Reinforcement Learning In Mouse Maze Experiment

Neeve Kadosh
Signal Processing & Pattern Recognition Laboratory, Department of Electrical & Computer Engineering
Rowan University, Glassboro, NJ 08028
kadsohn3@students.rowan.edu

Signal Processing and Pattern Recognition Laboratory

Introduction & Motivation

- ***Reinforcement Learning*** – *Teaching an agent in an unknown environment to achieve an unknown goal. The agent is only aware of the actions it can do. Overtime, the agent learns optimal actions to take based off the rewards given from each action until the agent reaches its objective and learns what the objective is.*
- ***Objective*** – Use q-learning to teach a mouse in a maze to find the cheese by taking an optimal path of which produces the least number of moves.
- ***Q-Learning*** – A method of improving the behavior of an agent. A q-table will optimize the best action based off a given state in the environment.
  - **Reward** – Positive rewards if the agent achieves its goal, else varying negative rewards until the agent succeeds. Reward given after each action.
  - **Action** – The agent will conduct an action (up, down, right, left) in an environment.
  - **State** – The location of the agent in the environment.
- ***Practical Examples***
  - ***Training Robots*** – Supervised by humans, they can learn through trial and error.
  - ***Advanced Game Play*** – Improve gaming experience and achieve super-human performance[2].

Modeling Q-Learning: Q [1]

- ***Exploitation vs. Exploration*** – The agent may learn poorly, to adjust the models learning an epsilon, ε, is added where $0 \leq \varepsilon \leq 1$. If ε=20%, the model will explore or conduct random actions 20% of the time, otherwise, the model will exploit the q-table to find the optimal action based off the state and reward 90% of the time. For this experiment, ε=80% to begin and was reduced to 20% overtime meaning the agent would explore in the beginning and in the end would be exploiting the q-table more often.
- ***Reward*** – Until the agent find the cheese the agent gets a negative reward for every move it makes. Moving into free cell that has not been visited yet is a small negative reward. Moving into an already visited location is double the negative reward of moving into a free cell. Moving into a wall is a larger negative reward. Moving into a state that the agent was just in is the largest negative reward. Finding the cheese is the only positive reward.
- ***Action*** – The mouse can move up, down, right, and left. Actions are only implemented if the moves are valid. An agent can conduct any action in any state. If the action is invalid, the agent will obtain the designated reward and will remain in the same state.
- ***State*** – Any position the mouse can enter in the maze environment. A wall is not a state because it cannot enter.
- ***Learning rate, α*** – The learning rate will affect how much impact the reward will have on the optimal move to make.
- ***Gamma, γ*** – Discount factor quantifies importance to future rewards
- ***Q-Table, Q*** – Utilizes all above factors to determine an optimal action in each state that will lead to the best reward. The formula is outlined in the algorithm.

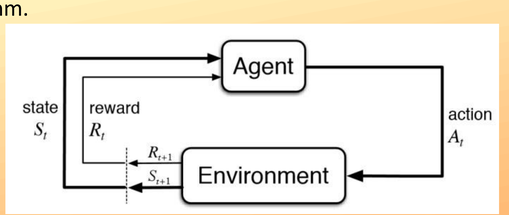

Figure 1. Block diagram of the q-learning algorithm implemented.

Proposed Algorithm

Algorithm: RL model to optimize moves for mouse to find cheese

Require: **Q** learning algorithm capable of utilizing rewards, states, and actions to optimize next move,
Reward provided to agent based off new state and action conducted,
State updates environment location,
Action played by agent determined by exploitation or Exploration

Input: Maze environment

Q Formula: **Q** = zeros((number_states, number_actions))
Q[old_state, action] = **Q**[old_state, action] + α * (reward + γ * max(**Q**[new_state, :]) – **Q**[old_state, action])

