Artificial Intelligence Project Report - Pong Game

GROUP 25 :- CHELLA SUDEEP , MALI RAJESH , R.ADARSH , B.RUSHYENDRA

1 Project Overview

- This project presents a Pong game implemented in Python using the Pygame library.
- The game includes a Q-learning algorithm to train two agents to play against each other.
- The primary objective is for the agents to learn optimal strategies for playing Pong and adapt their actions based on the rewards obtained during the game.

2 Introduction

Pong is a classic two-dimensional sports game. The game consists of two paddles, one on each side of screen, and a ball that bounces between them. The objective is to maneuver your paddle to hit ball past your opponent's paddle.

Paddles can move only in vertical direction to intercept the ball. When the ball collides with a paddle, it changes direction. If a player misses the ball and it passes their paddle, the opposing player scores a point.

Here, both paddles move based on the implemented AI algorithm. The AI algorithm we used is Q-learning algorithm, a form of reinforcement learning, to make decisions about the paddle movement based on learning from its interactions with the ball. The Q-learning algorithm updates its "knowledge" based on rewards obtained from hitting or missing the ball. The game then runs, and the overall average scores of the AI agent against itself are plotted over multiple episodes of play.

3 Code Structure

- **GameParameters Class :** The GameParameters class encapsulates essential parameters for the game, such as window dimensions, paddle and ball sizes, frames per second (FPS), and color definitions. This class ensures a centralized and organized way of managing the various parameters used throughout the game.
- **QLearning Class :** The QLearning class represents the Q-learning algorithm, consisting of methods for choosing actions, updating Q-values, and initialization.
  - In the __init__ method, the Q-table is initialized as a NumPy array with zeros. The size of the table is determined by the height of the game window (height // 10) and the number of possible actions (3 actions: move up, stay, move down).
  - The choose_action method implements the exploration-exploitation strategy. It decides whether the agent should explore (choose a random action) or exploit (choose the action with the highest Q-value for the current state).
  - The update_Q_values method is responsible for updating the Q-values based on the Q-learning update equation. It incorporates the current state, action taken, received reward, and the next state into the Q-values.
  - Key parameters include the learning rate (alpha), discount factor (gamma), and exploration-exploitation tradeoff (epsilon).These parameters are critical in fine-tuning the behavior of the Q-learning algorithm.
- **PongGame Class :** The PongGame class defines the Pong game environment. It includes game elements such as paddles and a ball, as well as methods for updating game state, handling collisions, and rendering the game on the screen.

- The get_state method is responsible for converting the current paddle position into a state representation that the Q-learning algorithm can use. This discretization of the continuous state space allows the algorithm to operate efficiently.
- The update_game_elements method handles collisions, updates scores, and ensures that game elements stay within the bounds of the game window.
- The draw_game_elements method uses Pygame to render the current state of the game on the screen. It displays paddles, the ball, and score information.
- The run_game method is the main game loop. It continually updates game elements, interacts with the Q-learning algorithm, and visualizes the game state. The loop continues upto 20000 episodes.

- **Main Script (if __name__ == "__main__") :** The main script initializes game parameters and Q-learning parameters, creates an instance of the PongGame class, and runs the Pong game loop. During each iteration, the Q-learning algorithm is applied to update agent strategies based on the game's current state and rewards.

4 Q-Learning Algorithm

4.1 Q-Table

- The Q-learning algorithm utilizes a Q-table, implemented as a NumPy array, to store Q-values for state-action pairs. The Q-values represent the expected cumulative rewards for taking a specific action in a given state.

4.2 Learning Process

- **Action Selection :** Agents choose actions based on an epsilon-greedy strategy, balancing exploration (random actions) and exploitation (choosing the best-known action).The choose_action method implements the exploration-exploitation strategy. With probability epsilon, a random action is chosen (exploration). Otherwise, the action with the maximum Q-value for the current state is selected (exploitation).

- **State Update :** After an action is taken, the game state is updated, and rewards are determined based on the game dynamics.

- **Q-value update :** Q-values are updated using the Q-learning update equation, incorporating the received reward and the maximum Q-value of the next state.

$$Q(s,a) = (1-\alpha) \cdot Q(s,a) + \alpha \cdot (R + \gamma \cdot max_{a'} Q(s^{'}, a^{'}))$$

  - Q(s,a) is the Q-value for state-action pair (s,a).
  - α is the learning rate, controlling the impact of new information ($0 \leq \alpha \leq 1$)
  - R is the immediate reward for taking action a in state s.
  - γ is the discount factor determining importance of future rewards.
  - $Q(s^{'}, a^{'})$ is the maximum Q-value for any action $a^{'}$ in the next state $s^{'}$

- **Training Iterations :** The overall average scores of the agents are calculated every 10 episodes, and the training progress is visualized using Matplotlib.

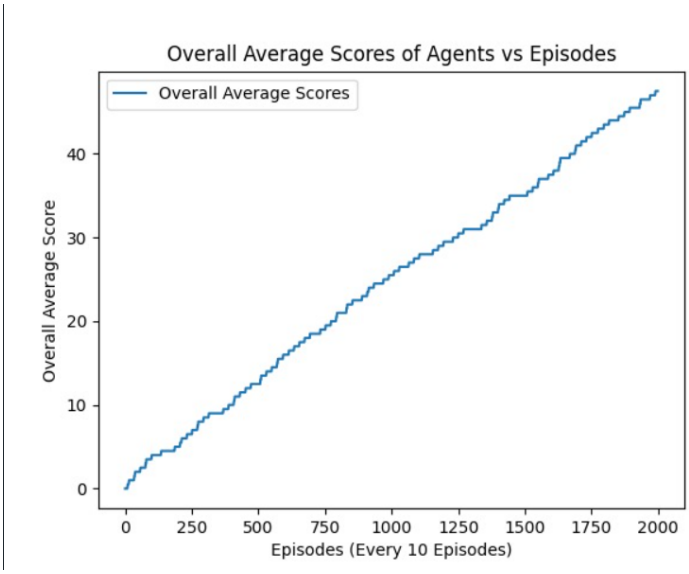

Figure 1: Graph between overall average score and number of episodes

5 Conclusion

This project helps to understand and implement reinforcement learning in the context of a simple Pong game. The Q-learning algorithm allows agents to learn effective strategies over time through a combination of exploration and exploitation. Users can experiment with various hyperparameters and game dynamics to observe their impact on the learning process.Further enhancements and customizations can be made to extend the project's functionality and explore additional reinforcement learning techniques.