Evolutionary Algorithm

By Abdullah

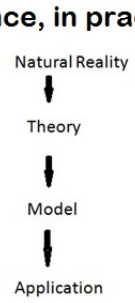
December 2020

Contents

0.1 Introduction

Charles Darwin presented a theory of evolution in 1859 which explained the phenomenon of evolution, diversity (variation) and adaption (selection of fittest).

Life is continuously evolving and changing and it was discovered in the turn of nineteenth century to the twenty century that life is tied up in some form of "genetic information" contain in organism and random events are occurring at molecular level having role in evolution and selection (adaption).



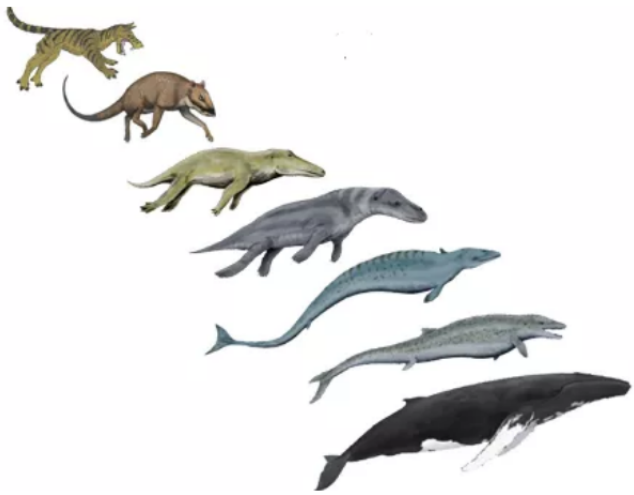
Our goal was to embed that natural phenomenon to an application by following a science model.

0.2 Analysis

0.2.1 Natural phenomenon

There are many Natural phenomenon occurring in nature like sunrise, rainbow, thunderstorm and many more.

The one introduced by Darwin was that every organism evolves with a passage of time. The process of change in organism may introduce physical or behavioral traits. These upcoming changes are due to environmental changes or natural forces that help offspring to survive. He also proposed that these changes always leads to the best product.



0.2.2 Theory

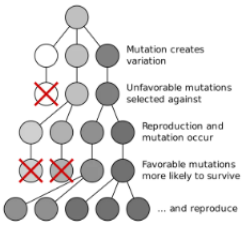
Theory has two main points "All life on Earth is connected and related to each other" and "this diversity of life is a product of modifications of populations by natural selection, where some traits were favored in and environment over others"

This theory is also known as "Survival of the fittest". Darwin proposed that organism of a same specie has a wide range of variation for a characteristic (Initial Population). Solution or individual with most suited characteristics survive and breed (Selection plus crossover and mutation). The characteristics that made them survive then passed on to the next generation (evolution-new generation).

0.2.3 Computational Model

There are five phases of model: **(Algorithm)**

- we are given with a set of solutions (population).
- we compute fitness of each solution.
- Best fitted solutions are selected (Natural Selection).
- Process of crossover and mutation occurs within (New generation formed).
- Repeat step 2, 3 and 4



(Pseudo-code)

- Initial population
- Compute fitness
- REPEAT UNTIL WE CONVERGE TO BEST FIT
- —-Selection
- —-Crossover
- —-Mutation
- —-Compute fitness
- END

0.2.4 Application

This is a search based algorithm. initially, we were given with 2 images (one small and a large), our keen work was to match the small image template in the large image and to find the best fit. This is also known as optimization problem. –I will try to explain my logic and code step by step precisely blew:

0.2.5 Initial Population

Here, in this application, population means (x,y) coordinates. These were generated randomly but with a constraint of height and width that means x,y should not go beyond the larger image's width and height.

0.2.6 Fitness calculation

Images are just a 2d array so as we are provided with 2 images (one larger and a small one) so we simply extract a 2d sub array from a large image and matches it with the provided small one. There is a formula plus a function predefined in python which takes both 2d arrays (2d sub-array and small image 2d-array) and returns a value between (-1 to 1) called as coo-relation value. This coo-relation value tells the match between 2d arrays. if value is near 1 so they maximally matched else not.

