Mini Project: Reinforcement Learning-based AI for Snake Game

Mihika Dravid – 112003038

Mihika Sanghvi – 112003084

Brief Description:

Our project aims to develop an artificial intelligence (AI) agent capable of playing the classic Snake game using reinforcement learning techniques. The AI agent will learn to navigate the game environment, avoid obstacles, and consume food to maximize its score. We will implement a Q-learning algorithm to train the AI agent, allowing it to learn optimal strategies through trial and error.

We will be implementing a Snake game using Reinforcement Learning to teach an AI how to play it. We plan on using RL in the following ways :

Game Environment: The game environment is created using Pygame, where the snake moves around the screen, eating food to grow while avoiding collisions with walls and itself.

State Representation: The state of the game is represented by the position of the snake's head, the positions of its body segments, and the position of the food.

Action Space: The AI agent can take four actions: move the snake up, down, left, or right.

Reward System: The agent receives a positive reward for eating food and a negative reward for colliding with walls or itself. The goal of the agent is to maximize its cumulative reward over time.

Q Learning Concept:

Q Value = Quality of action

1. Init Q Value
2. Choose action from model prediction
3. Perform the action
4. Measure reward
5. Update Q value
   Bellman equation used for updating Q value.
   Formula is :

$$\text{New } Q(s,a) = \text{old } Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - \text{old } Q(s,a) \right]$$

where s = state , a = action , s' = next state, a' = next action, y = discount rate, r = reward, alpha = learning rate (0.001)

Loss function used is mean square error : $(Q\_new - Q)^2$

Agent.py

Uses game.py and model.py functions. It trains the model and obtains the reward to a particular action moving from one state to another.

Training:
State = get_state(game)
action = get_move(state) ; using model
reward, game_over, score = game.play_step(action)
new_state = get_state(game)
remember() #uses this to remember the action done so that it can be used in training for the future steps.

model.train()

game.py

The game.py has multiple helper functions. The play_step(action) function is important and is used in agent.py.

model.py

The model used is a basic feedforward network.
It has a input layer (states), hidden layer, and a output layer (actions). The input layer has 11 nodes, hidden layer has 256 nodes and output layer has 3 nodes.
The model is called for making predictions.

Reward:

eat food : + 10
game over :-10
else: 0

Actions:

Straight : [1,0,0]
Right : [0,1,0]
Left : [0,0,1]

States: (Total 11 states)

[danger straight, danger left, danger right,
direction left, direction right, direction up,
direction down, food left, food right, food up,