Snake: Analysis of the Effects of Varying Environment Parameters on MDP's

Jerry Qin

December 13, 2021

Abstract

Reinforcement learning environments are modeled as Markov Decision Processes (MDP's) in which an agent has observable state and action spaces, as well as associated reward and value functions that it optimizes a policy over. As an agent interacts with it's environment it experiences positive and negative rewards associated with taking certain actions within a given state. In this paper, I discuss the importance of state space and reward function formulation on MDP's and POMDP's through analysis of the game Snake.

1 Introduction

Training reinforcement learning agents on an environment often involves sampling interactions within its environment by taking certain actions a in state s and optimizing its policy and value function in order to perform a specific task. As a result, the representation of the state space and reward function can have large effects in the observed convergence during agent training. While this effect has been highlighted with respect to model-based learning, it has not been explored in as much detail when considering model-free learning.

In this paper, I explore the effects of state-action space and reward function formulation on a model-free agent through the game Snake. I train a Deep Q-Network (DQN) agent with static hyperparameters as I vary the state space representation and reward function. Through this process, I illustrate the importance of proper reward and state space formulation and representation. Section 2 discusses some background information regarding the Snake environment and related work on the topic. Section 3 describes the DQN agent that is used in this study, along with the different variations of state spaces and reward functions used to train the agent. Section 4 discusses the experiments conducted using each of these variations as well as the results of these experiments. Section 6 discusses unique challenges present in this environment and gives general conclusions of this study.

2 Background

Snake is a popular video game on a 20 by 20 grid in which a player assumes the role of a digital snake that grows as it travels and eats apples. The snake may move in four different directions and grows by a single unit in size when it travels to an apple and consumes it. The snake's movement is restricted in the following manners: it is not able to directly reverse direction (a snake moving to the left must first turn either up or down before turning right), the snake cannot travel through itself, and the the snake may move beyond the bounds of the 20 x 20 grid. The game begins with the snake in the middle of the grid with unit length 1 and an apple being placed randomly on the grid, with the location of the apple being randomly changed each time the snake consumes it. The game ends when the snake either travels into the outer bounds of the environment or runs into itself, with the overall game score being the length of the snake minus 1. Gameplay of Snake can be found here.

2.1 Related Work

DQN's, first developed by DeepMind in 2015 [1], combines Q-learning with deep neural networks. This algorithm is incredibly scalable due to the application of experience replay, where stochastic gradient descent is used to minimize the expectation of the loss function instead of directly calculating it each time horizon. Additionally, an initial experiment into training a DQN to play Snake is detailed at the following link. The experiments in this paper train on the same environment defined in this previous project and will use the same agent hyperparameters as a baseline for training and testing.

2.2 MDP Formulation

In order to measure the effects of performance upon different state space formulations and reward functions, our agent initially trains on a baseline Snake environment. This environment leans closer to a POMDP as the agent is not provided information regarding where portions of it's body is.

2.2.1 Action Space

Actions are selected from the set $\{0, 1, 2, 3\}$ with each action representing the agent moving up, right, left, and down respectively.

2.2.2 State Space

The state space of our environment contains all possible directions of the apple and immediate obstacles relative to the snake, along with the current direction of the snake. One hot encoding is used to represent each of the variables that are within a given state. These variables include whether the apple is currently above, below, left of, or right of the snake. Additionally, we represent the location of immediate obstacles (walls or snake's own body) in a similar manner. Finally, the current direction (left, right, up, down) of the snake each are represented as binary variables as well. Note that while both the snake's body and grid walls are considered obstacles, the state space does not differentiated between the two.

2.2.3 Reward Function

The agent experiences the following state-condition rewards: 10 for eating the apple, 1 for moving closer to the apple, -1 for moving farther from the apple, and -100 for either running into the game boundaries or its own body.

3 Approach

The primary goal of this study is to experiment with different state spaces and reward functions as attaining an optimal state representation and rward function is less emphasized in literature than algorithmic and robust approaches towards addressing reinforcement learning tasks. In order to isolate the effects of these various formulations, agent hyperparameters and certain environment parameters are maintaned statically.

3.1 Algorithm

In this study, I use a feed-forward DQN with two hidden layers implemented using TensorFlow/Keras. The hyperparameters of this model are static and previously defined in a previous DQN experiment on Snake [3]. Each of the non-output layers are dense layers with 128 outputs and ReLU activation function and the output layer is a dense layer with softmax activation function. The input the DQN takes in is the vectorized state and weight updates are done through experience replay and learning rate $\alpha = 0.00025$ and batch size 500 where each training point is a single instance of a state-action transition with reward.

