Curriculum Learning with Snake

Swee Kiat Lim
Department of Computer Science
Stanford University
Stanford, CA 94305
sweekiat@stanford.edu

Abstract

Sparse rewards or a difficult environment is often a challenge for reinforcement learning algorithms that begin with randomly initialized agents. Curriculum learning, or progressively increasing the difficulty of environments, has been suggested as a way to alleviate this problem. A random agent starts off on a simple level with easy and common rewards and the environment gets increasingly harder as the agent improves. In this project, we apply curriculum learning to Snake and demonstrate how it helps to accelerate the learning of an agent, especially for large grid sizes. A code repository for this project can be found at https://github.com/greentfrapp/snake.

1 Introduction

Reinforcement learning revolves around the notion of an agent learning to take a sequence of actions that maximizes the total reward obtained in a given environment. The environment is often modeled as a Markov decision process, comprising of a state space, a transition matrix, an action space and a set of reward mappings. The state space consists of all the possible states that the environment can take on. The transition matrix refers models the distributions of the next state s', given the current state s and an action a. The action space consists of all the possible actions. The reward mappings refer to the immediate reward that the agent receives from taking action a in state s.

While previous reinforcement learning algorithms have relied on handcrafted features to model the environment, recent works have moved towards the use of deep learning methods such as Deep Q Network (DQN) (Mnih et al., 2013, 2015; Schulman et al., 2015; Lillicrap et al., 2015). These methods rely on the use of general learners, typically neural networks and variants, in order to extract high-level features from a given state. Despite challenges such as sparse and noisy rewards or labels and highly correlated samples in the training set, deep learning methods have demonstrated superior performance. In many cases, these methods have also resulted in superhuman agents that have beaten human world champions in very complex games such as Go (Silver et al., 2016, 2017), Dota 2 (Berner et al., 2019) and Starcraft 2 (Vinyals et al., 2019).

The sparsity of rewards in reinforcement learning has been a major challenge impeding effective implementation of learning algorithms. Specifically, these algorithms typically begin with a randomly initialized agent. This untrained agent explores the environment and learns from rollouts of its exploration experiences. However, if the environment is too difficult or have very few reward signals when explored randomly, the agent is unable to effectively learn from its experiences.

Consider the game of Snake, where an agent-controlled snake moves around in a grid world and gains points from eating food pellets. If the grid world is too large, an untrained agent is unlikely to randomly chance upon a single food pellet. As such, the agent might only learn to avoid death but never learns to move towards food pellets. Alternatively, the agent takes an extremely long time to learn from a slow accumulation of these rare instances where it randomly eats up a pellet.

One approach to alleviate this is the concept of curriculum learning (Elman, 1993; Bengio et al., 2009; Narvekar, 2017). As its name suggests, curriculum learning uses a "curriculum" - a modification to the environment such that initial levels are easier, with more common rewards, and gradually increasing in difficulty as the agent progresses.

This project explores the use of curriculum learning on the game of Snake. More precisely, we first examine the performance of a DQN algorithm (Mnih et al., 2013) on Snake environments with varying grid sizes, ranging from 4 by 4 to 20 by 20. Then, we demonstrate the effectiveness of curriculum learning on a 20 by 20 Snake environment and show that the agent's improvement is vastly accelerated compared to the baseline.

2 Problem

Snake is a classic video game where the player typically controls a snake in a 2D grid world. The snake is simply a line of pixels. Points are scored whenever the snake comes into contact with food pellets, which are commonly represented as single pixels. In addition, the snake also increases in length when it consumes food pellets. The game ends when the snake touches or "eats" itself.

For this project, we implement a custom version of Snake that inherits OpenAI `gym`'s `gym.Env` class (Brockman et al., 2016). This helped to integrate our work with various open-source libraries such as OpenAI `baselines` (Dhariwal et al., 2017) and `stable-baselines` (Hill et al., 2018).

