Writing 4

Jacob Sampson
samps284@umn.edu

April 12, 2021

Problem Description

My proposed project is an AI for a single agent competing in an environment emulating the online game Agar.io. Agar.io was released in mid-2015 and found widespread popularity on web browsers and mobile devices. The game is a competitive environment where circular agents ingest food particles to grow in size. Larger agents move more slowly but can ingest other agents who are smaller in radius. Agents are controlled by a mouse pointed in the direction of intended travel.

The full game includes bombs, which destroy a player if they are larger, splitting of agents into two, and shooting mass to decrease size. We ignore these additional features with the hope of developing an agent that can effectively use food particles and compete against agents to increase in mass. To reach and maintain a maximum size, an AI must balance searching for food with ingesting smaller agents, all while avoiding larger agents

My solution will consider a few approaches to developing AI in dynamic environments with an eventual solution built using Python and trained against other instances of the AI in a controlled server instance.

Approach

I will primarily use the method for developing a neural network-controlled AI in dynamic environments called NeuroEvolution of Augmenting Topologies (NEAT), outlined in [1].

The study in [1]–and an additional study [2] comparing algorithm performance specifically with AI agents in Agar.io–split the playing field/screen into a grid, using different methods to mark each cell with the corresponding value (e.g. obstacle, enemy, empty space).

Another team took an 'object-oriented' approach to parse video game information in [3]. Rather than using raw pixel data or a feature grid, they organized each object-type (such as the agent, competing agents, and obstacles) into variable-length feature vectors. They transformed these feature vectors into zero-padded input vectors for a convolutional neural network. I will use a similar approach, parsing the field into enemies, food particles, and the player. I will then transform these feature vectors into a fixed-length input vector for the NEAT neural network.

The output will be two different continuous nodes with values ranging from -1 to 1, representing 'x' and 'y' coordinates. These coordinates will represent the position of the mouse relative to the center of the screen, where the player is located.

Software Base

For the NEAT algorithm, the linked code repository and Python package *neat-python* in [1] will form the body of the approach. The *neat-python* package contains modules for a forward-feeding neural network with associated configuration, including the size of populations, number of generations, activation functions, mutation rates, and other parameters necessary for designing a genetic algorithm backed by a neural network. I will need to use the functions in the library to create neural networks for each instance of a species, test the agent in a training environment, extract the score, and pass the utility back into the network.

I will develop tools to parse the playing field, including identifying food particles, enemies, and the player's score/size. The *OpenCV* project includes utilities for adding filters to images and detecting circles. I will need to identify food based on the smaller size of the circles and assign all other circles as an enemy, assuming the circle in the center is the player.

For running the agent, I will use one of many existing clones of the Agar.io program running in local Docker containers with agents connecting using a headless browser controlled by the Selenium client. A similar setup is used for web-scraping dynamic websites for testing dynamic pages. I will run multiple instances of the server and randomly assign groups of agents to each with every generation.

I will develop a program that will take an image of the playing field on a set interval; will run image processing to identify food particles, enemies, and the player statistics; will transform the feature vectors representing each object-type into a fixed-length input vector; and will feed this into the neural network to choose the agent's movement (by controlling the agent's mouse cursor). On death or time-out, the agent will record its current weight for use in updating the NEAT-controlled neural network

Experiment

The AI will develop by running against other instances of the same and other species in a set of locally-hosted servers. The agents will be given a set amount of time to move about the play field before they are destroyed, their final measured mass being the output reward for the agent.

There is no explicit upper bound to the 'mass' an agent may attain. The maximum fitness a single agent could achieve is by ingesting all other agents and the maximum number of pellets on the field in the allotted time. I will monitor and display the average fitness of the whole population and individual species as the model trains.

As a baseline, I will develop an additional set of agents that are 'greedy' and will head toward the nearest food particle, taking the average final weight as the greedy agent's performance. If the AI does not develop competitive tendencies by ingesting other agents, they should not perform better than the greedy agents.

In addition to the effectiveness of the agent to attain a larger weight, I will measure the computation time for the greedy and NEAT-controlled agents for deciding a direction. I will also comment on the memory usage of each approach.

I am assuming the Agar.io game is simplified (excluding advanced features such as splitting of agents), the playing field is smaller than the live site (to force agent interaction), and the only competing agents are instances of the same and other species.

Timelines

I plan to develop the initial models by April 17th. From there, I hope to update the AI to the point of being able to run long-term training by the 24th. I hope to work on the paper during and after this period, planning to finish by May 1st. This allows for a few days to add any finishing details.

References

[1] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[2] Nil Stolt Ansó, Anton Orell Wiehe, Madalina M. Drugan, and Marco A. Wiering. Deep reinforcement learning for pellet eating in agar.io. In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence*, volume 2, pages 123–133, 2019.

[3] William Woof and Ke Chen. Learning to play general video-games via an object embedding network. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, page 8490438, 2018.