Several environment hyperparameters are also kept static in this study, including the discount factor and epsilon parameters. In order to promote eating the apple over simply moving in circles around it, the discount rate γ is set to 0.95. In addition, the agent is trained using epsilon-greedy Q-learning, with initial epsilon equal to 1, decaying by a factor of 0.995 each instance of experience replay down to a minimum epsilon 0.01. The use of epsilon decay allows the agent to initially explore randomly and later exploit the learned policy after a degree of game success has been achieved.

3.2 Varying State Representations

In addition to the baseline state space, I consider three additional state space representations: no body knowledge, no direction, and coordinate. While the initial state space indicates whether an obstacle (body or wall) is directly adjacent to the head of our snake, this representation does not include the snake's body in the list of obstacle locations. Although the values of the state space are still the same, the snake's body is no longer represented in the environment. No direction is a state space representation in which the agent is no longer provided with the direction that the snake is currently moving–values of each of the snake direction variables will instead be set to 0 for each state the agent is in. Finally, coordinate representation replaces the snake's directional information with respect to the apple and instead provides the agent with the coordinates of the snake's head and apple scaled down to the interval $[0, 1]$.

3.3 Varying Reward Functions

After observing training and testing results the four different state representations on the baseline reward function, I decided to train on two variations of the value function as well. The first, prioritized living, maintains the same reward values for all states except the snake crashing into itself, in which it received -175 reward instead of -100. This modification was intended to improve the agent's capability of surviving longer by not running into itself after it's body had grown sufficiently long. The second, stronger death punishment, modified the death reward to -1000 for the snake hitting a wall and -2000 for running into itself. These changes were intended to prioritize staying alive, as well as diluting the apple eating to death reward ratios in order to encourage the snake to eat more apples.

4 Experiments

Using the approach mentioned in the previous section, three iterations of training and testing took place. Holding the reward function constant, the agent was trained over 150 episodes on each of the state space representations and then tested over 10 episodes on the same environment. Training was limited to 150 episodes due to the long time horizon of each game and extremely long training time due to issues enumerated in Section 5. This process was repeated for each of the three reward functions and results were averaged across number of observed iterations. The state space representations were then evaluated on a per reward function basis, considering the number of episodes required to converge to a 25 point average over 10 training episodes, along with their average score over the 10 testing episodes. Additionally, reward functions were evaluated on the average testing game score and highest training game score observed.

4.1 Test Results

During both training and testing, it was apparent that almost all combinations of state space representations and reward functions resulted in agents that were able to learn to play Snake to a certain level. Agents trained on all environments, with the exception of the no direction state space environments, were able to learn to find efficient paths towards the apple and avoid crashing into grid walls. However, average testing scores were limited to about 25 due to issues with the snake consistently running into its own body after growing sufficiently long. Due to the lack of information regarding the location of the snake's body in the grid, the DQN tended to take actions that would eventually box itself in and result in an early loss.

	base state	no body knowledge	coordinate	no direction
Baseline Reward	17.5	15.8	25.0	5.9
Prioritized Living	24.9	18.7	23.5	16.9
Stronger Death Punishment	25.1	5.3	15.8	3.2

Table 1: 10-episode average test scores of each state space-reward function combination

Overall, the experiment results overwhelmingly favor the baseline and coordinate state space representations. We observe that these two environments consistently generate both the highest average test score over 10 episodes, but also highest achieved score over the 150 episode training period. This is theoretically justified as these two state representations incorporate the largest amount of game information in the environment, allowing the agent to effectively learn to play Snake. Meanwhile, no body knowledge and no direction suffer drastically in comparison, with no body knowledge performing noticeably worse in testing compared to the former two state representations, and no direction often times failing to average double digit scores within testing. Also note that the maximum score achieved during training is a rather deceptive metric as apples are randomly generated and even by chance, apples can be generated to avoid an agent's direct path to an apple not be restricted by it's body.

	base state	no body knowledge	coordinate	no direction
Baseline Reward	50	34	34	17
Prioritized Living	44	27	44	24
Stronger Death Punishment	45	34	37	26