- slicing is used to extract sub-array
- I used a predefined function of coo-relation from scipy python library

0.2.7 Natural Selection

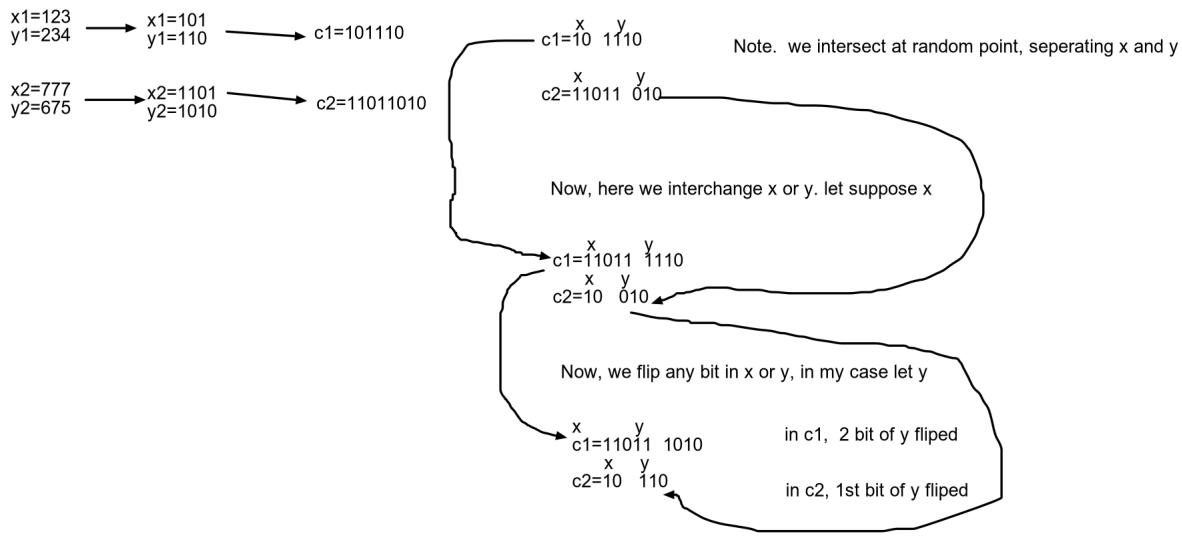
Once we are given with the population along with fitness. we will sort the population or solutions w.r.t there fitness values. Once we are done with sorting, we will select the first 2 (also the best 2) and apply cross-over and mutation on it. but in my case, i am not losing or discarding worst fit, instead i am apply cross-over and mutation on every pair starting from the top of list.

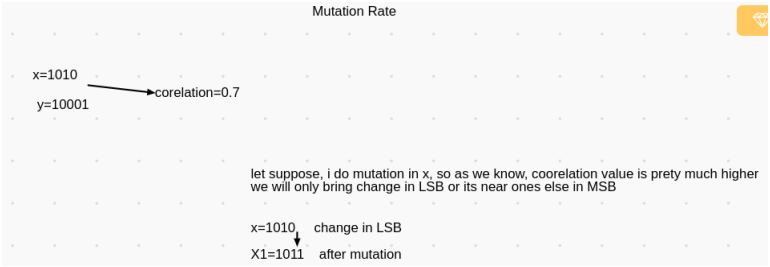
0.2.8 Crossover and Mutation

Now, here comes the tricky part.

- First of all, i pick first 2 parents and converts the x and y values of them in binary.
- Then i concatenates the binary of each parent x and y. (x before then y)
- Result is 2 binary concatenated string c1 and c2 (from parent p1 and p2).
- a random bit is selected from both and from that point, x and y are separated
- After separation, we came across to new values of x and y depending upon the point of intersection.
- Now here, crossover occurs as we interchange x or y. In my case, its randomly selected.
- Till now, we have new values of x and y (childs) generated from parents after crossover.

- Now for mutation, we have to select either x or y values of child for variation part. let suppose x, so we flip any bit in x of childs.
- In my case, x and y selection and bit selection for mutation is random.
- I also adjust the mutation rates i.e. if a parent have a respectable coorelation value then we will decrese the mutation rate else not.
- Similar method, will be applied on each pair of parent down the list.





