Minesweeper AI Agent

Zelalem Abahana

Alexandros Sfikas

Jared Heidt

Foundations of AI (*Spring, 2024*)

PennState

Agenda

- Problem Introduction
- Problem Description
- Solution Overview
- Model
- Code Snippets
- Performance
- Results
- Conclusion

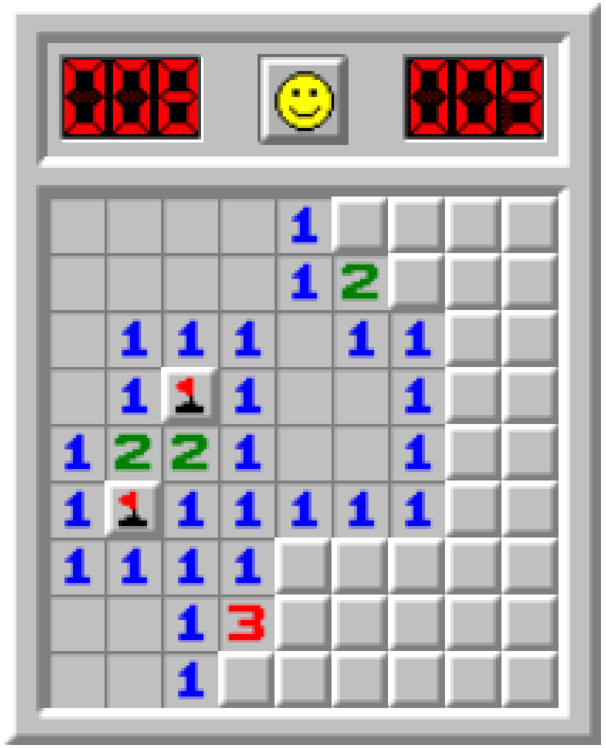
PennState

Don’t You Hate Being Stuck In a Minefield?

- One of the worst feelings in the world is when you have stumbled into a minefield
- Wouldn’t it be useful to have a tool that can detect where all the minefields are within a defined space
- You would not even have to think about how to get out
- This tool would be a great timesaver!

Context and Problem

- Minesweeper is a logic puzzle video game where players are meant to clean a “minefield” [2]
- The board is divided into cells, with mines randomly distributed
- The number on a cell shows the number of mines adjacent to it
- Cells suspected of being mines can be marked with a flag



Screenshot of a Minesweeper puzzle while being completed

PennState

Problem Description

- Single Agent
  - Minesweeper is a one player game
- Hidden State game
  - The location of the mines are hidden from the player
- State Space
  - The current layout of the board with each cell being unrevealed, revealed, or flagged
- Action Space
  - Revealing a hidden square
  - Flagging a mine
- Structuring rewards will determine how the agent learns

PennState

Overview of Solution and Contributions

Solution: Q-learning

- Created a Q-learning agent which is an implementation of reinforcement learning
- The agent learns an action-utility function (Q-function) giving the expected utility of taking a given action in each state [3]

Contributions

- Zelalem Abahana: Game logic and agent development
- Alexandros Sfikas: Report writing
- Jared Heidt: Visualizing agent performance, experiments, presentation

PennState

AI Model

- In Q-learning, the model is the Q-table
- The Q-table stores the Q-values – the expected reward of taking a particular action at a particular state
  - Rows represent states
  - Columns represent actions
- The agent updates the Q-values based on the rewards received from the environment after taking an action
- Using those values, the agent can select the action with the highest Q-value for a given state

AI Model: Code Snippets

```
self.q_values = np.zeros((board_size, board_size, 3))
```

(1) Initialization of the Q-table showing its size is determined by the number of states and actions

```
if board[state] > 0 and np.random.rand() < self.epsilon:
```

(2) Choosing exploitation or exploration

```
action, next_state = agent.take_action(state, board)
```

(3) Perform the selected action and getting
the reward and resultant outcome state

```
updated_q_value = current_q_value + self.learning_rate * (reward + self.discount_factor * next_q_value - current_q_value)
```

