Genetic Algorithms

Jason Wang

April 29, 2020

"But natural selection, as we shall hereafter see, is a power incessantly ready for action, and is immeasurably superior to man's feeble efforts, as the works of nature are to those of art."

- Charles Darwin, *On the Origin of Species*

1 Introduction

Whereas neural networks have been inspired by the brain, genetic algorithms have been inspired by life and evolution itself. First published in 1975 by John Holland, genetic algorithms (GAs) are a subcategory of evolutionary algorithms, which are a category of processes designed to *optimize*. The genetic process itself is quite straightforward, but the applications are innumerable.

2 The Genetic Process

Genetic algorithms *optimize* by mimicking natural selection.[1] GAs are different from artificial neural networks in that the genetic process does not approximate a function, but is rather searching the solution space for an optimal solution. Just as nature doesn't produce perfect organisms, GAs have no guarantee to converge at the absolute maximum. In practice, however, GAs have been shown to work well for some applications.

2.1 The Plan

Let us walk through genetic algorithms for two example problems:

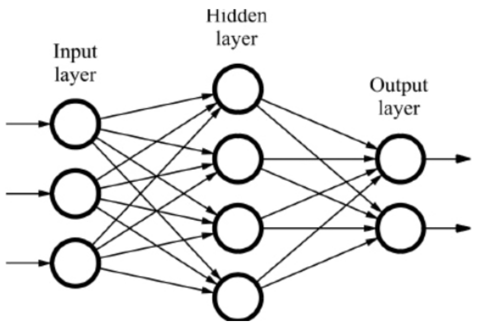
1. Tuning Hyperparameters for a Neural Network
2. Solving the Travelling Salesman Problem (nondeterministically)[2]

The TSP is a famous NP-hard optimization problem which concerns finding the path of least distance that goes through each vertex once and returns to the starting vertex. Genetic algorithms most notably are able to provide close-to-optimal solutions for this problem, and this problem is a common exercise for learning GAs.

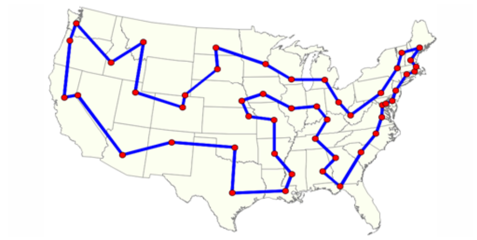
[1]Natural selection is defined by survival of the fittest where individuals with greater odds of surviving are more likely to pass along their genes and dominate the gene pool, and this process is driven in eukaryotic organisms by recombination of chromosomes in meiosis as well as mutations.

[2]Genetic algorithms rely on a lot of randomness.

Figure 1: Our Examples



(a) Neural Network

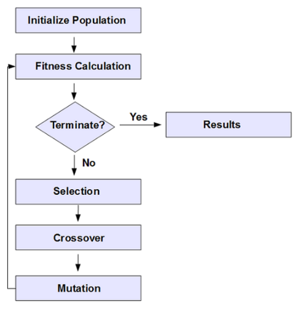


(b) Travelling Salesman Problem

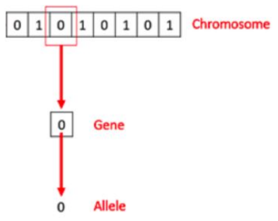
2.2 The Setup

Back to the genetic process. We are going to set up our "organism" as a single[3] chromosome represented by an array of information, where each index is considered to be a gene and the value is the allele. Values may be bools, ints, floats, etc., which *encode* the organism's proposed solution to the optimization problem. We will maintain a population of these organisms, usually randomly initialized[4], and conduct natural selection on them based on an individual's "fitness" (the optimization problem). Determining "fitness" usually consists of a fitness function or fitness simulation, and is the most time-consuming part of the genetic process.

