GridWorld Q-Learning Agent

Daksh M Jain
240319

Overview

This project implements a custom GridWorld environment inspired by the Harry Potter universe, where Harry must reach the Triwizard Cup while avoiding a Death Eater. Harry here acts as a Q learning agent and is trained using tabular methods with advanced reward shaping and state abstraction techniques.

Approach

- The environment is initialized with randomly placed positions of Harry, the Death Eater, and the Cup, ensuring all paths are valid using BFS.
- State abstraction is used where each state is encoded as a tuple of directional signs and proximity buckets between Harry, the Cup, and the Death Eater.
- Reward shaping includes:
  - Positive reward for reducing distance to the Cup.
  - Negative reward for reducing distance to the Death Eater.
  - Wall penalty based on local cell connectivity.
  - Backtracking penalty for loops.
  - Small reward shaping term proportional to step number.
- Death Eater moves optimally toward Harry using BFS at each step.
- Harry is trained using tabular Q-learning with epsilon-greedy action selection.

Assumptions

- The maze is grid-based with 'X' marking walls and passable cells marked as empty.
- All agents (Harry and the Death Eater) can move in four directions (up, down, left, right).

- Training episodes are reset only if either Harry reaches the cup before the Death Eater reaches him or the Death Eater reaches Harry before in turn reaches the Cup.
- The distance functions and reward shaping are computed using Euclidean distance.
- State representation abstracts away exact coordinates and uses relative information for generalization.

How to Run

1. Ensure all dependencies are installed:

```
pip install numpy pygame opencv-python tqdm
```

2. Place the environment code and training loop in a Python file (e.g., `train.py`).
3. To run training:

```
env = GridWorld(maze[0])
Q, rewards, successes = train(env)

# Print final success rate
print("Final success rate: ", np.mean(successes[-100:]))
```

4. To visualize environment dynamics, use the `render()` function inside the episode loop (e.g., after each `step()` call).

Training Parameters

- Learning Rate (α): 0.175
- Discount Factor (γ): 0.995
- Epsilon Decay: 0.999
- Episodes: 5000
- Epsilon Range: $[1.0 \rightarrow 0.175]$