(4) Updating the Q-table using the Bellman equation

PennState

Console output

```
Reward for current iteration: 1
Current Board:
0 0 0 1 1 1 0 0 0 0
0 0 0 1 M 1 0 0 0 0
0 0 0 2 2 2 0 0 1 1
1 2 2 2 M 1 1 1 2 M
2 M M 2 1 1 1 M 2 1
M 3 2 1 0 0 1 1 1 0
1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 U 0
1 1 0 0 1 1 1 0 0 0
M 1 0 0 1 M 1 0 0 0
Total Reward for current iteration: 1
```

```
Reward for current iteration: 1
Current Board:
0 0 0 1 1 1 . 0 0 0
0 0 0 1 M 1 0 0 0 0
0 0 0 U . 2 0 0 1 1
1 2 2 2 M 1 1 1 2 M
2 M M . 1 1 1 M 2 1
M 3 2 1 0 0 1 1 . 0
1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 . 0
1 1 0 . 1 1 1 . 0 0
M 1 0 0 1 M 1 0 . 0
Total Reward for current iteration: 9
Game over! You hit a mine.
```

PennState

Evaluations – Performance

- The agent solved Minesweeper puzzles of smaller sizes
  - Biggest puzzle solved: 9x9 with 9 mines
- Clear drop off in performance across increasing board sizes
  - This is because as state spaces increase, simple Q-learning demands more memory to store all state-action pairs [4]
- The agent can adjust to updated values to parameters in the Bellman's equation
- Experienced human players are able to outperform our agent

PennState

Discussion of Results

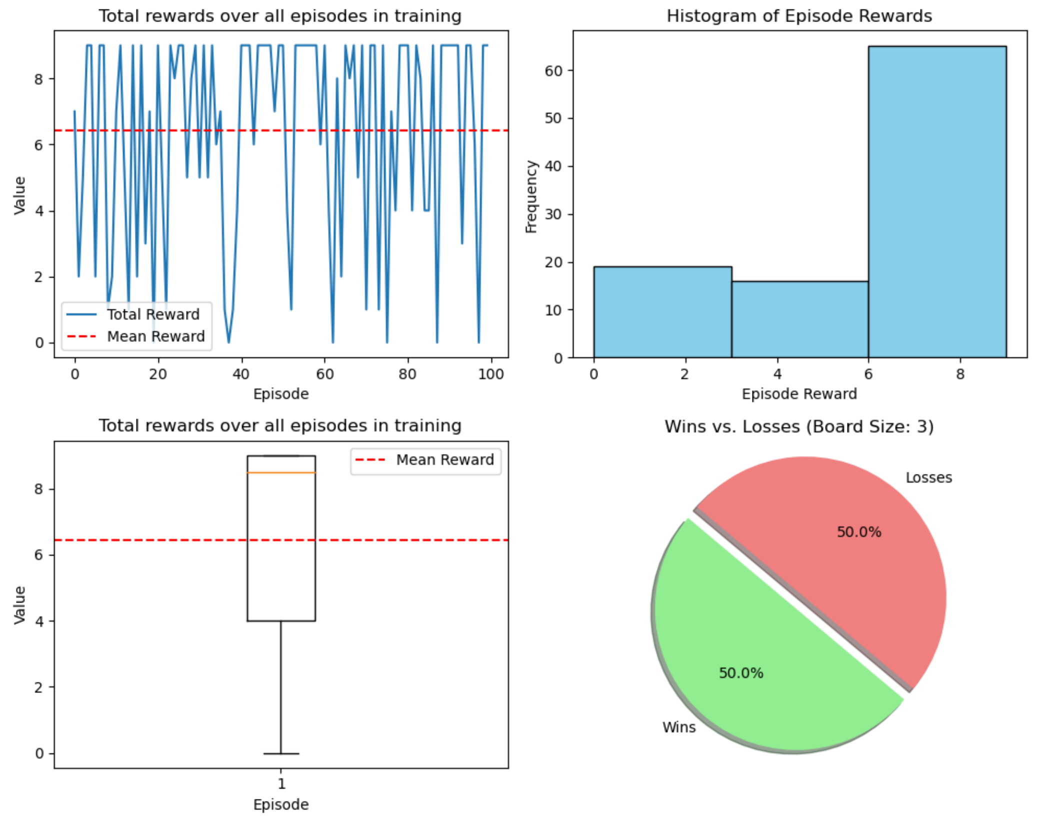
- Our agent struggled as the boards got increasingly bigger
- With more research and testing, we could assign reward values to each action that will enable the agent to learn how to better solve the puzzles
- Tuning the learning rate and discount factor parameters will help the agent solves puzzles by allowing new information to override old and how greedily it should take actions
- Deep Q-Learning or Reinforcement Q-Learning-based Deep Neural Network would have provided better performance because they utilize a model to train from and are more adept at handling complex state spaces [4]

