Classic Snake Game Using Reinforcement Learning

Jasmin Wilson

Abstract

This report gives the specifics of reinforcement learning applied to the snake game using different training models like Q-learning, Sarsa and Sarsa (λ) . By adjusting the epsilon after certain episodes, we can see the improvement in the scores. Because this is a competitive snake, the best results is achieved by Q learning but SARSA learns quicker and gives the about same results as Q-Learning.

1. Introduction

In Reinforcement Learning, we are not given any prior information but agent learns about the environment by exploring and exploiting it using Q values. The environment tells the agent if that particular state-action is good or bad.

2. Domain

2.1. History

The snake game is a simple one player game. The goal is to have the snake eat as many food as possible while avoiding rashing into the walls or its own body. The more food is consumed, the longer the snake gets, and the more difficult the game becomes. The game should end when the snake hit the wall or hit himself. For every food consumed, player gets the point and continues until snake is the size of the grid.

2.2. PyGame

Pygame is a Python library designed for the creation of simple games. Some of the key features are 2d-rendering capabilities, user input acquisition and options for audio output. I used Pygame to create the environment and the visual aspects of the game. Instead of user input, the agent uses the state and action space from the environment to calculate rewards which is then rendered showing snakes movements. The reward structure is as follows-

1. Getting food is good. +1 points to the snake.
2. Hitting a wall or itself is bad. -10 reward to it.
3. Anything else has no reward

Figure 1. Snake Game with Pygame

2.3. Learning Methods

I am using 3 different model-free learning methods to train the snake and compare the scores and reward of each algorithm to see which one gives the best results in our game. But before that, lets understand what these methods do -

Q-Learning - This is a greedy method that updates the Qvalues of one state at a time and uses the immediate reward and maximum of next state and action.Because it updates one Q value at a time, the learning in the beginning is slow but with more training it converges.

SARSA - Sarsa does the learning very quickly as it updates Qvalues at each step and not wait till the end of the episodes. So it converges faster than Q-learning.

SARSA (λ) - Here, we use the concept of eligibility trace to keep trace of what steps gave the maximum reward and update all the state-action pair at each step. This is extremely powerful as we get the Q values of all state-action pair very quickly.

2.4. Environment

The states are discretized into 12 each having boolean values because it is impossible to store continuous states as a state, action pair for Q value.These 12 features makes the learning doable without overhead on normal CPU processing.. These features are - [Direction is left, Direction is right, Direction is up, Direction is down ,Food is left ,Food is right, Food is up, Food is down, Danger left, Danger right, Danger up, Danger down]

Even With 12 states, each having 2 boolean values, that means we have atleast 2 pow(12) unique states and 4 actions.

1. Unique States = 4,096
2. Unique Actions = 4
3. Total Q Table Entries = 16,348

3. Hypothesis

Hypothesis are the following:

1. Out of all learning methods, Q-Learning returns the most average score over n episodes.
2. It is better to change ϵ after n episodes early on than decaying ϵ
3. We can increase the SARSA (λ) score by adding step penalty

4. Experiments

4.1. Hypothesis 1

Figure 1 also shows that out of Q-Learning, SARSA and SARSA(Λ) with γ=0.95, α=0.01 and λ(trace decay) =0.4 ,Q-Learning gives us the highest score. It starts learning slow but eventually gives us the highest score. It makes sense because of its greedy behavior. SARSA learns quicker than Q-learning but plateaued after some episodes. Its interesting looking at SARSA(Λ) which updates the Q values of all states at once Fig 2 shows initial spike in that average score within 1000 episodes that shows it learned better than SARSA and Q-learning but after few episodes it makes no scores and this can be explained from the average reward in Fig 2.

We can see that SARSA (λ) has the highest average reward. It is because it rather have reward =0 than get -10. Recall that we give +1 for food and -10 for colliding with its body and wall, rest of the time- it has no reward. So the snake keeps looping until steps without food reaches a certain point and then terminates the episode. This way it keeps the average reward high and Q-learning and SARSA being more brave and taking the hit. For Sarsa (λ) to work, we need a complex reward system. (2) Table 1 shows the Mean and Standard Deviation of the each algorithm.

