CA_1

Reinforcement Learning

AI-Driven Snake Game

July 2022

ISSUED BY
Amit Sharma

REPRESENTATIVE
Sunil Gauda
10595858

Introduction & Background

A snake game that plays itself using AI is implemented in various ways, with many recent implementations using RNN and LSTM. the implemented snake game in this project is based on improvement using algorithms to decide the flow of the games and reduce time and complexity.

AI-Driven Snake Game is developed in 3 iterations

1. Basic Snake Game - The basic game consists of 3 parts an **agent** to start the game and initiate the Markov decision process, a **model** file to implement the Q-Learning process, and a **game** file to implement the game itself using PyGame.
2. Hamiltonian Snake game - Using a mathematical model called Hamiltonian Path we have reduced the drawbacks of the base model. The improved model does not overexploit the greedy algorithm
3. Advanced Sanke game - Improving upon Hamiltonian Path to reduce the time taken to learn by using A* search algorithm

Formalizing as Markov Decision Process

The snake game process can be formulated as a Markov Decision Process, in the game the process implemented for all three Iterations are as follows.

1. S - The state of the basic game is automatically initiated from the location the snake is spawned at first, then the state is set by the **model** of the game, State has 11 values as follows
    - Danger Straight
    - Danger Right
    - Danger Left
    - Direction Left
    - Direction Right
    - Direction up
    - Direction Down
    - Food Left

- Food Right
- Food Up
- Food Down

2. A - Actions are Left, Right, UP, and Down and are decided by the **model** of the game which is clubbed as follows, No “BackTurn” as it risks eating its own tail this issue will be resolved in Hamiltonian and Advanced Implementations.
    - [1, 0 ,0] -> Straight
    - [0, 1, 0] -> right turn
    - [0 ,0, 1] -> left turn
3. P - The probability matrix for state transition is not used as the next state is calculated using Q-learning in determining the new state.
4. R - Reward is decided as if lost -10 if gain +10 and no reward when just playing
5. γ - Is used in the Q-Learning algorithm to find the new state

Reinforcement learning Algorithm

The following algorithms are used for the learning and optimization process of the game.

1. **Q-Learning Algorithm: To derive action for a particular state, Q-Learning is used as the base algorithm. The algorithm works perfectly with Markov's decision-making process to derive the optimal policy with respect to expected total rewards with every successive step taken. With the ability to handle stochastic transitions.**

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \Big)}^{\text{new value (temporal difference target)}} \overbrace{- \underbrace{Q(s_t, a_t)}_{\text{old value}}}^{\text{temporal difference}}$$

Fig 1. Q-Learning Algorithm Formula

2. **Hamiltonian Path: To optimize the basic model, we have used the hamiltonian path for the snake to traverse and derive the new Q-Value. The Hamiltonian path is a graph theory-based principle that derives the path by traveling through each node of the snake game only once. This reduces the chances of snake moving through obsolete paths again and again as the model learns which is the path that needs to be taken.**

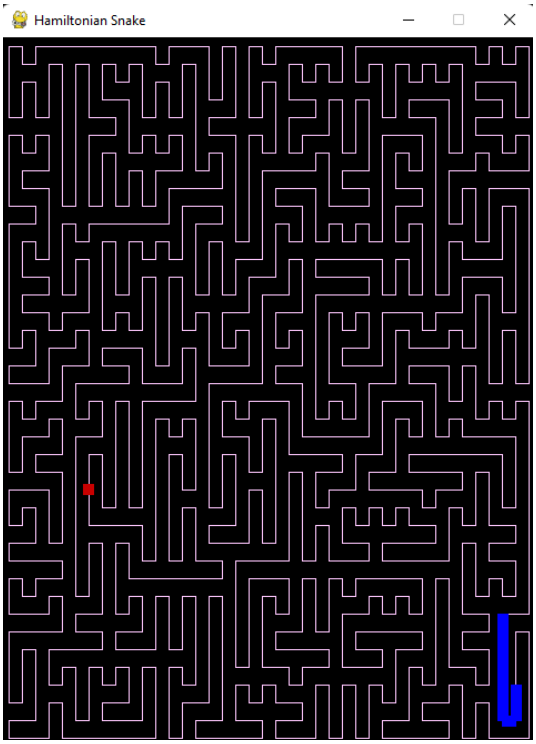

Fig 2. Hamiltonian path

3. **A* Search : The Graph traversal algorithms like A* provides completeness to the traversal of hamiltonian path, which in turn traversed through all paths to learn the optimal one, the search algorithm will help us reduce the time complexity of performance of the game by deriving the shortest path to the food and find appropriate Q-value.**

Parameter Choice and Exploration

1. **Parameters :**

a. In Q-learning the choice of parameters are as per the Markov Decision Process, but the complete value as demonstrated in Fig 1. The **Q value** is the basic parameter of the algorithm which helps predict the new value.
b. Path in Hamiltonion Cycle is important representation of the graph layout of the snake game , which the game uses to find the the food without eating its own tail
c. A* h(x) is the function that is run in the game to derive the shortest path.

