Reinforcement Learning for Game Playing: Pac-Man Environment

Riley McKinney, Nathan Dow, Zain Karim
CS 4375.001 – Introduction to Machine Learning
University of Texas at Dallas
Richardson, TX, USA

***Abstract*—This project proposes a reinforcement learning system for autonomous game playing using a custom Pac-Man environment. The goal is to train an agent to maximize its reward by learning optimal navigation strategies to collect pellets while avoiding an adversarial ghost. A tabular Q-Learning approach will first be implemented, followed by potential extensions into Deep Q-Networks (DQN) using PyTorch for scalable state representation. The project will evaluate agent performance based on cumulative rewards and convergence behavior across multiple training episodes.**

***Index Terms*—Reinforcement Learning, Q-Learning, Game AI, Pac-Man, Machine Learning**

I. Project Topic

The proposed topic is **Reinforcement Learning for Game Playing**. A simplified Pac-Man grid environment will serve as the testbed for developing and analyzing reinforcement learning algorithms. The environment will simulate the classical pursuit-evasion problem, where the Pac-Man agent learns to collect pellets while avoiding a chasing ghost.

II. Team Members

This project will be conducted by **Riley McKinney, Nathan Dow, and Zain Karim**.

III. Technique / Algorithm

The planned technique is **Q-Learning**, a model-free reinforcement learning algorithm based on temporal difference updates:

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

The agent will follow an ϵ-greedy policy to balance exploration and exploitation. After verifying the basic Q-Learning setup, the project may extend to a **Deep Q-Network (DQN)** implemented with PyTorch to handle larger or more complex state spaces.

IV. Dataset / Environment Details

This project will not rely on a static dataset. Instead, a **procedurally generated environment** will serve as the data source for learning:

- **Environment:** 10×10 grid world with five randomly placed pellets.
- **Agents:** Pac-Man (runner) and Ghost (seeker).
- **State Space:** Encoded as discrete position pairs for both agents, yielding 10,000 total states.
- **Actions:** Four possible moves (up, down, left, right).
- **Reward Function:** $+1$ for eating a pellet, $+20$ for clearing all pellets, -10 for being caught, and -0.01 per time step.

Each episode will start with randomized pellet locations to encourage diverse exploration and learning stability.

V. Coding Language / Tools

- **Language:** Python 3
- **Libraries:** NumPy, Matplotlib, PyGame (for visualization), and PyTorch (for future DQN work)
- **Planned Modules:**
  - `environment.py` — Defines the Pac-Man grid and environment logic.
  - `agent.py` — Implements Q-Learning update and policy functions.
  - `main.py` — Training loop and episodic simulation.
  - `utils.py` — Plotting reward curves and logging results.

VI. Expected Results

The project aims to demonstrate the ability of a Q-Learning agent to improve cumulative rewards over successive episodes, showing evidence of learned strategies. Expected outcomes include:

- A working RL simulation environment with interactive visualization.
- Reward convergence curves plotted across episodes.
- A trained policy where Pac-Man efficiently collects pellets and avoids the ghost.

References

[1] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[3] IEEE, "Conference Templates," 2025. [Online]. Available: https://www.ieee.org/conferences/publishing/templates.html