In our custom implementation, the snake starts off as a line of 2 pixels and gains a pixel whenever it consumes a pellet. In addition to the rules described above, the game also ends when the snake exceeds the boundary of the grid world. The reward scheme is as follows:

- Consuming a pellet: +10
- Eating itself or exceeding boundary: -1 and game ends
- Consuming the last pellet (i.e. no free space left on grid): +100 and game ends
- Otherwise: 0

This also implies that the maximum total score attainable in a single game with a $S \times S$ grid is $10S^2 + 70$.

The state of the environment is represented as a $H \times W \times 3$ matrix, where H and W are the height and width of the grid world. The 3 channels correspond to (1) food location, (2) the head location of the snake and (3) the full body location including the head. Since the state representation only shows the current frame, the head location is required to help the agent disambiguate the head and tail ends of the snake. Other possible implementations include showing the last n frames, where n is equal to a small value such as 3. Values in the matrix are either 0 or 255, to accommodate the input range to the neural network. For example, a 2×2 grid with the snake at the top left and a food pellet at the bottom right will be represented as follows:

$$\begin{bmatrix}[[0 & 255 & 255] & [0 & 0 & 0]] \\ [[0 & 0 & 255] & [255 & 0 & 0]]\end{bmatrix}$$

Finally, to facilitate our experiments, our custom implementation also allows for defining of the grid world dimensions (i.e. height and width).

A code repository for this project, including scripts for the environment and experiments, can be found at `https://github.com/greentfrapp/snake`.

3 Experiments

3.1 Implementation

We use the DQN implementation in the `stable-baselines` library (Hill et al., 2018) along with a custom network architecture. We implement a convolutional neural network with the following layers:

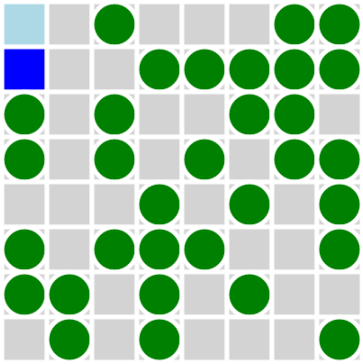


Figure 1: A random start state for $S = 8$ and $C = 0.5$. The green circles represent food pellets and the blue square represent the snake with the darker blue representing the head. With $C = 0.5$, 50% of the board is covered with food at the beginning, instead of just 1 pellet. In addition, since $C = 0.5$, if the agent turns left or up (i.e. touching itself or the boundary), it will have a 50% chance of dying instead.

- 32 channels of 3x3 kernel with stride 1 and padding 1
- 64 channels of 3x3 kernel with stride 1 and padding 1
- 128 channels of 3x3 kernel with stride 1 and padding 0
- Linear layer of 256 neurons

In the DQN algorithm, this network parameterizes the action-value function or Q function. As such, the network takes the state as input, which is a $H \times W \times 3$ matrix, where H and W are the height and width of the grid environment. The network outputs a vector with 4 elements, each corresponding to the Q-value of an action taken from the input state (UP, DOWN, LEFT, RIGHT).

3.2 Baseline

We first evaluate the baseline performance of the DQN algorithm without curriculum learning. We do this by training an agent with DQN across a variety of grid sizes in the Snake environment. In all environments the grid is a $S \times S$ square, where S ranges from 3 to 10. During the training, we periodically test the trained agent every 10k steps, with a deterministic policy via choosing the action with the highest Q-value. The agent is tested on 10 randomly initialized $S \times S$ Snake environments for a maximum of 500 time steps. We then average the total scores obtained in the environments. We report the change in averaged scores across the training iterations. We stop the training when the agent obtains the maximum score in at least one of the random test environments or when the highest obtained average score does not change for 100 tests or more (equivalent to 1 million iterations).

