SELF-LEARNING SNAKE GAME USING OPEN-AI GYM

Sumith Gannarapu, Jaya Harsha Thippana
University of Oklahoma
660 Parrington Oval, Norman, OK 73019

ABSTRACT

In this paper in contrast to other more advanced approaches in literature, we explored the application of reinforcement learning algorithms to make an agent learn how to play Snake game. By training the agent we showed comparisons among different state spaces and difference between two major reinforcement learning methods.

INTRODUCTION

Snake is the game popularized over the net with Nokia mobiles. It is played by a single player who controls the direction in which the snake is moving and tries to eat as much food as possible. Food would be spawned at random places. Eating a food will increase the size of the snake. The game becomes difficult as the length of the snake increases.

Reinforcement Learning is the most interesting area in Machine Learning. This teaches an agent on how to take the optimal decision based on random actions. We implemented two Reinforcement learning algorithms to train our agent which are Q-Learning & SARSA. The goal of our project is to train a self-learning snake game which performs much better than humans.

The specific characteristic of the game is as follows. The objective of each episode is to maximize the number of times agent eats the food. For every food eaten the agent has rewarded 500 points and for every time the snake is killed 100 points are subtracted. There are three possible actions possible for our agent at any given time: turn left, turn right, move straight.

In our work, we create a machine learning agent that plays snake, we hypothesize that by creating an agent using different state spaces and different Reinforcement Learning algorithms we will be able to achieve the goal proposed.

Fig. 1 Classic Nokia Snake Game

RELATED WORK

Bowei et al. (2016) discussed self-learning snake game using reinforcement learning and convolution neural network as the size of the state space is too large. They also demonstrated the comparison between Q-learning & SARSA and its performance using reduced state space. We also implemented Q-learning and SARSA for our project and compared the methodologies, but we didn't use neural networks which is different from the paper.

Zhepei et al. (2018) presented a refined reward mechanism to solve the delayed reward time by using the training gap strategy to remove all the improper training experiences. Authors implemented a Deep Q-Learning Network & compared the performance of humans with his model. As we didn't implement Deep Q-Learning Network, we only cared about their reward system. From this paper, we used the high-level information they provided for our state space development.

Peter (2017) talked about using best reward function to the snake. If the reward is zero when moving from one state to another state, it may feel turning in circles and staying in middle is the safest place which makes reward function

hard to get correct. Hence introducing a reward function which gives a slight penalty if snake stepping away from the food. This really helped us to make our agent smarter.

Ashley (2013) talked in detail about the Markov Decision Process and the epsilon-greedy strategy in detail. The author also provided different methodologies about the reward structure to achieve good results. From this paper, we highlighted the use of the high-level information they provided for the reward structure of our model.

METHODOLOGIES

Q-Learning

In Q-learning, an agent follows the process of Markov Decision Process and it only remembers the previous state information <state, action, reward, next state>. All of this information will help our agent to learn for future steps hence the agent tries to maximize the Q table value. We implemented epsilon-greedy because initially agent doesn't know how to play the game so based on this policy, agent explore more i.e. random actions. This enables the agent to exploit and make optimal actions later.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

SARSA (State Action Reward State Action)

SARSA is an on the policy-based algorithm and it estimates the policy being allowed <state, action, reward, next state, next action>. It figures the next action based on the policy and updates the Q-table.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

EXPERIMENTS & RESULTS

Experiment 1

Based on assumed State Space, Parameters and Reward Structure our agent initially explored more to learn the snake game and gradually exploration helped to exploit and take an optimal action. We also considered decay rate for epsilon i.e. epsilon value changes from 1 to 0.1 with a constant density of 0.01 and it holds at 0.1 when it reaches. From the learning curves for Q-learning (Fig 2.1) and SARSA (Fig 2.2) we could clearly observe in short run for few episodes Q-learning performed better than SARSA. As the number of episodes increases SARSA outperformed Q-Learning.

State Space	Parameters	Reward
Head position Food position Wall Positions (8) Left (2) Right (2) Up (2)	episodes = 100000 Learning rate = 0.2 Gamma = 0.4 Epsilon = 1 Decay rate = 0.01	Dead = -50 Food = 100 Step to food = 1 Away from food = -2

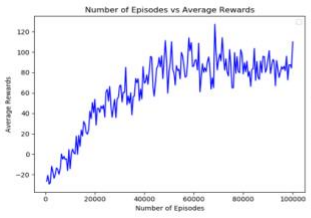

Fig. 2.1: Q-learning for Experiment 1

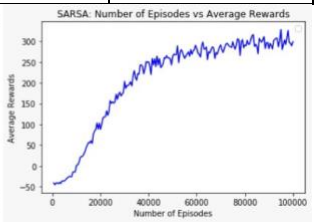

Fig. 2.2: SARSA for Experiment 1

Experiment 2

we removed the decay rate from experiment 1 and added fixed epsilon value and added the last tail of the snake to the state space. As the state space for this experiment is huge hence it needs more exploration than the first experiment. From the learning curves for Q-learning (Fig 3.1) and SARSA (Fig 3.2), during initial episodes Q-learning better than SARSA but when the episodes increases SARSA started performing well than Q-learning.

