Neuroevolution Agent Based Model

Chris Harding
University of Michigan
CMPLXSYS 530 - Final Project
M.Eisenberg
4/28/24

1 Introduction

In this agent-based model, agents possess its own neural network, driving its decision-making processes. The aim is to explore how the simple evolutionary principle of survival of the fittest can produce emergent behaviors seen in other systems such as fluids and swarming. Using evolving neural networks (Stanley et al. 2019), agents adapt through survival of the fittest to better explore their landscape. The NEAT algorithm (Stanley et al. 2002) for evolving topologies of neural networks was used to optimize the amount of surviving agents at the end of each generation. The NEAT algorithm is often used for control problems such as making a quadruped walk or landing a drone. It is advantageous compared to reinforcement learning and gradient descent algorithms because the network it produces is much smaller than fixed-model methods and therefore the dynamics are more tractable. Once the agent's brains are developed, they have motion that qualitatively appear similar to boids (Reynolds, 1987), swarming (Bernoff and Topaz, 2011) and fluids depending on the scale and parameters of the simulation.

2 Methods and Model

2.1 Environment

The environment is a 2D grid with parameters x and y lengths as parameters. For this model, the environment does not wrap around from 0 to x_max. There are three types of grid spaces - barriers, survival zones and normal spaces. Most of the system is normal spaces and they do not have any special features. Barriers act like the edges of the system. Agents cannot see or move through them and die if they collide. Survival zones are the areas of the environment where agents need to move to in order to survive the generation. Figure 1 shows the environment used for training, however once agent brains are trained, they can adapt to changing environments.

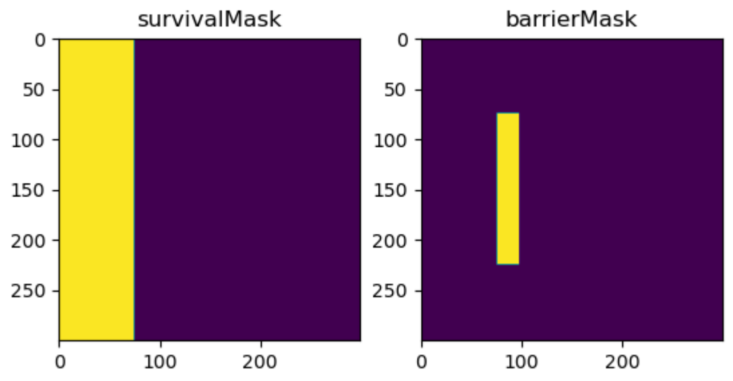

Figure 1. Survival and barrier masks used for training.

The model runs in discrete time. The environment is static over each generation. In training, the model runs for 50 time steps per generation. Once the brains are evolved, time is a parameter that can take any integer value. In general, time scales with the x,y size of the environment. Larger environments require more time to see the motion of the agents.

2.2 Agents

In this model, all agents are treated the same. Internally, they can have different neural network brains, but they still receive the same inputs and are able to produce the same actions. Each agent has a width, x, y and theta position, as well as an angular velocity omega and forward velocity **v**. Theta controls which direction the agent moves in and it moves with velocity **v** in that direction. Figure 2 is an image of agents populated in an empty environment. To reduce randomness while training brains, agents spawn randomly everywhere outside of the survival zone. Agents in figure 2 are distributed randomly from [0,300], [75,300] in x and y.