Figure 2: Our Setup



(a) The Genetic Process



(b) Data Structure

What does that mean for our two examples?
For our neural network, let's say that we want to find the best combination of learning rate, number of layers, batch size, momentum constant, and dropout rate. I'm going to *encode* these values into an array as doubles between -1 and 1. To get the fitness of our array/chromosome, we'll *decode* these values into the appropriate range (e.g. learning rate could be between 0 and 1, 0 and 0.1, etc.) and run them on the data set and take their accuracy.

For TSP, we'll encode the path of these cities as a string of ordered vertices. The fitness function will be the reciprocal of the distance of the path.[5]

[3]It is interesting to consider multiple chromosomes.
[4]Genetic processes may also run on previously optimized solutions (i.e. a population can be saved and loaded).
[5]So that the smallest distance will have the greatest fitness.

2.3 Selection

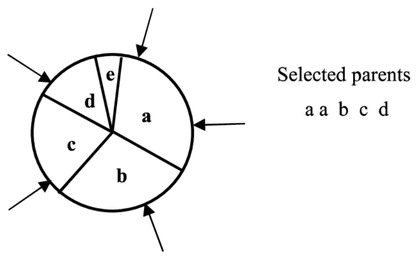
There exists numerous methods to apply natural selection. Essentially, we want to select the most fit individuals for the next generation[6] *while still making it possible for beneficial mutations to arise and maintain a genetically diverse population.*

Selection schemes:

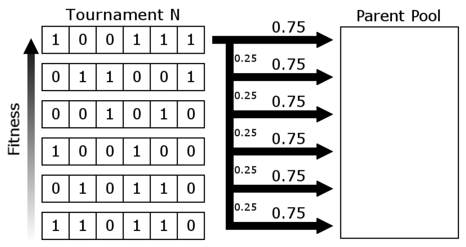
- Fitness proportionate selection (FPS)
    - Exactly what the title implies
    - $p_i = \frac{f_i}{\sum_{k=0}^{N} f_k}$
- Ranking selection (Rank proportionate selection)
    - Exactly what the title implies
    - Use FPS but set the fitness from 1 to N.
    - Helps preserve genetic diversity
- Stochastic universal sampling (SUS)
    - Think of it as spinning a spinner allotted by fitness (FPS), but instead of taking a single individual and spinning multiple times, we take individuals at specific intervals around the spinner and spin just once
    - More generous to low fitness individuals than FPS
- Tournament selection
    - For every child to be produced, choose a random sample of individuals
    - Go down the sample's individuals from highest fitness to lowest fitness
    - There is a certain probability of either selecting that individual or moving down the list
- Boltzmann selection
    - Less fit individuals are more likely to be selected in earlier generations than later
    - Based on Newton's Law of Cooling[7]

Besides from selection, two measures exist to ensure that the most fit or a few of the most fit individuals survive from generation to generation: elitism (cloning the best fit individual(s)) and steady-state selection (natural selection does not replace the entire generation)

Figure 3: Select Selection Schemes Visualized



(a) SUS



(b) Tournament Selection

[6] I describe a genetic process that kills everyone after they breed. This does not necessarily have to be the case. GAs that use either elitism or steady-state selection are common, ensuring that not everyone dies so to speak.

[7] The formula is $p_i = e^{-\frac{f_{max}-f_i}{T}}$ where T "cools" by Newton's Law of Cooling over the generation. Wow, also inspired by nature.

2.4 Recombination

Recombination, also called crossover, is the main mechanism in which new genetic material (new guesses to the solution) is created. The available recombination methods depends on whether the encoded information represents a combination (e.g. hyperparameters) or a permutation (e.g. TSP).

Recombination schemes for combinations:

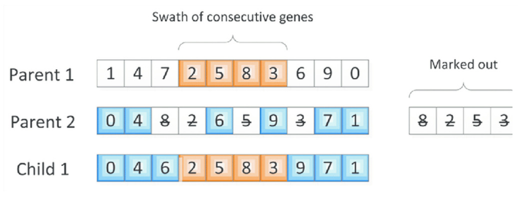
- One Point Crossover
  - One index is selected randomly, all genes before that index are inherited by parent 1, and all genes after that index are inherited by parent 2.
- Multi Point Crossover
  - One point crossover, but with multiple points.
- Uniform Crossover
  - A set probability of parent 1's genes are inherited, and the rest comes from parent 2.