3.3 Curriculum Learning

Next, we train agents with curriculum learning and show that the agent is able to improve more quickly. Our curriculum scheme is designed to alleviate two challenges:

1. Sparse rewards because of difficulty in encountering food in large grid world
2. Sparse experience in end-game states

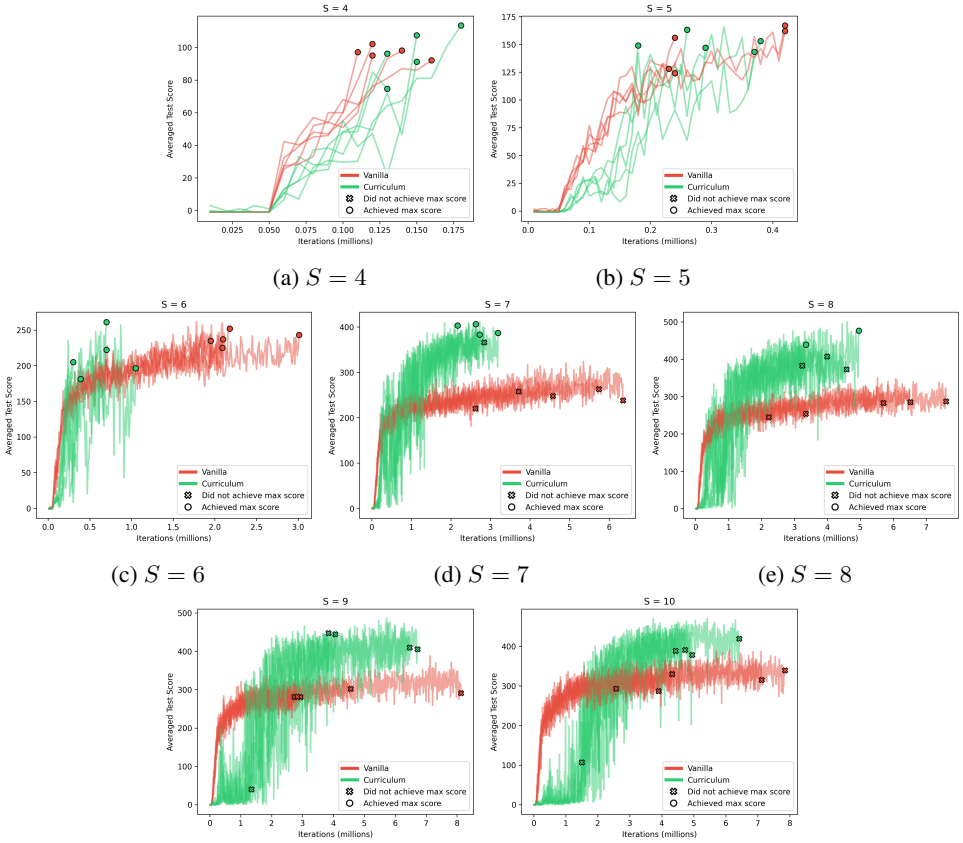
Our implementation of curriculum learning modifies the environment slightly tuned by a curriculum parameter C, which has a range of [0, 1].

Instead of randomly placing a single food pellet, we start by placing food pellets on randomly selected $C \times 100\%$ of the grid world (see Figure 1. We do not immediately replenish pellets as they are eaten. Instead, we only replenish the last pellet to ensure that at least one pellet remains in the grid world. This increases the probability of encountering food pellets compared to the vanilla environment.

In addition, instead of immediately ending the game when the snake hits the side of the grid world or eats itself, there is a $C \times 100\%$ chance that the agent continues the game from the previous state, with a reward of -1. This increases presence of end-game states in the stored rollouts.

We begin with $C = 0.5$ at the start of the training and update $C \leftarrow 0.975C$ every time the agent gets a new maximum average score in the periodic tests. This means that C gradually decreases as the agent improves and as C approaches 0, the curriculum environment approaches the real test environment. Note that in our periodic tests, the agent is subjected to the actual test environment.

