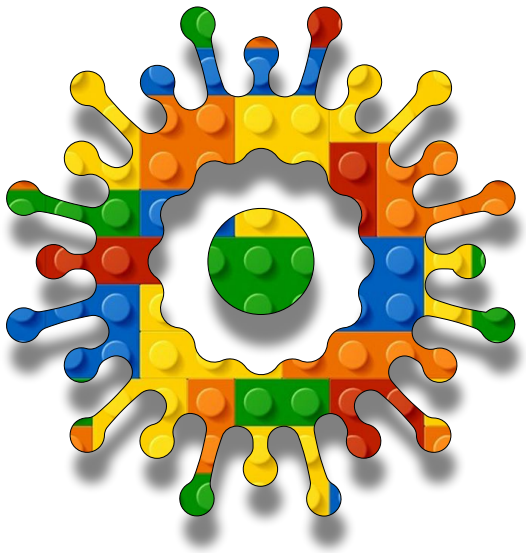
Artificial Intelligence Module C Unit #2 neuro evolution explained



Content

Module C Unit #2 introduction to neuroevolution

What is neuralEvolution?

Early on, I described three basic types of learning:

1. supervised learning,
2. unsupervised learning, and
3. reinforcement learning.

NeuroEvolution is a form of reinforcement learning. It is a method whereby a model is not trained on any data but is rewarded by its performance. What we are going to look at is a particular branch of reinforcement learning called Genetic Algorithms.

A genetic algorithm is an approach whereby we use an evolutionary approach to solving a problem. We select the best-performing pairs of models and mate them; their offspring will be better able to tackle the problem we give them.

You could see it as an evolutionary approach of the survival of the fittest. We ascribe a greater fitness level to the model that performs well, a kind of reward. The better it does, the higher the reward.

In a genetic algorithm, this process is repeated until we have a population of objects that perform a task very well, even though they have been given no data about what they are doing or why they are doing it.

You don't need to use a neural network to do this, but neuroevolution is an approach whereby we give each object a random model (or brain) and see how it performs. We do the same process of selecting the best ones and mating (crossover) them to produce a child that should have the attribute that we want.

It is important to stress at this point that we do not train it on any data; we don't even change the model; we are simply selecting pairs of well-performing models and using a function called crossover(), which is available in ml5.js, to produce the next generation of offspring. We will dive into this in a little more detail later.

Smart cars and flappy birds

We are going to look at two examples:

Smart cars

This is using the `seek()` function you coded in `code snippets 4` previously, as well as the Perlin `noise` covered in that unit. In this scenario, we have a bunch of vehicles chasing a target that is moving around randomly according to Perlin noise. Each vehicle has its own brain, and that brain is a random neural network. This is a regression task.

Flappy Bird

This is based on the very simple yet very addictive Flappy Bird game. If you are unfamiliar with the game, you have to fly through the gap between a series of pipes without hitting them, and all you can do is jump when you press the space bar (or click the mouse). Here, we give lots of flappy birds their own bird brain (random neural network) and reward those who last the longest. Eventually, you have a bird that can fly through the pipes without hitting them. This is a classification task.

What is a genetic algorithm?

A genetic algorithm is a form of reinforcement learning where an object learns through exploration and experience. A genetic algorithm is based on how nature selects the fittest to survive. Although it takes many aspects of natural selection from nature, these are the key elements for a genetic algorithm.

There are five elements to a genetic algorithm:

- Population
- Fitness
- Selection
- Crossover
- Mutation

Although you don't have to have all the elements, they should play a part in creating a thriving ecosystem. As we are creating a bunch of vehicles that will evolve over time, I will refer to a vehicle or vehicles rather than living things.

Population

We create our initial population of vehicles in the setup() function. Each vehicle will have characteristics or a DNA. They may be random to start with but will evolve over time to be better to survive as they pass their better traits on down through their offspring. Creating a population that has enough variety is also key. I know that they are vehicles, but use your imagination and imagine that they can replicate and reproduce!

Fitness

Each vehicle will have a fitness score attached. Depending on its characteristics. This will determine their chances of selection for reproduction.

Selection

This is where the ones which have the highest fitness scores pass their genes onto the next generation.

🀄 Crossover

This is where you take the best bits of one parent with the best bits of the other parent and cross them over, combining them into a new offspring. This new offspring is added to the population, thus improving on what was in the initial population.

🀄 Mutation

You may be thinking that you can just stop after crossover, but there is one last phase that makes it more likely to create a healthy and thriving ecosystem: mutation. Mutation. This means that there is a small chance of something random happening that might actually be an improvement. It avoids the possibility of the gene pool becoming stagnant and stale. It introduces more variety.

🀄 And Finally

You could reach a point where there are no more significant changes to the population, and it is said to have reached convergence, and it is thus as optimal as it is going to get. At this point, you can stop as you have reached the solution.

Smart cars

This is our first example. We will give each vehicle (car) a brain, a random neural network (ml5.js) which will have five inputs and two outputs. It will be given a fitness score depending on how close it gets to the target, which will be a wandering circle. The wandering circle will move around the canvas randomly using Perlin noise.

The inputs are:

1. The x component of the vector between the vehicle and the target
2. The y component of the vector between the vehicle and the target
3. The distance between the target and the vehicle
4. The x component of the velocity of the vehicle
5. The y component of the velocity of the vehicle

The outputs are:

1. The angle the vehicle must turn to move towards the target
2. The magnitude of the power to move it to the target

This is a regression task because we want values for the outputs, the angle, and the magnitude. Each generation will improve its movement towards the target not because it has been trained to do so but because the neural network (brain) is predisposed to do that. We only select those who do that and ignore those that don't. Eventually, after many generations, they will nearly all follow the circle even though they don't really know why.

For a diagram of the brain inside the vehicles, see figure 1

Flappy Bird works in the same way. Each bird is given a random neural network (brain) using ml5.js. The neural network will have five inputs and two outputs.

❗ When you begin the unit, you will see that we need only four inputs because the gap is constant between the top pipe and the bottom pipe, so it only needs to know one of them, but you get the point below.

The inputs are:

1. The y position of the bird
2. The bird's velocity
3. The position of the top pipe
4. The position of the bottom pipe
5. The x position of the pipes

The outputs are:

1. Jump
2. Don't jump

This is a classification task because it is either jump or not jump. The neural network looks like the diagram below (Figure 1). This works through seeing how long the bird with the best brain, for this challenge, lasts. Each bird will get a fitness score depending on how long it lasts. Each generation will get better and better until at least one bird never hits the pipes.

This is why AI is so powerful; it can do things that humans can do but never gets tired or loses interest.

Figure 1: the brain!