Recombination schemes for permutations:

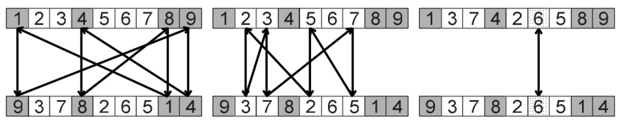
- Order Crossover (OX)
  - The child consists of n random regions[8] from parent 1, and the rest of the alleles that haven't been copied over are inserted in order from parent 2.
  - This is a fast and simple operation.
- Cycle Crossover (CX)
  - The child inherits alternating "cycles" from the parents. For example:
    Parent 1 is [3,1,2,4,5,6]
    Parent 2 is [1,2,5,6,3,4]
    Cycle 1 is $3 \rightarrow 1 \rightarrow 2 \rightarrow 5$
    Cycle 2 is $4 \rightarrow 6$
    So the child would Parent 1's side of Cycle 1 and Parent 2's side of Cycle 2: [3,1,2,6,5,4]
  - Convince yourself of the notion of a cycle.
- Partially Matched Crossover (PMX)
  - A hybrid between OX and CX
  - The child consists of a random region from parent 1,
    Genes from parent 2 that don't create conflicts are inherited.
    Genes in conflict from parent 2 are substituted by going down the cycles previously described in CX.
  - Convince yourself that going down the cycles like this will always produce valid chromosomes.
- Edge Recombination
  - Consider the permutation as an adjacency list (easiest to think of as a path for problems like the Travelling Salesman Problem, but applicable in other permutation-based problems),
    Take the union of the two parent adjacency lists,
    Create a permutation by greedily choosing the allele with the least adjacencies, and removing that allele from the other adjacency lists.
  - This a slower operation, but produces children of less disruption to the permutation.

[8] 1 or 2 regions are usually sufficient.

Figure 4: Recombination Schemes Visualized



(a) Order Crossover



(b) Cycle Crossover

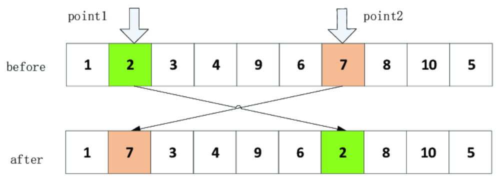
2.5 Mutation

There are also many schemes for mutating a chromosome. Keep in mind that only a percentage of chromosomes should undergo mutation.

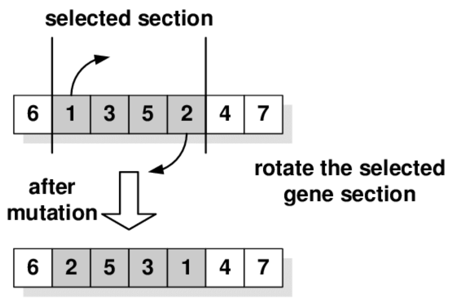
Mutation schemes:

- Randomizing a value[9] (to uniform dist., to Gaussian dist., to extrema, etc.)
- Swap two random indices
- Reverse the array between two random indices

Figure 5: Mutation Schemes Visualized



(a) Swap Mutation



(b) Inversion Mutation

At the end of each pass of the genetic process, it is advised that no pair of individuals be allowed to be genetically identical. Otherwise, genetic diversity will decrease greatly.

Genetic algorithms are generally run until a human overseer decides that the solution has converged to an acceptable optimality, a specified number of “generations”, a specified fitness, or a specified time limit.

3 Fun Stuff

Check out these games showcasing genetic algorithms!

