Snake game using Reinforcement Learning

Md Abdullah Al Symum
Id 18201007
School of Data and Sciences
Brac University
Dhaka, Bangladesh
md.abdullah.al.symum@g.bracu.ac.bd

***Abstract*—Machine learning techniques like reinforcement learning teach an agent to base decisions on information it receives from the environment around it. The renowned game of Snake is one example of how effectively this approach has worked in game development. The snake's basic movements, evaluation skills, and general expertise can all be improved through reinforcement learning. The agent may improve its methods for making decisions while gaining knowledge from its own observations through using this approach. Agents earned higher and survived competitors better as an outcome of the application of reinforcement learning in the snake game. The future use of this technology in other video games as well as applications appears promising.**

***Index Terms*—Tentative Keyword 1, Tentative Keyword 2**

I. Introduction

There are a lot of machine learning algorithms we can see on the internet that are profound in giving results based on their training data and test data, but among them, Reinforcement learning is a kind of machine learning that helps an agent, through trial and error, make the best possible decision in a specific scenario. Through this approach, an agent can learn to make decisions that maximize a specific reward or outcome.

The basic premise of reinforcement learning is that an agent receives feedback from an environment in the form of rewards or penalties for its actions. The agent proceeds to employ this information to adjust its actions and make further choices that are more likely to result in an improved outcome.

In contrast to other types of machine learning, reinforcement learning does not rely on pre-labeled data or supervised learning. Instead, an agent learns by interacting with its environment and continuously receiving feedback. This allows it to learn and improve with experience, adapting to changing circumstances and continually optimizing its decision-making processes.

One of the most notable applications of reinforcement learning is gameplay. Through reinforcement learning, an AI agent can learn to play games such as chess or go at a surprisingly high level of skill, often surpassing that of human players. Reinforcement learning is also used in robotics and autonomous systems, as it allows these systems to learn how to navigate and interact with their surroundings.

Overall, reinforcement learning is a potential tool for creating artificial intelligence that can learn and adapt to complicated and unpredictable situations. Its ability to improve decision-making processes through trial and error continues to make it a critical area of research and development in the field of machine learning.

Reitman et al. [**?**]

II. Literature Review

Reinforcement learning is an aspect of machine learning that allows a machine to figure out how to make intelligent choices by communicating with its surroundings. In Reinforcement Learning, an actor develops their behavior by performing operations in an environment while getting suggestions in the form of awards or penalties.

The key idea behind RL is to optimize the overall benefit gradually by learning an optimal approach that improves anticipated benefits. Some examples of RL applications include robotic control, gameplay, and autonomous driving.

The RL algorithm consists of defining a state space, an action space, a reward function, and the agent's policy. The agent learns by interacting with the environment, observing the current state, taking action, receiving a reward, and then updating its policy to maximize future rewards.

One of the challenges of RL consists of the overview-exploitation choices. In order for the machine to make the best possible decisions, it must try new actions, which implies collecting more information concerning the surrounding environment. However, it also needs to exploit the knowledge it has already gained to maximize rewards. This requires striking a delicate balance between exploration and exploitation.

There are several RL algorithms, such as Q-learning, SARSA, and actor-critic methods. Q-learning is a value-based algorithm commonly used to acquire the most effective behavior-value functions by modifying the Q-value of the present state action pairing based on the highest Q-value in the following region. SARSA is also a value-based algorithm

that acquires the best policy by updating the Q-value for a state-action pairing according to the next state-action pair following the policy. Actor-critic methods combine policy- and value-based approaches by acquiring information about both the actor's values and policies.

There are also Deep Reinforcement Learning (DRL) techniques that use neural networks as a tool to estimate the Q-function. DRL has achieved remarkable success in many applications, including Atari games and Go. But because of the data's non-stationarity and the relationship among samples, DRL is also susceptible to unpredictability issues.

In recent years, several advancements have been made in RL, such as multi-agent RL, meta RL, and model-based RL. Multi-agent RL enables agents to learn together by interacting with each other and the environment. Meta-RL enables agents to adapt quickly to new tasks by leveraging the knowledge learned from previous tasks. Model-based RL learns the model of the environment and uses it to plan the optimal policy.

A scalable alternative to Reinforcement Learning called "evolution strategies" (ES) is a black-box optimization technique that optimizes a large number of parameters simultaneously by randomly perturbing them and selecting the best candidates in each generation. ES can be used to solve complex reinforcement learning tasks, especially when the environment is non-differentiable or the rewards are sparse. ES is also shown to be more sample-efficient than traditional Reinforcement Learning algorithms. The effectiveness of ES in a variety of challenging environments, including Atari games, MuJoCo physics simulations, and humanoid robot locomotion. ES may be particularly useful for deep Reinforcement Learning tasks, where the high-dimensional state space makes traditional techniques more challenging.