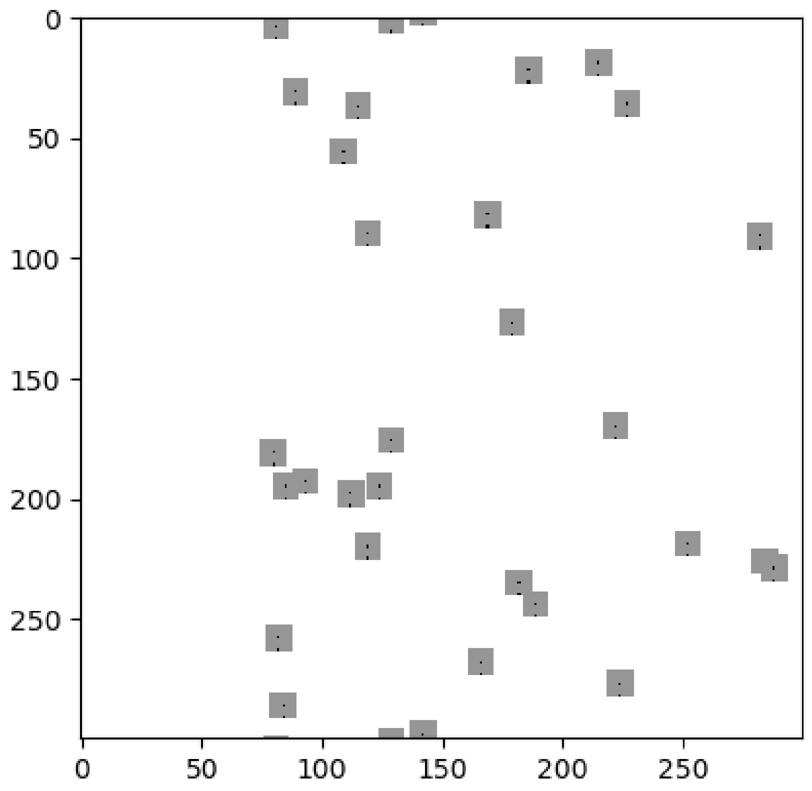

Figure 2. Agents populated on an empty environment.

Shown is 50 agents with width 5 on a 300x300 grid.

At every time step, each agent updates its sensors and produces an action based on its neural network. The inputs and outputs of the agent brain are shown in tables 1 and 2.

Sensor	Value
X location	xCoord / environment_xSize
Y location	yCoord / environment_ySize
X max dist	1 - X location
Y max dist	1 - Y location

Age	Time / steps_per_cycle

Oscillator	sin(2π * freq * Age / steps_per_cycle)
	freq = 4 in this model

Random	Random 0-1 uniform
Theta	Theta
Omega	Omega
Vision rays	See 2.2.1
Survival vision rays	See 2.2.2

Table 1. Inputs to agent neural network

Action	Agent effect
Change angle	Theta +/- π/10
Change omega	Omega +/- 1/10
Change velocity	Velocity +/- 1

Table 2. Outputs and effects from agent neural network

2.2.1 Agent Vision

Each agent uses rays as vision sensors. In the model, 7 rays were used with a max offset of π/3 from the center. Rays are uniformly spaced. In the model, the ray length was set to the environment width. Smaller values can be used, however agents who spawned further away from a survival zone than their vision allowed them to see would often have no actions. Future models could take advantage of a shorter vision distance to encourage exploration by agents. Along each ray, agents have 2 inputs. The distance until hitting something, and the distance until hitting a survival zone. If the survival zone rays hit another agent or a barrier, they end there and do not continue. Figure 3 shows an example of these rays.

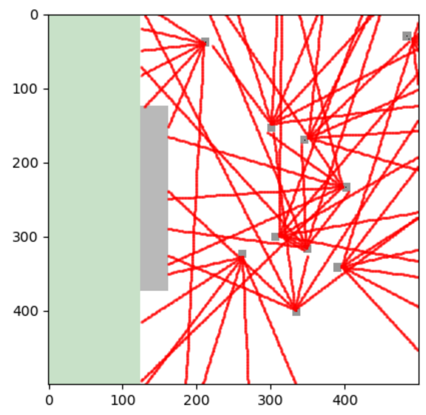

Figure 3a: Vision to survival zone of agents

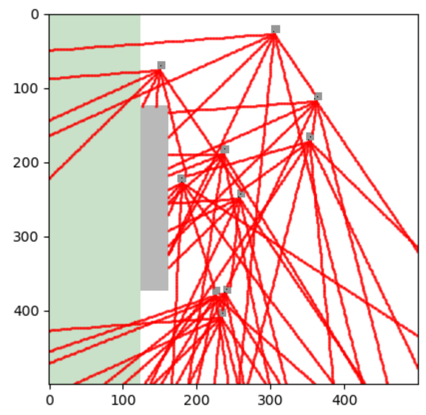

Figure 3b: Vision of agents

2.3 Evolution and Neural Networks

The model used the NEAT package developed by McIntyre et al. in python. The fitness of a gene was chosen to be the percentage of the population that survives at the end of a generation. A gene is considered fit if the average over a population size 6 has a fitness greater than 20%. Figure 4 shows two examples of genes that were evolved by the network.