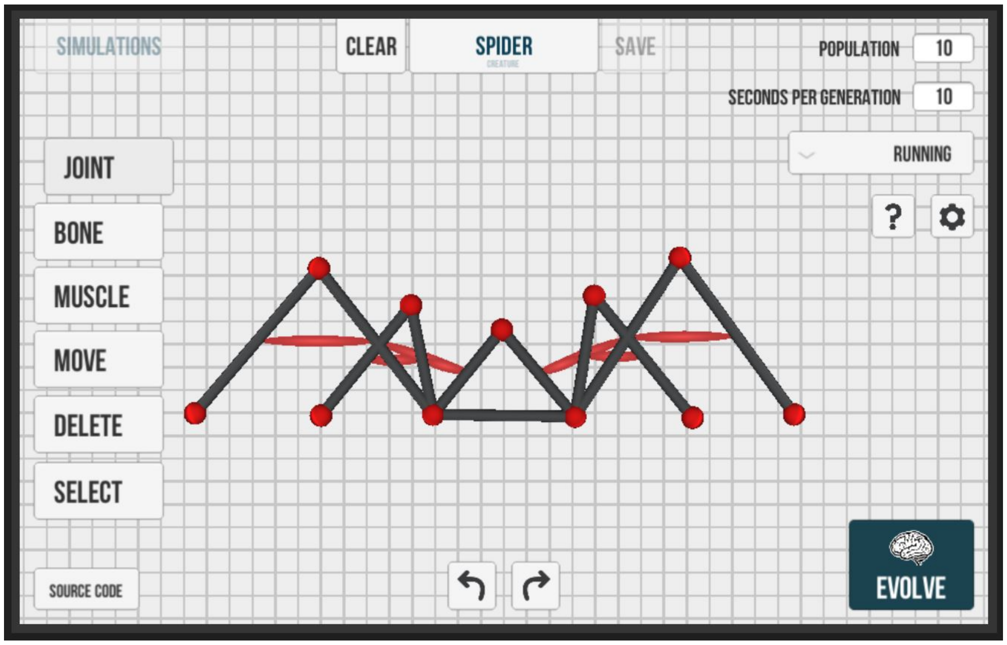
[9]Not applicable to permutation-based chromosomes.

3.1 Motor Control

`https://keiwan.itch.io/evolution`

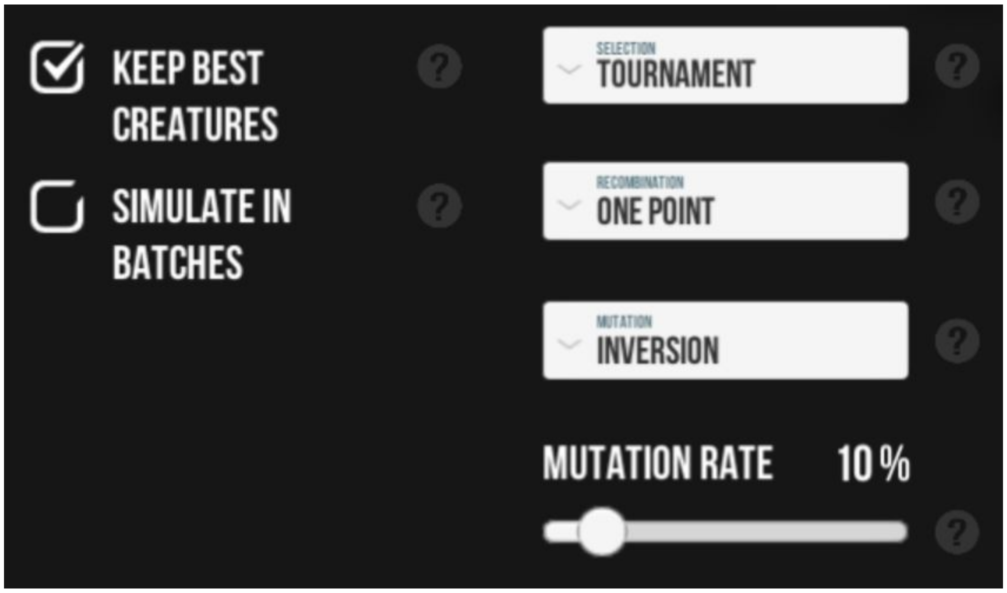
Can you figure out what the genetic code controls?[10]

Figure 6: Genetic Evolution Gameplay



POPULATION 10
SECONDS PER GENERATION 10
RUNNING

(a) Model Creation



KEEP BEST CREATURES ✓
SIMULATE IN BATCHES
SELECTION TOURNAMENT
RECOMBINATION ONE POINT
MUTATION INVERSION
MUTATION RATE 10 %

(b) GA Settings!