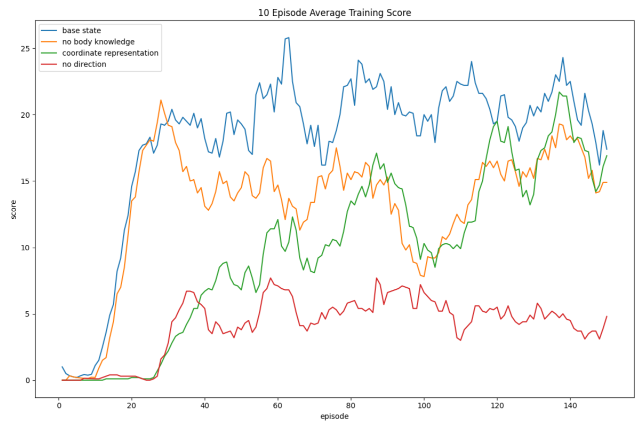
Table 2: Maximum game score obtained during training

In this experiment, we trained the agent on three different reward functions as well. While both of these reward functions result in agents that exhibit better testing reward then the baseline state, their results vary with regard to the three other state formulations. Although no single reward function is optimal for all four state space representations, although prioritized living is rather close. Through prioritized living, the agent experiences a trade off between slightly lower coordinate representation performance and much higher no direction representation performance. Similarly, the baseline and coordinate state space representations are comparable in testing results regardless of the reward function trained on. The presence of this characteristic indicates the possibility of relative optimally of reward functions and state spaces–that is, there exists several state space representations or reward functions that allow an agent to effectively learn a task.

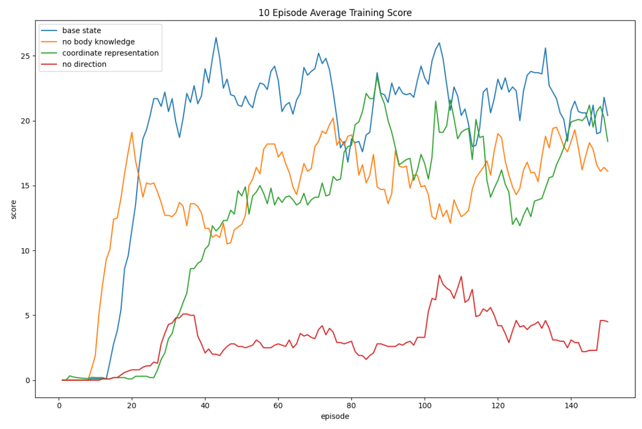
These initial results additionally indicate that there is a drastic effect on fixed-training period convergence properties of DQN's. State space representations that include more information about the environment yield better convergence properties than those those that omit critical information about the environment. In a sense, this reinforces the intuition that naive reinforcement learning algorithms are insufficient in addressing POMDP's as they bend the Markov property through lack of information.

4.2 Training Results

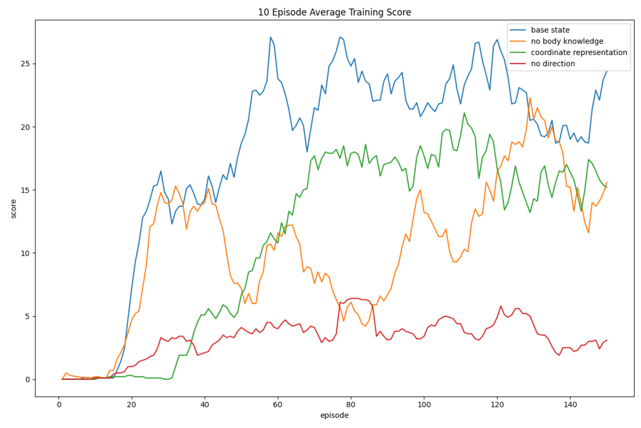
While the baseline and coordinate state space representations are comparable in their average testing score, convergence speed is an important aspect in reinforcement learning. This applies to our study as well, although we are are experimenting with the environment itself instead of different RL algorithms. While the baseline and coordinate state space representations were identified for their optimal testing results, Figure 1 clearly depicts the baseline representation's convergence time during training. It is consistently able to converge to a 25 score average within 60 episodes, while all other state space representations take about twice as long to do so, even when each representation's agent converges to a similar score ceiling (Figure 1a). Within the context of this experiment, this indicates that while there can be multiple states representations that may allow an agent to effectively learn a task, there is an optimal representation that will allow an agent to learn fastest.



