear separation between the game's state (percepts) and the actions taken, facilitating easier debugging, and understanding of the AI's behavior.

Percepts:

1. AI's Turn: Perception that it is currently the AI's turn to make a move.
2. Positions of AI's Pawns Closest to Player's Home Row.
3. Positions of Player's Pawns Closest to AI's Home Row.
4. Available Legal Moves for AI: Available legal moves for the AI to make in the current turn, including en passant.
5. Determining Game Win or Loss.

Actions:

1. Move pawn closer to opponent's home row - "Move Closer".
2. Move to capture opponent's pawn if possible: "Capture".
3. Move to block and capture the player's pawn if it's near rows 4 or 5: "Block Capture".
4. Use two-square move option on the first move if available: "TwoSquareMove".
5. Consider en passant move if applicable: "En Passant".
6. Focus on reaching the other end of the board quickly for a win: "Endgame Focus".

Moves:

1. Forward Movement Only: Move a pawn closer to the opponent's home row in the forward direction only.
2. Two-Square First Move: Use the two-square move option on the first move if available.
3. Capture Opponent's Pawns: Move a pawn to capture the opponent's pawns, considering forward and diagonal directions.

4. En Passant: Special move to capture an opponent's pawn that has moved two squares forward from its starting position.

Percept to Action Mapping Table:

Percept	Action
AI's Turn	Move Closer, Capture, Block Capture, Two Square Move, En Passant, Endgame Focus
Positions of AI's Pawns Closest to Player's Home Row	Move Closer
Positions of Player's Pawns Closest to AI's Home Row	Capture
Available Legal Moves for AI	Block Capture
Forward Movement Only	Block Capture
Two-Square First Move	Two Square Move
Current Position of All Pieces	Endgame Focus

Agent Execution:

1. The agent, given the current state, analyzes the available percepts and looks up to the corresponding action from the table.
2. The action is then executed, and the agent transitions to the next state.

2.2 Intelligent Agent

The Breakthrough Game AI becomes intelligent through a reinforcement learning approach. The agent learns from experience and adjusts its strategy based on the observed outcomes of its moves. we used the Q-Learning algorithm to solve the problem. The following steps outline the application of Q-learning to the Breakthrough game:

The Breakthrough agent implicitly discretizes the state space by representing the game board as a string. This representation allows the agent to effectively manage and learn from different game states.

Q-Table Initialization:

The Q-table is a crucial component of the Q-learning algorithm. It represents a mapping of state-action pairs to their respective Q-values, indicating the expected cumulative reward for taking an action in a particular state. This table starts with initial values, and this class initializes it, ensuring that continuous state variables, such as pawn positions, are appropriately converted to discrete indices for efficient learning.

Epsilon Greedy Algorithm:

This method is used to strike a balance between exploration and exploitation. Epsilon-greedy ensures that the agent continues to explore different actions, even those that might not have resulted in high rewards in the past. This exploration is crucial for discovering better strategies and navigating the complex terrain effectively.

In epsilon-greedy, the agent makes decisions as follows:

1. With probability $1 - \epsilon$, chooses the action with the highest estimated Q-value (exploitation).
2. With probability ϵ , the agent chooses a random action from the action space (exploration).

At the beginning of the learning process, the agent has limited knowledge about the optimal actions to take in different states. Purely exploiting this incomplete knowledge might lead to suboptimal policies. Epsilon-greedy encourages the agent to explore new actions, gather more information, and update its Q-values, enabling it to make more informed decisions as the learning progresses.

As the agent gains more experience and refines its understanding of the environment, the need for exploration decreases. Epsilon-greedy accommodates this by gradually reducing the exploration rate over time.

The dynamic adjustment of epsilon ensures that the agent becomes more focused on exploiting its learned knowledge as it becomes more experienced.

Training Process:

The agent undergoes training over a series of episodes, where actions, updates, rewards, and Q-value adjustments take place:

- The agent's position starts randomly, and actions are iteratively selected.
- Updates, rewards, and Q-value adjustments occur based on the Bellman equation, incorporating current rewards, discounted future rewards, and existing knowledge.