PennState

Results Dashboard

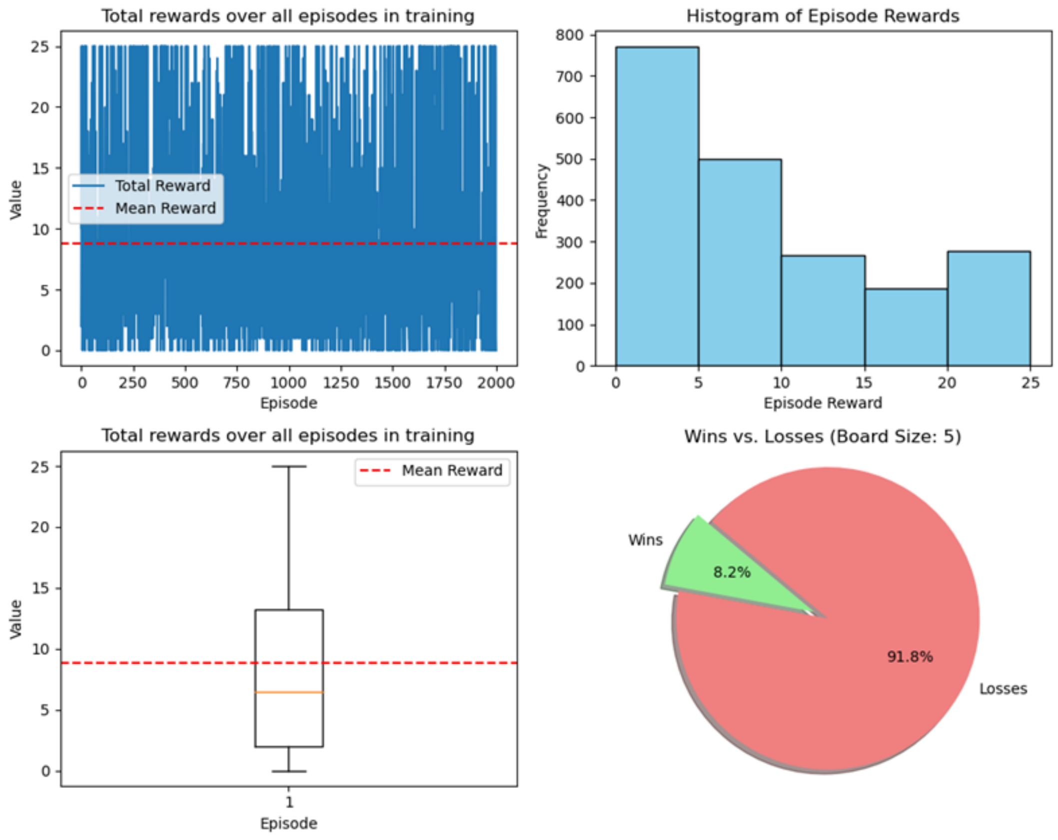
Four visualizations

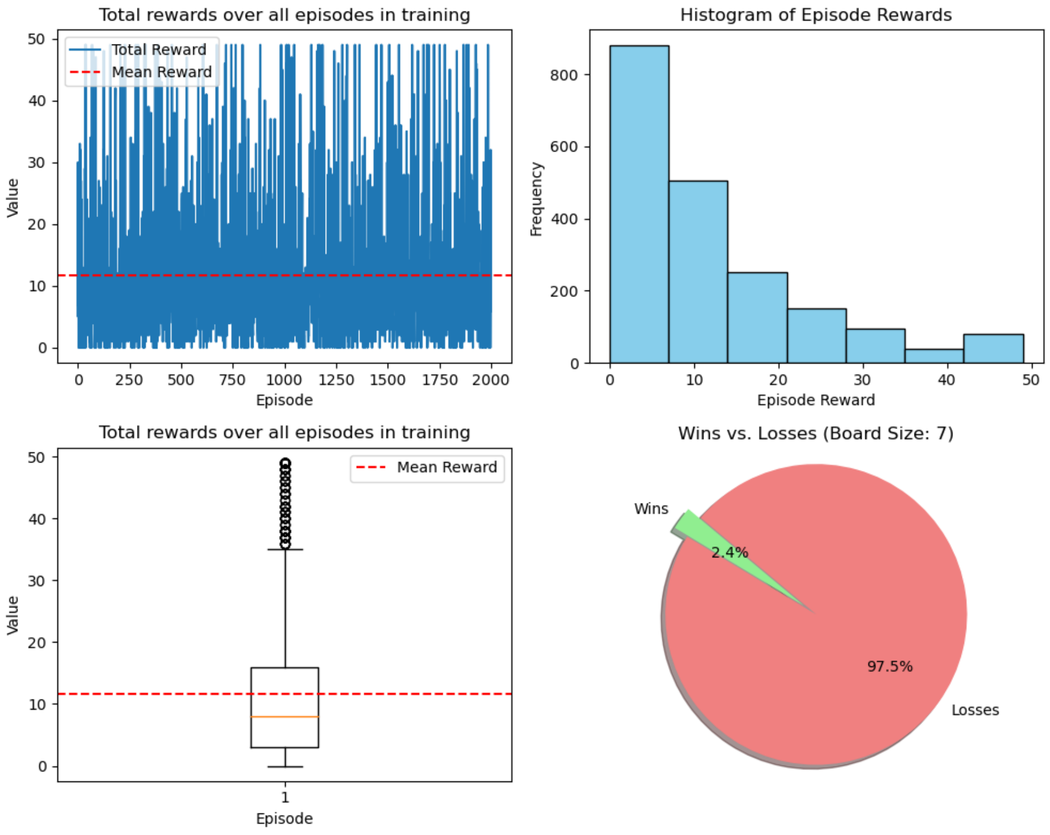
1. Line chart showing rewards over each successive episode
2. Histogram showing reward distribution
3. Box and whisker showing reward distribution
4. Pie chart showing percentage of puzzles solved



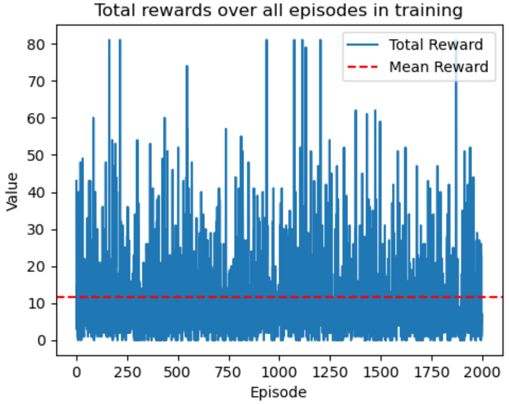
Dashboard for our agent ran against a 3x3 board