(a) baseline reward function



(b) prioritized living



(c) stronger death punishment

Figure 1: 10-episode average training score for each state space-reward function combination

In conjunction with an optimal state space representation, an optimal reward function that allows many representations to converge to high reward ceilings may reduce training time. As seen in Figure 1b, the prioritized living reward function allows agents trained on each state space representation to converge much more quickly than their counterparts from the baseline reward function. In the case of the coordinate and baseline state space representations, this reward function allows agents to converge to the score ceiling between 20 and 30 episodes earlier that their counterparts using the baseline reward function. Note, I did not include plots of the the training reward with respect to each reward function as I did with the score, but the results are generally similar.

4.3 Visual Observations

In addition to the numerical results of our experiment, I made several visual observations of the game play by the DQN over each of the state space-reward formulations, which appeared relatively consistent over all reward functions. One noticeable change across the differed reward functions environments was that as the relative punishment for the snake running into itself increased, the more the learn policy became roundabout, favoring long loops around the grid of the game in order to avoid crashing into itself. State space representation observations include the following:

1. The baseline agent tended traverse the board diagonally when possible, making many direction changes.
2. The coordinate agent tended to move similarly to the baseline agent, although it would assume a policy in which it often traversed alongside its own body.
3. Although the two previous agents tended to box themselves in before running into itself, the no body knowledge agent would instead directly run into itself as it did not see the snake's body as an obstacle.
4. The no direction agent tended to crash into walls very often, and had a much longer period of random movement during training compared to the other state space representations.

5 Challenges

5.1 Addressing Randomness

Each time the snake travels to the apple's location, the apple will randomly be relocated on the grid. During training and testing there are instances in which less optimal state representations during training resulted in high test scores, which are often caused by serendipitous relocation of the apple that do not require the snake to safely traverse past it's own body. In the future, this randomness can be alleviated by running over more training and testing iterations, although this was not feasible in this experiment as a result of the extremely slow game interface during training.

5.2 Reward Function Design

Designing pseudo-optimal reward functions was a challenging task–several preliminary reward functions resulted in completely undesired agent behavior. For example, when removing the rewards associated with moving closer/farther from the apple, the agent learned policies in which it would take extremely long paths to reach the apple, often times moving in a random fashion. This problem is further exacerbated when the agent is provided with a slight positive reward for living each round–instead of attempting to find the apple, the agent would simply circle around the perimeter of the game to avoid contact with itself or the walls. The main problem regarding designing reward functions then became one of finding the best representation that would allow the agent to effectively learn to not crash into itself. Aside from defining reward functions, possible ways to address this issue would be to use recurrent LSTM's to solve the POMDP [2], or to incorporate the snake's entire body into the state space.

5.3 Implementation Level Details

The Snake environment used in this project is implemented using Python's Turtle graphics package, which is not designed for rendering complex Gym environments. Additionally, Turtle is not optimized for use in loops, with each iteration of the game running slightly slower than the previous one, making training over a large number of episodes relatively difficult. Finally, there is not option to run the environment without rendering, resulting in a relatively slow environment that takes a long time to train an agent in, even over a small number of episodes.

6 Conclusion

In this study, we explored various state space representation and reward functions and a DQN's ability to train over combinations of these variations. The results of this experimentation reinforce the idea that removing environment-relevant information from an agent (and thus creating an POMDP) is detrimental to the learning capabilities of an agent. Additionally, we identified the possibility of multiple reward optimal state space representations that allow an agent to eventually converge to an optimal policy. However, different state space representations still result in required training length even when the same convergence result is observed. Additionally, there is an and potentially several optimal reward functions that allow agents trained on various state space representations to perform well and train quickly. In comparison to common reinforcement learning literature, the results of this study underscore the importance of properly modeling the state space of an environment and designing a reward function that not only allows an agent to converge to an optimal policy, but to do so quickly. Furthermore, when an optimal state space has been identified, the representation of the state space can result in faster training and convergence to optimal reward values. This is especially applicable for work in real-world problems using reinforcement learning (ie. offline learning) in which an agent is desired to train quickly and many not have the luxury of repeated interaction with the environment.

References

[1] Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015). https://doi.org/10.1038/nature14236

[2] Hausknecht, Matthew, and Peter Stone. ”Deep recurrent q-learning for partially observable mdps.” 2015 aaai fall symposium series. 2015.

[3] Harder, Hennie de. “Snake Played by a Deep Reinforcement Learning Agent.” Medium, Towards Data Science, 9 Aug. 2020, https://towardsdatascience.com/snake-played-by-a-deep-reinforcement-learning-agent-53f2c4331d36.