Bellman Equation for Q-Learning:

$$Q(s, a) = (1 - \alpha) * Q(s, a) + \alpha * [r + \gamma * \max_{a'} Q(s', a)]$$

Where

s is the state

a is the action

s' is the next state

α is the Learning rate.

r is the Immediate reward.

γ is the Discount factor.

Strategy Mapping:

To enhance its decision-making process, the function maps optimal moves to predefined strategies. These strategies encompass actions such as prioritizing movement closer to the player's home row, evaluating opportunities to capture the player's pawns, blocking the player's progress strategically, and more.

The agent becomes "intelligent" by iteratively exploring and learning from its interactions with the environment. The Q-learning algorithm allows the agent to update its Q-table based on the observed rewards, gradually improving its policy for choosing actions in different states.

Implementation & Technology Stack

3.1 Technology Stack

The following packages are used for the implementation of the Breakthrough game Problem using Q-Learning:

1. **Programming Language:**
The code is written in Python.
2. **Libraries used:**
 1. **NumPy:** Used for numerical operations, especially for handling arrays and mathematical computations efficiently.
 2. **Sys.**
 3. **IPython display:** Used for interactive displays and HTML rendering.
3. **Development Environment:**
The code is designed to run on standard Python development environments, including VsCode, Google Colab, and Jupyter Notebook. The use of IPython.display is specific to IPython environments and enhances interactive display capabilities.

3.2 Implementation (Key Classes and Functions):

I. Breakthrough Board Class:

- **Initialization:** The `__init__` method initializes the game board, sets player attributes, and defines key parameters like learning rate, discount factor, and exploration probability.
- **Display Board:** The `display_board` method generates an HTML representation of the game board for visualizing the current state.
- **Make Move:** The `make_move` method executes player moves, updates the board, and incorporates the Q-learning mechanism for learning from the game state.
- **AI Make Move:** The `ai_make_move` method encapsulates the intelligence of the AI agent. It explores predefined strategies, calculates Q-values, and determines optimal moves based on the current state.
- **Q-learning Methods:** `get_q_value`, `get_q_values`, and `update_q_value` methods facilitate Q-learning by retrieving, updating, and managing Q-values for state-action pairs.
- **En Passant:** The `check_en_passant` method identifies and handles en passant moves, contributing to the strategic decision-making process.

II. Helper Methods:

- **Coordinates Conversion:** Methods like `square_to_coordinates` and `coordinates_to_square` convert between square notation and matrix coordinate.
- **Validity Checks:** Methods like `is_valid_move` and `is_valid_input` ensure the validity of player inputs and moves, contributing to a robust gameplay experience.
- **find_closest_pawns:** Identifies the closest pawns for both players.
- **Display Move History:** Methods like `display_user_move_history`, `display_ai_move_history`: Outputs move history for user and AI players.

III. Game Flow:

- The `play_game` method orchestrates the flow of the Breakthrough game, alternating between player and AI turns, executing moves, and determining the game's outcome.

Q-learning principles are implemented through functions like `get_q_value`, `update_q_value`, and `ai_make_move`, which assess the current state and make decisions based on learned values. The strategy table in `ai_make_move` reflects predefined strategies corresponding to different game states.

Within these classes and functions, decision-making, learning mechanisms, and game state control are all stated, giving the Breakthrough game code a readable structure. This makes it easier to understand the AI agent's intelligence and how it interacts strategically with the game environment.

Conclusion

In conclusion, the implementation of Q-learning in our Breakthrough intelligent agent introduces a dynamic and adaptive element to the decision-making process. Unlike the static nature of table-driven approaches, Q-learning allows our agent to learn and update its strategies based on the outcomes of previous interactions with the game environment. The agent learns from its own experiences, adjusting its approach over time to improve its decision-making capabilities.

By continuously updating its Q-values based on the rewards and penalties associated with different actions, the agent evolves and refines its strategies in response to the evolving dynamics of the game. This adaptability enhances the agent's ability to navigate through complex patterns and make informed decisions, contributing to a more robust and difficult gameplay.