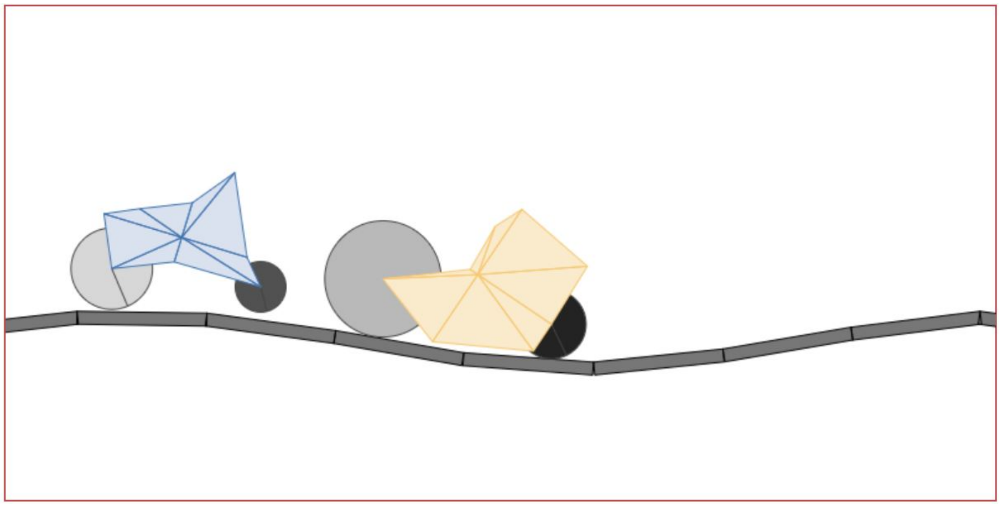
[10]Click on the ? in the editor and read "The Brain" and "The Simulation".

3.2 Car Design

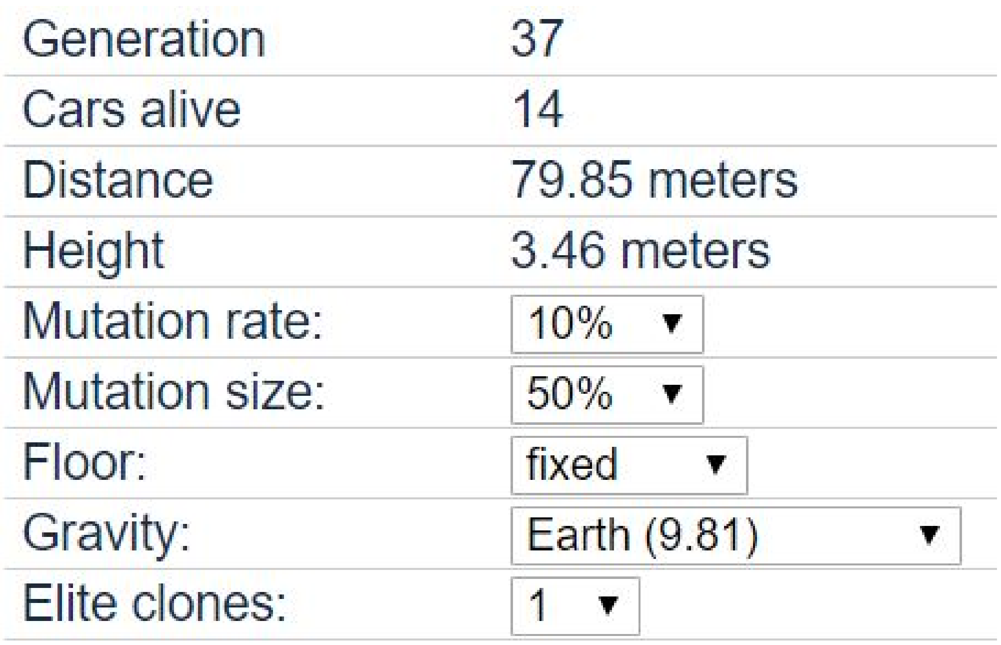
`https://rednuht.org/genetic_cars_2/`

Can you figure out what the genetic code controls?[11]