Soft Actor-Critic (SAC), a new algorithm for deep Reinforcement Learning with a stochastic actor. SAC combines the benefits of maximum entropy Reinforcement Learning and stochastic actor-critic methods. It achieves state-of-the-art performance on several benchmark tasks in terms of sample efficiency and final performance. Unlike previous methods with similar properties, SAC has no hyperparameters that require tuning on the task at hand. The algorithm is based on finding a stochastic policy that maximizes the expected cumulative reward while also maximizing entropy. By doing so, the agent explores the environment more optimally and can avoid getting stuck in local optima. Furthermore, the algorithm uses a critic, a neural network that estimates the state-action value function in order to update the policy. The Soft Actor-Critic algorithm offers an efficient and easy-to-use approach for reinforcement learning that can lead to faster convergence and better performance. The combination of entropy and value-based reinforcement learning with a stochastic actor addresses several challenges in deep reinforcement learning, including exploration-exploitation tradeoffs and sample efficiency.

In conclusion, Reinforcement Learning is an effective machine learning model that makes it possible for an agent to figure out how to execute actions by engaging with their surroundings. RL algorithms require defining a state space, an action space, a reward function, and a policy. RL algorithms have achieved remarkable success in many applications, including robotic control, gameplay, and autonomous driving. DRL techniques use artificial neural networks to estimate the Q-function or policy, and recent advancements in RL include multi-agent RL, meta-RL, and model-based RL.

III. Research Methodology

Reinforcement learning represents a subfield of machine learning that analyzes how artificially intelligent agents need to act in an environment with the aim of maximizing the principle of cumulative bonuses. Snake is a well-known game that provides a strong basis for constructing an environment where intelligent agents may acquire knowledge. In this methodology, we will discuss how to create a snake game for reinforcement learning.

Environment Design: The first step in creating a Snake game for reinforcement learning is to design the game environment. It should have a 2-D grid where the snake can move, the snake's starting position, the food position, and the boundary of the game environment.

Agent Design: The second step is to design the agent. The agent is the entity that will discover how to navigate the game. In a Snake game, the agent is the Snake. The agent will learn to make decisions depending on the present circumstances of the challenge in order to maximize its score. At the beginning of the game, the snake has a default direction. The agent will act responsibly as the game evolves, and as the behaviors have an outcome, they will be awarded or punished.

Reward Function: The reward function determines the feedback that the agent gets for its actions. In a snake game, the agent will be rewarded for eating food and penalized for hitting the boundaries or hitting itself. The reward function should be designed in a way that encourages the agent to eat as much food as possible while avoiding hitting the boundaries or hitting itself.

State Representation: The state representation is a way of summarizing the relevant information about the current state of the game. It should be concise and capture all the relevant information. In a snake game, the state representation should include the current location of the snake's head and the food. It should also consider the presence of obstacles in the snake's path.

Action Space: The action space is the set of actions that the agent can take. The snake can make four possible moves – up, down, left, or right. These movements can be defined as the ”action space.”

Learning Algorithm: The reinforcement learning algorithm is the heart of the methodology. The most prominent reinforcement learning algorithm is Q-learning. Q-learning is a model-free reinforcement learning algorithm that learns to maximize the cumulative reward. It is a value-based algorithm that discovers the best approach by repeatedly modifying the Q-function. The Q-function indicates the probable future reward for a given condition and behavior.

Training Process: To train the agent, we simulate the game environment and let the agent play the game. At each new stage, the agent examines the present state while deciding a move in accordance with its current state of policy. After taking the step, the agent obtains an award. The Q-function gets updated using the reward that was acquired and a state-action pair.

By doing this, after a certain period of time, our snake can gradually perform better in the environment. Through continued action taken by the snake, it will improve its understanding of its surroundings, and after a while, it will eventually become an unbeatable snake.

IV. Results

The snake game is a classic game where a player controls a snake to collect food while avoiding obstacles and their own tail. Using Reinforcement Learning, agents have been developed that can learn to play and succeed in the snake game.

The agents in the Reinforcement Learning approach use a trial-and-error method of learning while playing the game. They learn through rewards and punishments, with rewards being given for positive actions like collecting food and punishments for negative actions like crashing the snake into the wall or the tail.

The agents start with no knowledge of how to play the game, but over time they learn through playing multiple games and collecting rewards. The agent’s main task is to maximize the reward it receives by taking actions that successfully collect food while avoiding punishment.

The results of using Reinforcement Learning in the snake game have been quite impressive. The agents learn to play the game effectively and score high. They learn to avoid obstacles and navigate the game board to collect food. Over time, the agents become better and better at the game, achieving higher scores and surviving longer. The Reinforcement Learning approach to playing the snake game has been successful in training agents to play the game effectively. The agents learn to adapt to the changing environment and make decisions determined by the current situation of the game. For instance, the agents learn to strategically move around the board to avoid obstacles and collect food, which in turn helps them get a higher overall reward.