The sensitivity analysis on hyperparameters, including the discount factor, learning rate, and epsilon decay rate, reveals insights into their impact on the algorithm's learning dynamics. The balance between exploration and exploitation, achieved through an epsilon-greedy strategy, contributes to the agent's ability to converge on optimal policies. Overall, the code provides valuable insights into the application of Q-learning for solving simple reinforcement learning problems.

Future Scope: The AI's decision-making process could be further refined, and additional features, such as a more sophisticated evaluation function, could enhance the agent's overall performance.

References

References

brainking. (n.d.). Retrieved from brainking: <https://brainking.com/en/GameRules?tp=84>

Breakthrough (board game). (2022, January 21). Retrieved from wikipedia:

[https://en.wikipedia.org/wiki/Breakthrough_\(board_game\)#:~:text=To%20play%20the%20game%20on,moving%20one%20piece%20per%20turn.&text=A%20piece%20may%20move%20one,the%20target%20square%20is%20empty](https://en.wikipedia.org/wiki/Breakthrough_(board_game)#:~:text=To%20play%20the%20game%20on,moving%20one%20piece%20per%20turn.&text=A%20piece%20may%20move%20one,the%20target%20square%20is%20empty).

Kerner, S. M. (2023, May). *Q-learning*. Retrieved from techtarget:

<https://www.techtartget.com/searchenterpriseai/definition/Q-learning>

Q-learning. (2023, October 27). Retrieved from wikipedia: <https://en.wikipedia.org/wiki/Q-learning>

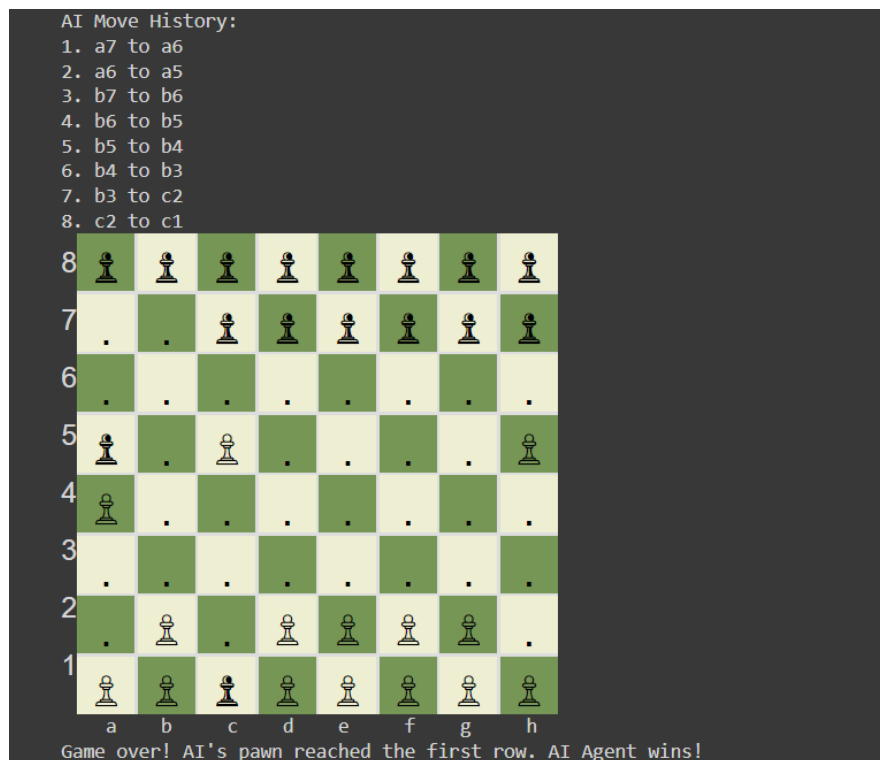
Appendix

Colab Notebook:

<https://colab.research.google.com/drive/1rJF-BEhjmXPAv3yZs3SJEF2W3ngqUS3K?usp=sharing>

Output:

For the Video click this <https://clipchamp.com/watch/Dh9h2BFwAR9>



Code Till Table Driven Approach: <https://onlinegdb.com/BgKdKUxhc>

Final Code: <https://onlinegdb.com/PQPafVTy8>