Implementing and solving a multiplayer snake game environment using self-play

Lawrence Francis

InstaDeep™

MOTIVATION

The primary motivation behind this project is to assess the behaviour that agents acquire in a multiplayer snake game environment when trained using a reinforcement learning algorithm with self-play. Also, this project tries to tackle the challenge termed Slitherin'[5] in the openai's "request for research 2.0".

RELATED WORKS

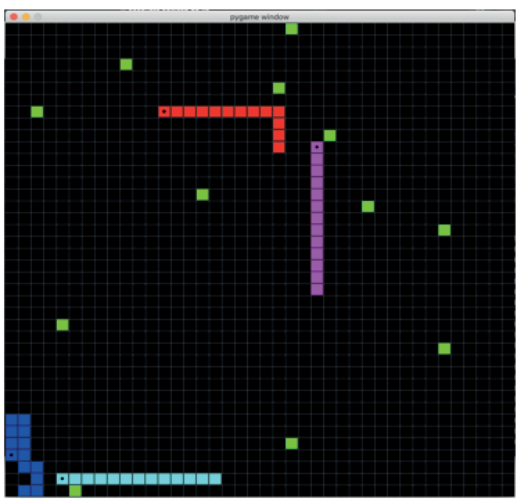
Several works have been done to address the multiplayer snake game and of course, self-play. In a project, called "Multiplayer Snake AI", The performance and inherent behaviour of using adversarial search and reinforcement learning algorithms (Q-learning) to solve the multiplayer snake game environment was analysed[1]. Also, the work on competitive self-play by openai yielded tremendous results as agents learned complex behaviours in a simple environment with sparse rewards[2]. However, there has not been any use of self-play for the multiplayer snake game.

THE ENVIRONMENT

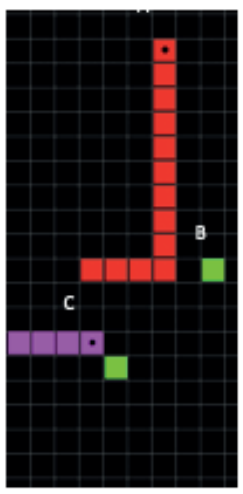
The environment, built with pygame as a Gym environment, consists of a 40x40 2D grid with fruits and snakes. Snakes increase in length when they eat(collide against) fruits and a new fruit appears at a random grid position whenever any fruit is eaten. Snakes die when they collide with the walls, themselves or other snakes. When a snake dies, it turns into fruits. The game ends when all snakes die.

An agent's observation is designed in such a way that it always seems to be moving northward. This gives it a choice of just 3 actions; turn left, turn right, and keep straight. An observation, for an agent, is a tuple consisting of the obstacles in front of and beside its head position, closest opponent's position relative to the head and closest food position relative to the head.

An agent gets a reward of +30 when it eats a fruit and -100 if it dies. The environment is solved with an overall score of 500 for 2 agents and 1000 for 4 agents.



(a) The game environment



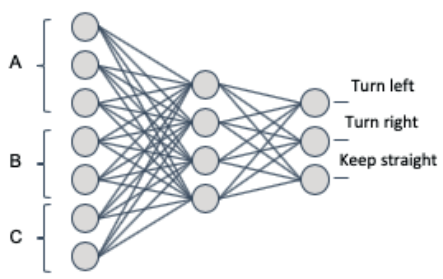
(b) The red snake's observation

Red snake's observation:
(A,B,C)
A = (0,0,0)
B = (2, -9)
C = (-3, -12)

Figure 1: The Slitherin' game environment. (a) The environment showing 4 snakes, and fruits. (b) The red snake's observation; A = obstacles at agent's head, B = closest food position relative to agent's head, C = closest opponent's head position relative to agent's head.

THE AGENT

In this multi-agent setting, the environment is solved with self-play. The reinforcement learning algorithm used is PPO(Proximal Policy Optimization)[6]. The policy network is an MLP that takes in an agent's observation as input and probability over 3 actions as output. The agent's observation forms a 7-dimensional vector when concatenated. 'Rewards to go' was used instead of a value network (critic).



PPO key equations from *spinning up*[4]

$$\theta_{k+1} = \arg\max_{\theta} \mathop{\mathrm{E}}_{s,a\sim\pi_{\theta_k}} [L(s,a,\theta_k,\theta)]$$

$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a),\ g(\epsilon, A^{\pi_{\theta_k}}(s,a))\right)$$

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0 \end{cases}$$

Figure 2: The policy network (actor), $\pi^{\theta}(a|s)$.
An MLP with 1 hidden layer with 4 neurons.
Input is agent's observation, A, B,
and C concatenated.

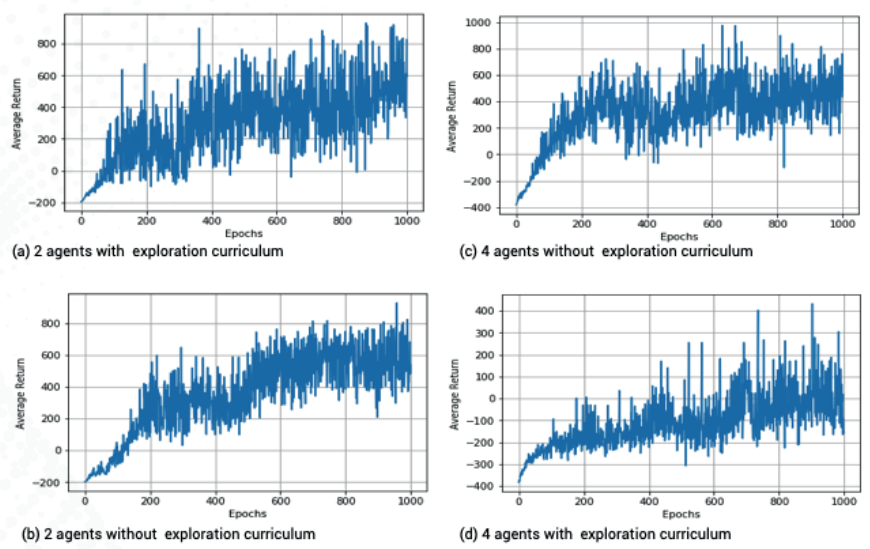


Figure 3: plot showing the average return at each epoch for all training experiments.

Learned Behaviour:

When 2 agents were trained with exploration curriculum, they both learnt to always go for the nearest fruit and avoid the walls and each other when in range. For 2 agents, when trained without exploration curriculum, One agent learnt to always chase and eat the fruits while the other learnt to run towards the fruits without actually eating any (probably for the fear of the other agent's presence) with the goal of taking out the other agent when it comes around.

For 4 agents, 2 of the agents learnt to curl around a fruit, preventing other agents from eating that particular fruit and causing them to die by collision when they attempt to eat the fruit. The other 2 agents learnt to always chase the fruit.

Conclusion

In a simple multi-agent environment, agents can learn interesting behaviours when trained with self-play. In this multiplayer snake game, one unusual behaviour observed is that of curling around fruits in wait for other snakes. In future work, the entire grid could be an agent's observation, making the policy network to take in an image as input.

REFERENCES

1 . Felix Crevier, Sebastien Dubois and Sebastien Levy, "Multiplayer snake AI" cs221 project final report, 2016.
2 . Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever and Igor Mordatch, "Emergent complexity via multi-agent competition" arXiv preprint arXiv:1710.03748
3 . Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel and Igor Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments"
4 . Joshua Achiam, "Spinning up", https://spinningup.openai.com
5 . "Request for research 2.0", https://openai.com/blog/requests-for-research-2/
6 . John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.