State Space	Parameters	Reward
Head position Food position Last tail position Wall Positions (8) Left (2) Right (2) Up (2)	episodes = 200000 Learning rate = 0.2 Gamma = 0.4 Epsilon = 1 Decay rate = 0.01	Dead = -50 Food = 100 Step to food = 1 Away from food = -2

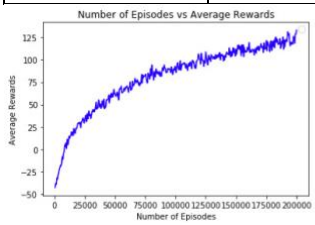

Fig. 3.1: Q-learning for Experiment 2

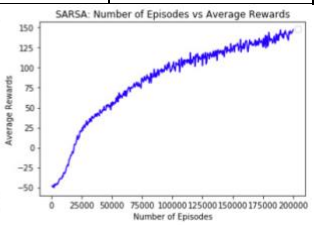

Fig. 3.2: SARSA for Experiment 2

Experiment 3

Experiment 3, We totally changed our state space from the other two experiments by only considering relative position from food to head, tail to head and wall positions. From the learning curves for Q-learning (Fig 4.1) and SARSA (Fig 4.2), we could observe both the algorithms did not perform well because the agent could not able learn from the features that we tried.

State Space	Parameters	Reward
Relative position Food position Wall Positions (8) Left (2) Right (2) Up (2)	episodes = 100000 Learning rate = 0.2 Gamma = 0.4 Epsilon = 1 Decay rate = 0.01	Dead = -50 Food = 100 Step to food = 1 Away from food = -2

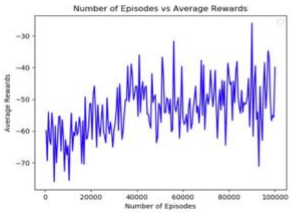

Fig. 4.1: Q-learning for Experiment 2

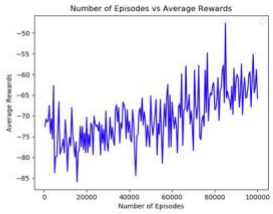

Fig. 4.2: SARSA for Experiment 2

With the presented experiments, it needs a lot of training to achieve a complete goal in order to the snake fills the entire grid. Experiment 3 didn't perform well with the reduced state space but there is a chance of improving state space with better reward system mechanism which is discussed Ashley (2013). For the Experiment 1, we considered the size of the grid n * n, the exact position of the food n * n and also considered position of the last tail n * n for the Experiment 2. The state space for Experiment 1 is more than n4 and state space for experiment 2 is more than n6, with the large state space learning rate would be very slow, hence to improve the learning rate we may need to implement Deep Q-network as suggested by Zhepei et at. (2018).

Q-Learning doesn't perform as good as SARSA because of Q-Learning takes the next action completely based on the Q values even with small learning rate present which often leads to bad decisions, Whereas SARSA is an on-policy algorithm which chooses its next state completely based on policy.

FUTURE WORK

To improve the learning rate of snake agent, deep Q-learning could be used. Reduced state space should be more accurate. There is a chance of improving the learning curve of SARSA by running a greater number of episodes for experiment 2. Also, perform different experiments on state spaces to figure out better state space.

CONCLUSION

In this project, we have developed two reinforcement algorithms Q-Learning and SARSA. We tried different types of experiments by changing State Space, Parameters and Reward Structure for a long period of episodes. Also, we compared two algorithms and justified that SARSA is performing better than Q-Learning. Due to large in state space, experiment 2 needs much more exploration to perform better for a long run of episodes. We also observed reduced State space did not work well for both of the algorithms.

REFERENCES

- Bowei Ma, Meng Tang and Jun Zhang, "Exploration of Reinforcement Learning to SNAKE", 2016.
- Zhepei Wei, Di Wang, Ming Zhang, Ah-Hwee Tan, Chunyan Miao, You Zhou, "Autonomous Agents in Snake Game via Deep Reinforcement Learning IEEE", *International Conference on Agents (ICA), 2018.*
- Peter Binggeser, "Designing AI: Solving Snake with Evolution", 2017. https://becominghuman.ai/designing-ai-solving-snake-with-evolution-f3dd6a9da867
- Ashley Jin and Joseph Tsai, "From Snake, Play All the Games", 2013.