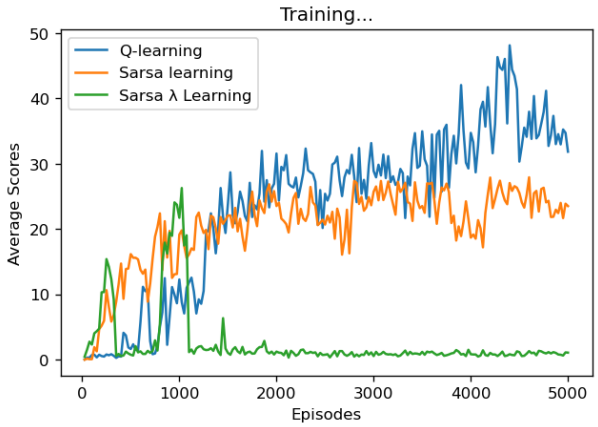

Figure 2. Average Score with Q-Learning, SARSA and SARSA (λ) using epsilon-greedy policy, using $\alpha = 0.1$, $\gamma = 0.9$, $\lambda = 0.4$ and decreasing to $\epsilon = 0.01$ after 1000 episodes

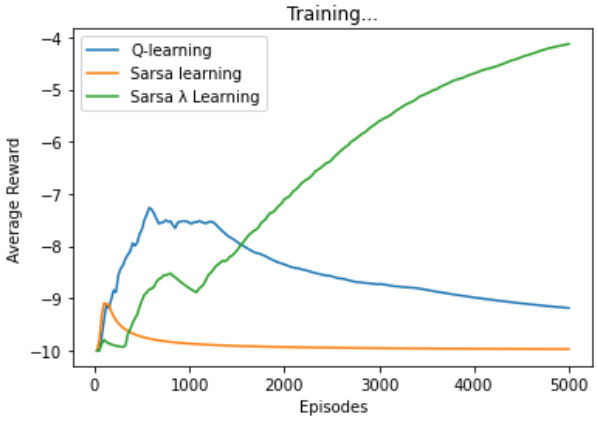

Figure 3. Average Reward with Q-Learning, SARSA and SARSA (λ) using epsilon-greedy policy, using $\alpha = 0.1$, $\gamma = 0.9$, $\lambda = 0.4$ and decreasing to $\epsilon = 0.01$ after 1000 episodes

4.2. Hypothesis 2

All the three models - Q-Learning, SARSA and SARSA(Λ) with hyper-parameters γ=0.95, α=0.01 and $\epsilon = 1$ and λ(trace decay)= 0.4, were trained using 3 ways. First with a well-knownn way, step decay which is calculated as max(γ* ϵ, min ϵ) . Second is static ie, there is no decay and ϵ remains at 0.4. Another way of mixing step + static is to decay normally until 1000 episodes and after that point to decrease it to min ϵ. In Fig 3 we can see, the average score at 10000 episodes for Q-Learning. Since the ϵ- greedy policy involves a lot of randomness, using the third method we can restrict the snake to always take the best move after certain

ALGORITHM	SCORE	MEAN	STD. DEV.
Q-LEARNING	25.430	-9.051	0.6833
SARSA	17.145	-9.675	0.255
SARSA (λ)	1.344	-5.336	1.289

Table 1. Mean total reward and standard deviation for 5000 episodes using various learning algorithms.

METHOD	MEAN	STD. DEV.
STEP	15.134	5.256
STATIC	2.588	0.580
STEP + STATIC	14.135	5.075

Table 2. Mean total reward and standard deviation for 5000 episodes using various learning algorithms.

exploration.Table 2 indicates that step method gives us the best score but with high variability in data.

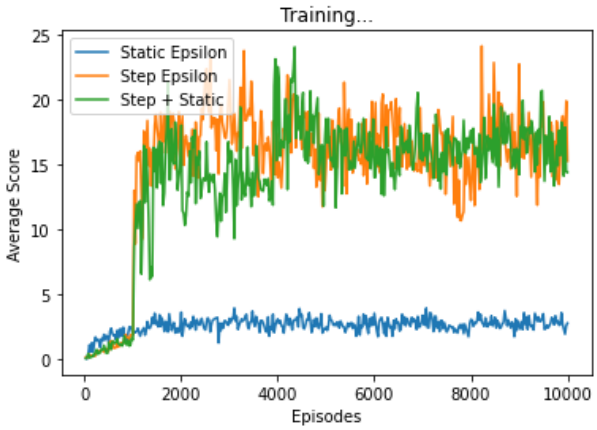

Figure 4. Average total score using methods STEP, STATIC, STEP + STATIC on Q-learning using epsilon-greedy policy, using α = 0.1, γ = 0.9, starting at ϵ = 1 for step, ϵ = 0.4 for static and ϵ = 1 and then decreasing it to 0.01 after 1000 episodes

4.3. Hypothesis 3

As mentioned in Hypothesis 1, SARSA (λ) makes no real scores with the current reward system hence we will experiment to see if it performs well when we punish when it doesn't eat within n steps. In Fig 4, we can see a slight change in the scores. It is getting better average scores yet mainting a higher reward than Q-Learning and Sarsa

5. Related Work

DeepMind creatively combined deep learning with reinforcement learning and came up with the well known deep Q-learning network (DQN) model(1).On a number of Atari