Reinforcement Learning has shown a lot of promise in the game development industry, and the snake game is a perfect example of how this approach can be successfully applied. Through the use of Reinforcement Learning, agents can be taught to engage in the game like humans, adapting to the different situations encountered in the game.

Overall, the results of using Reinforcement Learning in the snake game have been highly effective, and the technology has shown great promise for use in other games and applications as well. With further developments and improvements in the algorithm, Reinforcement Learning can be a critical component of game development going forward.

A. Discussion

Reinforcement learning is a type of machine learning algorithm in which machines learn to make decisions based on the feedback they receive from the environment in which they operate. One practical implementation of this paradigm is in game development. The snake game is a classic game that can be suitably adapted for reinforcement learning purposes.

A game of snake is a finite, turn-based game that is suited for reinforcement learning at multiple levels. The game involves a snake (a collection of squares) that must maneuver around a rectangular field. The snake starts with a length of one, and whenever it eats the food (a single square), its length increases by one. The game ends when the snake collides with the wall or itself. In this context, reinforcement learning can be applied to:

Improve the snake’s basic movements: At the simplest level, reinforcement learning can be applied to teach the snake how to move around the field without running into the wall or itself. This is essentially an exploration problem, where the machine needs to explore different movements and learn which movements lead to a longer life. The reward function can be defined as the length of the snake. As the game progresses, the machine can learn from its experiences and improve its movements.

Improve the snake’s decision-making ability: Reinforcement Learning can be applied to teach the snake how to make better decisions when it comes to approaching food. At the core of this exercise is the problem

of pathfinding, where the machine needs to formulate a path from its current position to the food without running into any obstacles. The reward function can be defined as the length of the snake and its proximity to the food. As the game progresses, the machine can learn from its experiences and improve its decision-making abilities.

Improve the overall experience: Reinforcement Learning can be applied to improve the overall experience of the game. This can be done by introducing additional challenges to the game, such as varying sizes of the field or additional bonuses for the snake. The main challenge lies in defining the appropriate reward functions for these additional challenges. Once the appropriate reward functions have been defined, the machine can learn from its experiences and improve the overall experience of the game.

Reinforcement Learning provides a robust mechanism for improving the game of snakes. At its core, the technology is well-suited for teaching the snake how to make better decisions, improve its overall experience, and make the game more enjoyable to play. However, there are challenges associated with implementing reinforcement learning in the context of snake, such as defining the appropriate reward functions and balancing the reward system to ensure that the machine does not become too aggressive or passive. Nonetheless, with the right approach and focus, the game of snake can be greatly improved through reinforcement learning techniques.

V. Conclusion Future Work

The use of reinforced learning in the development of the Snake game has proven to be a promising approach. The game has successfully provided an enjoyable and engaging experience to players while also demonstrating the potential of artificial intelligence in the gaming industry.

The application of reinforced learning has enabled the Snake game to learn from its mistakes and improve its performance based on the rewards it receives from the environment. This has resulted in a game that is more challenging and responsive, providing players with a more immersive gaming experience.

Moreover, the Snake game has also highlighted the versatility of reinforced learning in the development of different types of games. The approach can be applied to create games that are more dynamic and adaptable to different skill levels and playing styles.

Finally, the use of reinforced learning in the Snake game can also have far-reaching implications for the gaming industry. Game developers can use this approach to create games that are more engaging, challenging, and personalized to each player, ultimately resulting in a more satisfying gaming experience.

However, there are still several challenges that need to be addressed in the application of reinforced learning in gaming. These include issues of scalability, interpretation of results, and the need for a more comprehensive understanding of player behavior and preferences.

Despite these challenges, the Snake game has demonstrated that reinforced learning has enormous potential in the gaming industry. As technology continues to advance, we can expect to see more sophisticated and immersive games that capitalize on the power of artificial intelligence to provide a more engaging and personalized gaming experience to players.

The following are the possible sectors in that Reinforced Learning can be improved:

Scaling up: One area of future work for reinforcement learning is to scale up the techniques for large-scale, complex environments. This is important because many real-world problems are complex, and reinforcement learning needs to be able to handle these large-scale situations.

Transfer learning: Transfer learning is the ability of a trained model to apply its knowledge to new, similar tasks. One area for future work is to develop transfer learning techniques that enable models to learn faster and transfer knowledge more effectively between tasks.

Explainability: Reinforcement learning models can be difficult to understand, as it can be hard to figure out why a model made a particular decision. Future work in this area may focus on developing techniques for making models more explainable and transparent.

Robustness: Robustness is the ability of a model to perform well in different environments and under different conditions. Future work may focus on developing reinforcement learning techniques that are more robust, such as models that can adapt to changing conditions or handle unexpected events.

Multi-agent systems: Multi-agent reinforcement learning is an area of research focusing on developing models that can learn in environments with multiple agents. Future work in this area may include developing more robust and scalable techniques for multi-agent learning.