3.4 Results



(a) $S = 4$ (b) $S = 5$

(c) $S = 6$ (d) $S = 7$ (e) $S = 8$

(f) $S = 9$ (g) $S = 10$

Figure 2: Average test scores over time as the agent is trained for both vanilla (red) and curriculum (green) environments, across different grid sizes S. Circles indicate that the agent achieved the maximum score available for that grid size, while crosses indicate that the agent plateaued before achieving the maximum score. Notice that for larger grid sizes (above $S = 5$), the agent learns significantly faster in the curriculum environment.

Figure 2 shows the average test scores over time as the agent is trained for both vanilla and curriculum environments. Note that both vanilla and curriculum agents are subjected to the same vanilla environment at test time. We see that the agent trained under the curriculum scheme improves faster than in the vanilla scheme. In addition, under the curriculum, the agent is more likely to eventually achieve the maximum score, such as in Figures 2d and 2e, where the curriculum agent achieved the maximum score in several runs, while the vanilla agent failed to achieve the maximum score in any run.

However, Figure 2 also shows that curriculum training may result in a train-test mismatch that causes worse performance, especially in the early stages of training. This is especially clear in Figures 2f and 2g, where the curriculum agent is initially much worse than the vanilla agent in the first 1 to 2

Figure 3: Comparison of maximum average test scores achieved by vanilla versus curriculum training.

million iterations. This mismatch is likely worsened by large grid sizes, since the chance of randomly encountering a pellet is very different between vanilla and curriculum environments, particularly at the early stages. However, we do see that if the agent manages to get past this initial hurdle, curriculum training does help to improve performance significantly.

Finally, Figure 3 explicitly shows that curriculum training outperforms vanilla training, especially on larger grid sizes. This advantage diminishes as grid sizes increase, which may be attributed to the train-test mismatch mentioned previously.

4 Conclusion

In this project, we explore the application of curriculum learning to the Snake environment. We demonstrate that curriculum learning can help to accelerate the training of agents, especially in large grid worlds. However, we also note that the creation of a curriculum is non-trivial. While undocumented in this report, we also experimented with alternative curriculums, such as imposing an artificial boundary on the grid that gradually expands as the agent improves. In contrast to the curriculum documented in this report, these alternative schemes did not demonstrate any significant improvement on agent training. In light of this, there are alternative approaches such as teacher-guided curriculum (Graves et al., 2017), curriculum through self-play (Sukhbaatar et al., 2017) and automatic goal generation (Florensa et al., 2018; Racaniere et al., 2019), which provide an automated way of designing a suitable curriculum.

References

Bengio, Y., J. Louradour, R. Collobert, and J. Weston
2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, Pp. 41–48.

Berner, C., G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al.
2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Brockman, G., V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba
2016. Openai gym.

Dhariwal, P., C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov
2017. Openai baselines. `https://github.com/openai/baselines`.

Elman, J. L.
1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Florensa, C., D. Held, X. Geng, and P. Abbeel
2018. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, Pp. 1515–1528.

Graves, A., M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu
2017. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003*.

Hill, A., A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu
2018. Stable baselines. `https://github.com/hill-a/stable-baselines`.

Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra
2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller
2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al.
2015. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Narvekar, S.
2017. Curriculum learning in reinforcement learning. In *IJCAI*, Pp. 5195–5196.

Racaniere, S., A. K. Lampinen, A. Santoro, D. P. Reichert, V. Firoiu, and T. P. Lillicrap
2019. Automated curricula through setter-solver interactions. *arXiv preprint arXiv:1909.12892*.

Schulman, J., S. Levine, P. Abbeel, M. Jordan, and P. Moritz
2015. Trust region policy optimization. In *International conference on machine learning*, Pp. 1889–1897.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al.
2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al.
2017. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Sukhbaatar, S., Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus
2017. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*.

Vinyals, O., I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al.
2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354.