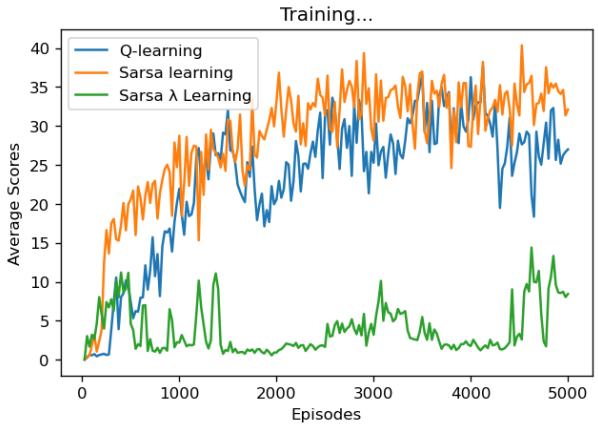

Figure 5. Average Score with Q-Learning, SARSA and SARSA (λ) using epsilon-greedy policy, using α = 0.1, γ = 0.9, λ = 0.4 and decreasing to ϵ = 0.01 after 1000 episodes with -10 reward for 1000 steps without food

ALGORITHM	SCORE	MEAN	STD. DEV.
Without n-step food penalty	1.344	-5.336	1.289
With n-step food penalty	3.80	-9.3686	0.0729

Table 3. Mean total reward and standard deviation(reward) for 5000 episodes using various learning algorithms for SARSA (λ)

titles, including the snake game, the DQN the agent was able to outperform humans on nearly 85 percent Breakout games. Mnih et al were the first ones to demonstrate effective ways of Reinforcement learning. It uses a replay buffer that stores the experiences at each step and then sampled across 3 networks to update the Q values. It uses similar approach as Sarsa λ that looks back 3X times that makes the learning super fast.

Another notable work is Zhepei Wei who designed reward mechanism such that as snake grows in size, the reward structure changes.(1) One of the interesting strategy mentioned was punishment for timing out. However in DQN, because of experience relay, it performs well but SARSA(λ) only knows till a certain point. Zhepei applied double triple experience relay affected their learning.

In addition to RL, other AI techniques have also been applied to the snake game, such as genetic algorithms (GA) and neuroevolution. For example,Jia-Fong Yeh, Pei-Hsiu Su, Shi-Heng Huang, Tsung-Che Chiang used a GA-based approach to evolve neural networks for playing the snake game. The authors found that their approach was able to learn effective strategies for the game, although it required a large number of evaluations.(3)

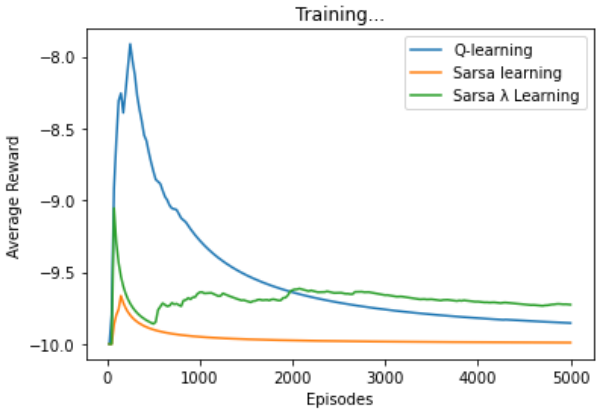

Figure 6. Average Reward with Q-Learning, SARSA and SARSA (λ) using epsilon-greedy policy, using $\alpha = 0.1$, $\gamma = 0.9$, $\lambda = 0.4$ and decreasing to $\epsilon = 0.01$ after 1000 episodes with -10 reward for 1000 steps without food

6. Contributions

All the finding in this paper is solely done by myself. From making the pygame and creating the environment to running the experiments.

7. Future Work

I have not been able to get a score more than 50 for 5000 episodes. The main reason being Snake has not learned to escape its own body. It wants to go to food but it will get stuck within itself. In future, I would like to use function approximation such as gradient descent to observe its performance.

8. Conclusion

I was successfully able to use Q-Learning, SARSA and SARSA (λ) and compare their learning on which performs better. The goal was to score many points which was achieved by Q-Learning, SARSA. In the view of learning, SARSA and SARSA (λ) learns the Qvalues quicker. Using the step decay of ϵ, we get maximum results. To improve the score of SARSA (λ), we added an additional penalty that improved the score by few.

(2)

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[2] Zhepei Wei, Di Wang, Ming Zhang, Ah-Hwee Tan, Chunyan Miao, and You Zhou. Autonomous agents in snake game via deep reinforcement learning. In *2018 IEEE International Conference on Agents (ICA)*, pages 20–25, 2018.

[3] Jia-Fong Yeh, Pei-Hsiu Su, Shi-Heng Huang, and Tsung-Che Chiang. Snake game ai: Movement rating functions and evolutionary algorithm-based optimization. In *2016 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 256–261, 2016.