2. **Exploration:**
    a. Base Model - Explores through the Q-value derived from the Q-learning algorithm, has chances of running into its own tail.
    b. Hamiltonion Model - Explores using Hamiltonian Path that is played out in the game layout, goes through many iterations to converge into the shortest path.
    c. Advanced Model - Search algorithm implemented in it helps find the shortest path to the food, thereby improving upon the previous models.

Implementation

1. **Basic Model :**
    a. Implementation : Base model is implemented using 3 parts, game, model, agent.
        i. Game : Game is a python implementation of snake suing pygame, which contains the UI of the game
        ii. Model : Model is the Q-learning algorithm implementation inside the game, it helps us determine the State, Action , Reward of the game
        iii. Agent : Helps moderate between Game and Model to make the process work from launching the game to add ing the new states.

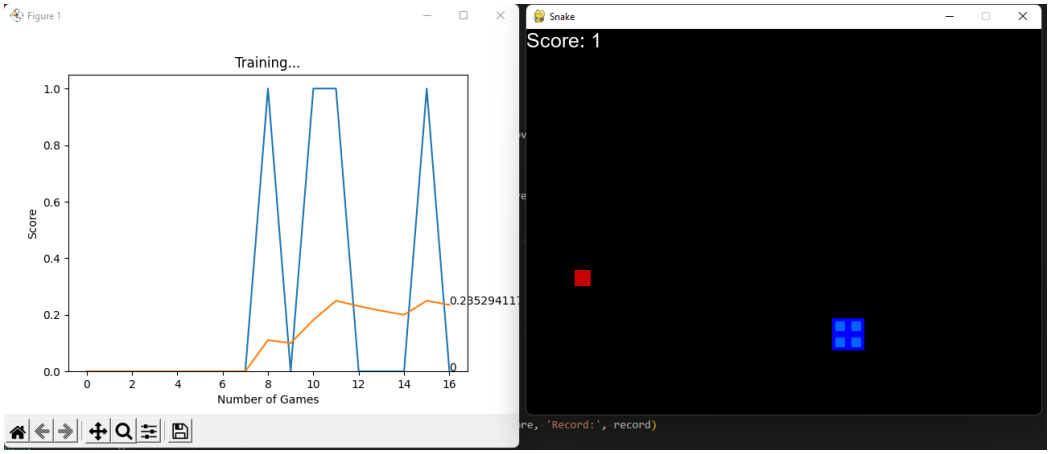

Fig 3. Basic Game

b. Drawbacks:
    i. Snake Eating its own tail
    ii. Large Traning Time
    iii. No Shortest path detection

2. Hamiltonian Model :

a. Implementation: The Hamiltonian path is drawn in the game using the **cycle.py** file, the snakes follow the path and learn a new shortest path.