While $optimal\ path == False$:
A. **While** $winner == False$:
  1. Reset maze environment
  2. **if** $random.uniform(0, 1) < \varepsilon$:
     a. Explore maze
        i. Conduct **Action**
        ii. Obtain new **Reward** and new **State**
        iii. Calculate **Q**[state, action]
        iv. Update maze environment
  3. **else**:
     a. Exploit maze
        i. Calculate **Q**[state, action]
        ii. **Action** = max argument(**Q**[state, :])
        iii. Conduct **Action**
        iv. Obtain new **Reward** and new **State**
        v. Update maze environment
  4. **if** $game == 'Winner'$:
     a. End game because agent found the cheese
     b. winner = True
B. **if** $number\ moves > move\ limit$:
  1. Increase ε and α
C. **else**:
  1. Decrease and normalize ε and α
D. **if** $number\ moves < move\ stopping\ condition$:
  1. Agent found optimal path
  2. $optimal\ path = True$

Conclusions and Future Work

- ***Conclusions***
  - By developing a reinforcement learning model for maze solving, it becomes possible for an agent to learn the optimal path to find the best reward.
  - Modifying varying learning parameters and utilizing the q-table allowed the agent to perform in the least number of moves possible for the easy, medium, and hard difficulty maze environments.
- ***Future Work***
  - Apply reinforcement learning concepts to increasingly challenging problems.
  - Use built in keras-rl and tensorforce reinforcement learning toolboxes to enhance model performance.

[1] S. Gite, "Practical Reinforcement Learning – 02 Getting Started with Q-Learning," *Towards Data Science*, April 4 2017.
[2] A. Joy, "Pros and Cons of Reinforcement Learning," *Pythonista Planet, March 31 2019.*
[3] E. Santana, "Deep Reinforcement Learning for Maze Solving," np., nd.

Results from Algorithmic Model

- The q-learning algorithm was tested on 3 different level difficulty environments: easy, medium, and hard.
- As difficulty increased, number of games played increased because the mouse took longer to find the optimal route to find the delicious cheese.
- The agent begins each game at the same starting position, (0, 0) or the top left corner. The cheese is always at the same position, (number_rows-1, number_columns-1) or at the bottom right corner.
- Figure 2 represents the results of the experiment:
  - Easy maze environment: 9 moves with move limit at 10,000 moves
  - Medium maze environment: 25 moves with move limit at 50,000 moves
  - Hard maze environment: 51 moves with move capped at 1,000 moves

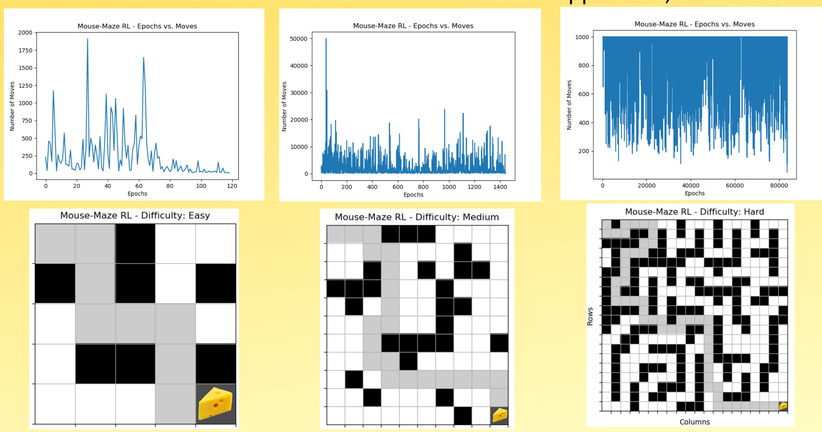

Figure 2. The first row or the 3 blue plots above represent the number of epochs or games played vs. the number of moves it took the mouse to find the cheese. The second row or 3 gray scale mazes below are the mazes and optimal moves it took the mouse to find the cheese. The first column is the easy difficulty: 5x5 maze. The second column is the medium difficulty: 10x10 maze. The third column is the very hard difficulty: 20x20 maze. The white boxes in the maze are free cells. The black boxes of the maze are walls or invalid cells. The gray boxes are visited cells the mouse has been to.

Ethical Issues

- The method in which an agent succeeds may be a nontraditional method done by humans.
- Rewards and punishments can exceed moral standards.

Standards and Constraints

- ***Standards***
  - NSPE Code of Ethics
  - PEP 8 Guidelines
- ***Constraints***
  - Model implemented from scratch
  - Hyperparameter tuning
  - Computer speed
  - Time to run the model