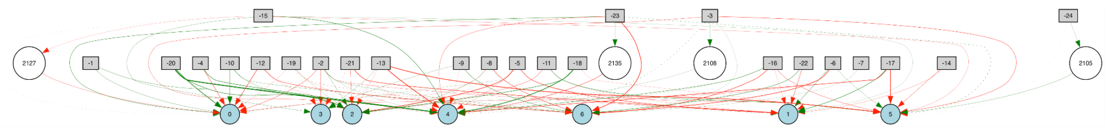

Fig 4. Example Neural Network

Green lines represent excitation, red represents inhibition.

Gray boxes are sensors and blue circles are activators.

An in-depth analysis of this network and the similarities and differences with other learned networks is out of the scope of this paper, but a possible future work. Specific parameters used for the NEAT algorithm in the model can be found in NeatConfig.ipynb and TrainBrain.ipynb.

3 Results

There are a lot of parameters that could be tested for sensitivity analysis in the network evolution section of the model. Unfortunately the model takes around 2 hours to find a fit gene, so it would be unreasonable to do sensitivity analysis on that section of the model.

3.1 Agent Count sensitivity

Below is an analysis of the effect of agent count on percentage survival of a generation. The same environment as figure 1 is used. It is 300x300 with 350 time steps per generation. For each sample, ten generations were averaged.

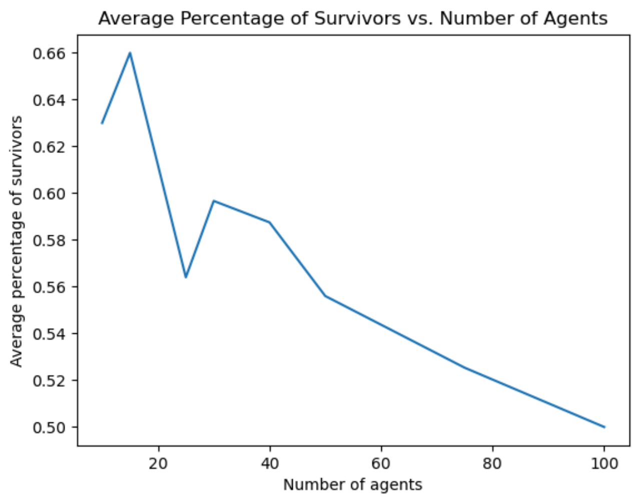

Fig 5a: Agent count sensitivity without collisions

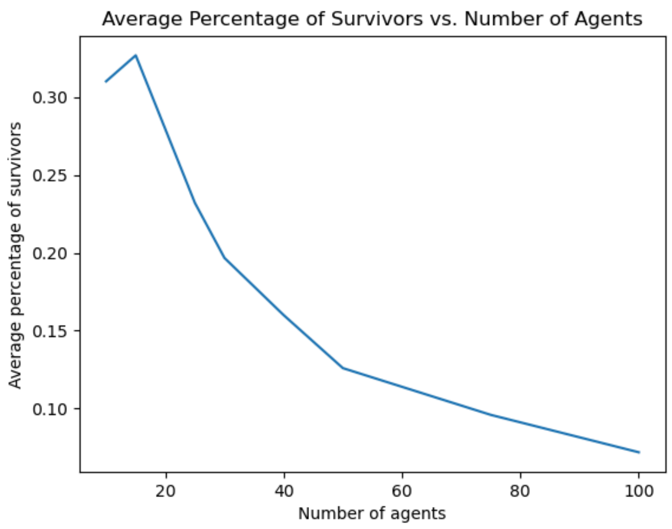

Fig 5b: Agent count sensitivity when collisions kill agents

There is some local maximum around 20 agents at which the highest percentage of agents survive. When there is a small number of agents, some are likely to spawn looking into a wall or colliding with a barrier. Since these agents will not reach the survival zone, they make up a larger percentage of the population than when you have a large number of agents. When there is a large number of agents, it is harder for each one to detect the survival zone. For an agent to know where it is, it must have an unobstructed visual path to the survival zone. As the number of agents increases, it is increasingly difficult for this path to exist. It would be interesting to run this experiment across different agent widths as well, however this would take an unreasonable amount of time. Also - as expected - when collisions kill agents, the survival rate drops faster with larger populations. The entire graph is shifted down, however it is important to note that the slope in 5b is steeper than 5a between 20 and 60 agents.