Figure 7: Genetic Cars Gameplay



(a) Racing Cars



Generation	37
Cars alive	14
Distance	79.85 meters
Height	3.46 meters
Mutation rate:	10% ▼
Mutation size:	50% ▼
Floor:	fixed ▼
Gravity:	Earth (9.81) ▼
Elite clones:	1 ▼

(b) GA Settings!

[11]Scroll down and read the “Genome” section.

3.3 Applications

Albeit being just games, these give us ideas of real applications of GAs. For example, GAs have been used to teach robots motor control and have been used by engineers and designers to create airfoils.

In fact, Wikipedia has a list dedicated to the subject.[12]

Try making your own genetic algorithm game to simulate one of these applications!

4 Food for Thought

There are so many different hyperparameters and genetic schemes to play around with.

As seen in our example, genetic algorithms can be used to *determine the optimal hyperparameters for a neural network*. This combination of methods can produce more desirable/reproducible results. As you can imagine, though, training countless neural networks takes a huge amount of time/resources.

For further reading, take a look at Adaptive Genetic Algorithms (AGAs), which don't have static parameters during the genetic process, providing finer control over the genetic process.

Genetic algorithms, as mentioned before, *don't necessarily converge to absolute maxima*. In addition, GAs can be quite *sensitive and are generally slow to converge to an acceptable optimal solution for some problems*.

Along with neural networks, genetic algorithms show extraordinary ability to be used to solve completely dissimilar problems (cough cough, our two examples). This is the power of machine learning.

5 References

https://www.analyticsvidhya.com/blog/2017/07/introduction-to-genetic-algorithm/
http://www.cs.cmu.edu/~02317/slides/lec_8.pdf
https://pathmind.com/wiki/evolutionary-genetic-algorithm
https://ai.stackexchange.com/questions/9077/selection-methods-in-genetic-algorithms
https://en.wikipedia.org/wiki/Genetic_algorithm
https://www.obitko.com/tutorials/genetic-algorithms/selection.php
http://www.rubicite.com/Tutorials/GeneticAlgorithms.aspx
https://towardsdatascience.com/introduction-to-optimization-with-genetic-algorithm-2f5001d9964b

6 Image Sources

https://apacheignite.readme.io/v2.7.6/docs/genetic-algorithms
https://www.researchgate.net/profile/Teresa_Ludermir/publication/220256902/figure/f

[12]https://en.wikipedia.org/wiki/List_of_genetic_algorithm_applications

ig11/AS:643198904967169@1530361977034/Stochastic-Universal-Sampling.png
https://www.researchgate.net/profile/Shuihua_Wang2/publication/282998951/figure/fig4/AS:433588496801795@1480386961920/An-example-of-order-crossover.png
https://camo.githubusercontent.com/9cb9e734cf88cdf9c45c5b3c57dc29f215336cb5/687474703a2f2f7777772e63732e76752e6e6c2f7e6775737a2f6563626f6f6b2f666967757265732f332d31372e6a7067
https://databricks.com/wp-content/uploads/2019/02/neural1.jpg
https://physics.aps.org/assets/a38de7c6-00ac-45fb-9bcf-3b3e14a72b41/es32_1.png
https://www.researchgate.net/profile/Yong_Zhou17/publication/331675012/figure/fig4/AS:735600009310209@1552392117053/SWAP-mutation.png
https://www.researchgate.net/profile/Ming-Shen_Jian/publication/235945370/figure/fig10/AS:668753410936844@1536454646828/An-example-of-inversion-mutation.png