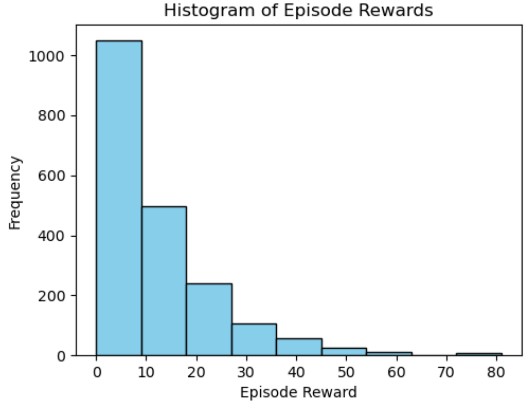
Results Dashboard: Board Sizes 5 & 7

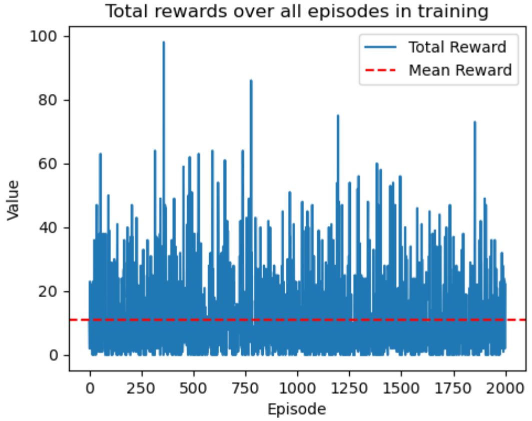


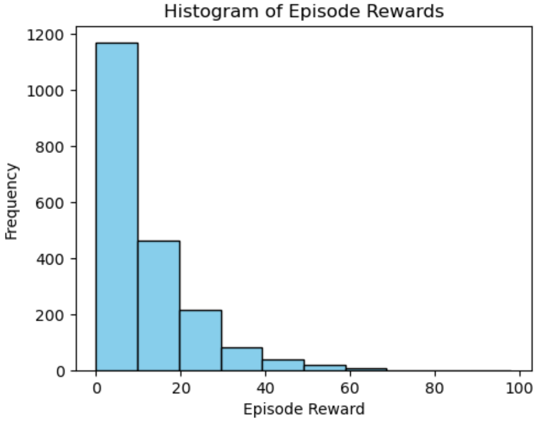


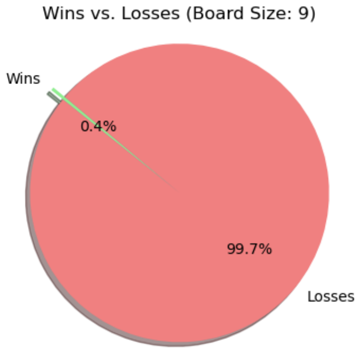
Results Dashboard: Board Sizes 9 & 10

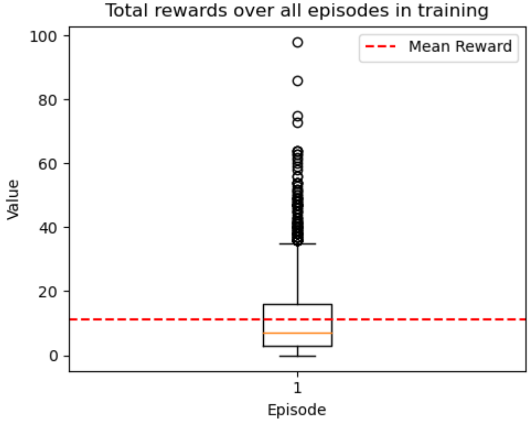


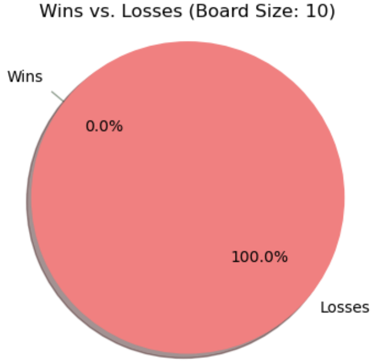












Conclusion

- Our team implemented a Q-learning agent in python to solve Minesweeper puzzles
- Future Direction
  - Continue testing and research to determine best rewards for each action
  - Continue testing and research to tune Bellman's equation parameters
    - Learning Rate
    - Discount Factor
  - Implement safe first click feature
  - Evolve agent to a Deep Q Network
    - We expect this agent to outperform our current simple Q-learning agent
    - Better accounts for larger state spaces

PennState

References

[1] Lin, et al. "Using a Reinforcement Q-Learning-Based Deep Neural Network for Playing Video Games." *Electronics*, vol. 8, no. 10, 2019, p. 1128.

[2] *Minesweeper Online*, minesweeper.online/.

[3] Russell, Stuart, and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Createspace Independent Publishing Platform, 2016.

[4] Souchleris, Konstantinos, et al. "Reinforcement Learning in Game Industry—Review, Prospects and Challenges." *Applied Sciences*, vol. 13, no. 4, 2023, p. 2443.

PennState