3.2 Agent Width sensitivity

Stemming from the hypothesis that the difficulty of sight rays reaching the survival zone directly influences the survivability of a generation, a sensitivity analysis on the width of the agents was conducted. Again, the study was done with collisions both killing and not killing the agents. Each generation was done on a 300x300 grid with 200 time steps and a population of 30 agents.

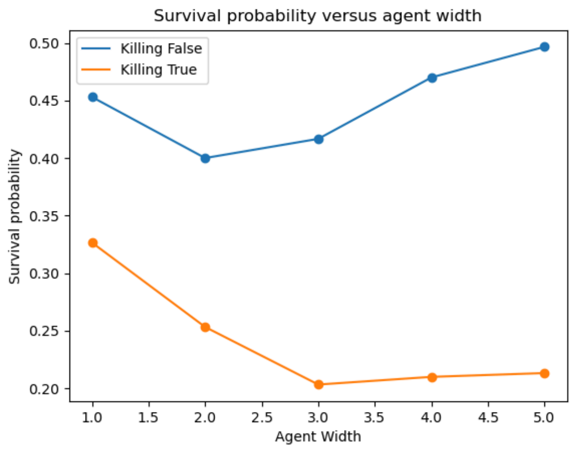

Figure 6: Survival probability versus agent width for killing on collision true and false

As shown in figure 6, agent width does not have as large of an impact as one might expect. Future works might simulate this in conjunction with 3.1 and see what the correlation between agent width, agent count and survivability is.

4 Discussion

The presented agent-based model offers an intriguing exploration of emergent behaviors through the lens of evolutionary principles and neural network-driven decision-making. By leveraging evolving neural networks and the NEAT algorithm, the model aims to understand how simple survival dynamics can lead to complex collective behaviors reminiscent of natural systems like swarms and fluids.

There are a lot of possible future works that are outside the scope of building this model. It would be interesting to compare the motion of these agents to swarming behavior, boids and fluids. A major takeaway is that training the agents on a small model allows you to expand to different scales without re-training the network for each individual environment. In earlier models of my model, I did not use the NEAT approach. Instead I trained the model to each environment and watched as the agents evolved over time. I had some success in this strategy, but the interesting aspect was the way the agents ended up understanding how to survive instead of what specific rules they needed to survive. For example, in early iterations agents would often move in the same direction every single time step. In cases where the survival and barrier masks from figure 1 were used, agents learned to move up or down, and to the left. However, upon changing the survival zone, the agents had no shot of adapting.

The NEAT algorithm allowed me to train the agents on a small, simple system, and then allow them to explore more complicated environments. Since it is difficult to represent the motion of these agents in still images, I have provided a notebook called 530_final.ipynb in which the agents all use the same brain and explore different environments. Still one of the biggest flaws is that they don't seem to explore enough if they cannot find the survival zone by chance. As mentioned earlier, another interesting future work would be training the agents on a more difficult environment with shorter vision range to see if this makes them 'smarter'. Unfortunately the NEAT algorithm is slow to converge on a solution, and takes approximately 2000 iterations that take 2-3 seconds each, and even then is not guaranteed to find a fit solution.

References

Stanley, Kenneth, Jeff Clune, Joel Lehman, and Risto Miikkulainen. "Designing neural networks through neuroevolution." Nature Machine Intelligence 1 (2019): 1. DOI: 10.1038/s42256-018-0006-z.

Stanley, Kenneth O., and Risto Miikkulainen. "Efficient evolution of neural network topologies." In Proceedings of the 2002 Congress on Evolutionary Computation, vol. 2, pp. 1757-1762. IEEE, 2002. URL: https://api.semanticscholar.org/CorpusID:9948468.

Craig W. Reynolds. 1987. Flocks, herds and schools: A distributed behavioral model. SIGGRAPH Comput. Graph. 21, 4 (July 1987), 25–34. https://doi.org/10.1145/37402.37406

J. Bernoff and C. M. Topaz,A primer of swarm equilibria, SIAM J. Appl. Dyn. Syst.,10 (2011), pp. 212--250, https://doi.org/10.1137/100804504.

McIntyre, A., Kallada, M., Miguel, C. G., Feher de Silva, C., & Netto, M. L. neat-python [Computer software]