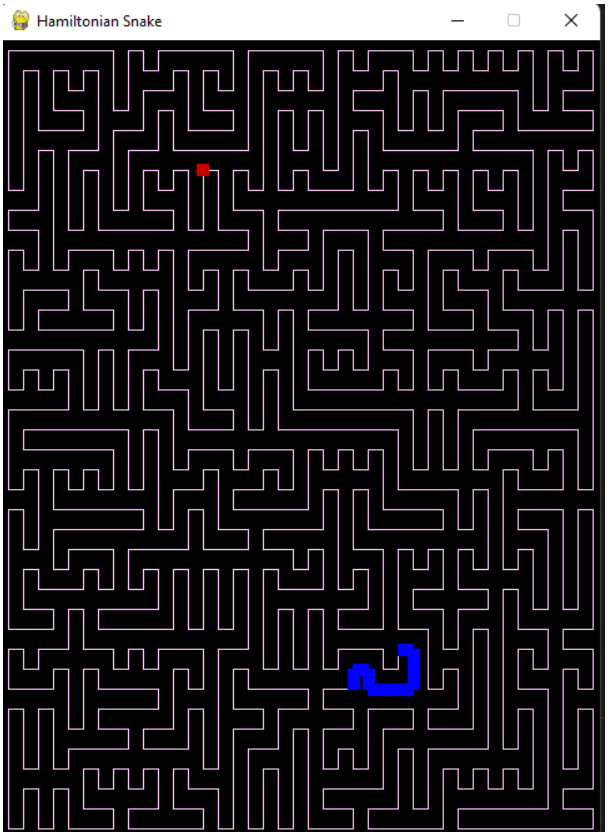

Fig 4. Hamiltonion Path

b. Drawbacks :
    i. No Shortest path detection
    ii. Time taken is large because of traversed paths

3. Advanced Model :

a. Implementation : has impelted shortest path algorithm for greedy based path selection.

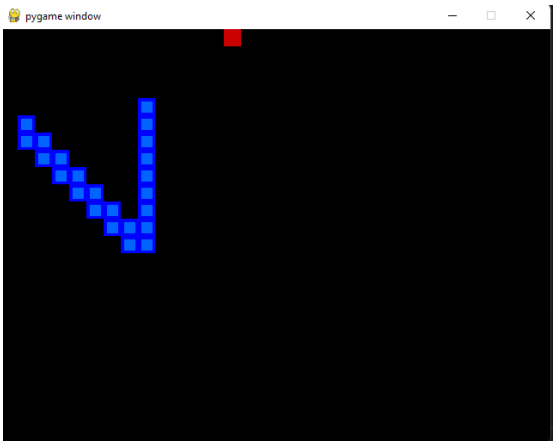

Fig 5. Advanced Implementation

b. Drawbacks: None

A Parameter to Study

Q-Learning, is an algorithm to implement model less learning for the game, as shown in the formula on fig 1we can observe the Q value is derived with the help of Previous Q value, predicted Q- value and old Q- value, with the multiple of learning rate and discount factor. The basic cycle is as follows.

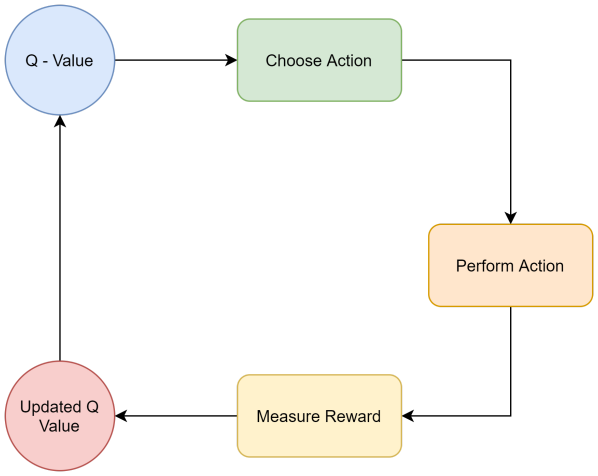

Fig 6. Basic Cycle

The Q-learning aspect of the model works on state and action combination as the formula states. The new state is the new Value for Q and thus the algorithm does not require any specific model, as it takes only the previous state to derive the new state.

Conclusion

The Conclusion of the experiment notes the following points.

1. Basic Agnet takes a huge amount of time as much as 3 hours or more to derive a conclusive action with maximum success, then too it has fear to run into itself

2. Hamiltonion implementation takes less time than basic due to drafted path in the game.
3. Advanced game targets directly towards food and takes the greedy approach using A* algorithm, which helps perform better in the game.

Refrences

1. Sprangers, O., Babuška, R., Nageshrao, S.P. and Lopes, G.A.D. (2015). Reinforcement Learning for Port-Hamiltonian Systems. *IEEE Transactions on Cybernetics*, [online] 45(5), pp.1017–1027. doi:10.1109/TCYB.2014.2343194.
2. Jang, B., Kim, M., Harerimana, G. and Kim, J.W. (2019). Q-Learning Algorithms: A Comprehensive Classification and Applications. IEEE Access, 7, pp.133653–133667. doi:10.1109/access.2019.2941229.
3. Wikipedia Contributors (2019). *Q-learning*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Q-learning.
4. GeeksforGeeks. (2021). *AI Driven Snake Game using Deep Q Learning*. [online] Available at: https://www.geeksforgeeks.org/ai-driven-snake-game-using-deep-q-learning/.