0.2.9 Conclusion

According to my understanding and w.r.t evolutionary algorithm, my population should converge to the fittest, which means if i plot a graph, it should have a rising behaviour but i got some surprising results below:

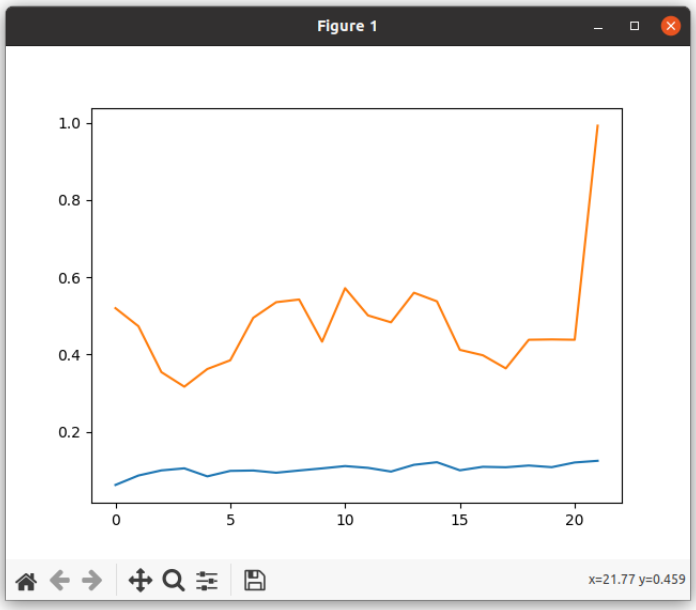
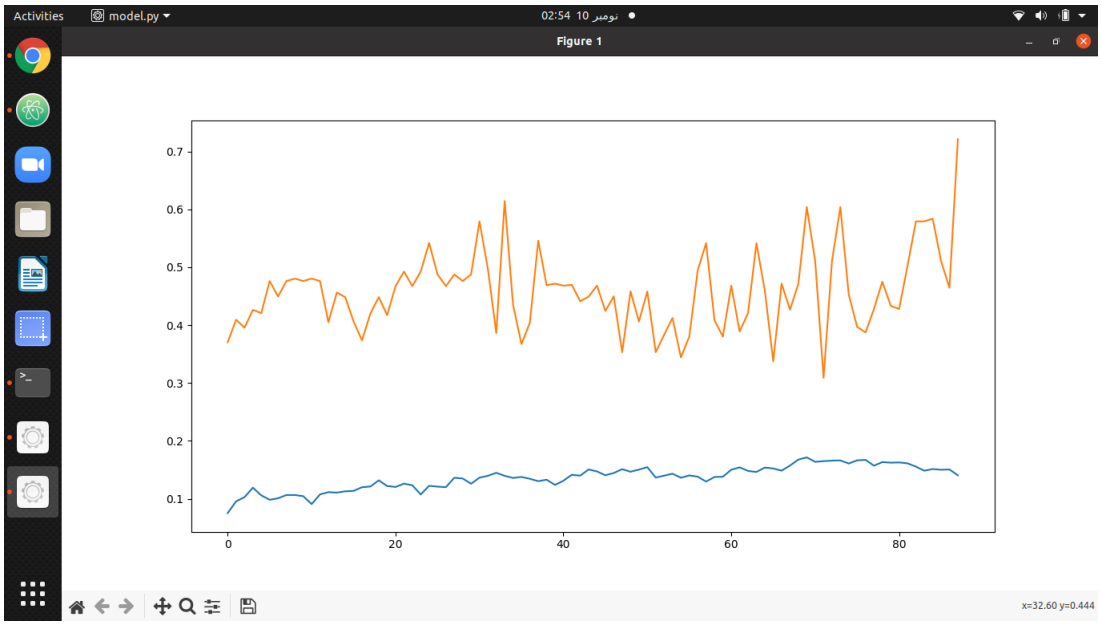


Figure 1
1.0
0.8
0.6
0.4
0.2
0
5
10
15
20
x=21.77 y=0.459


Activities
model.py
02:54 10 نومبر
Figure 1
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
x=32.60 y=0.444

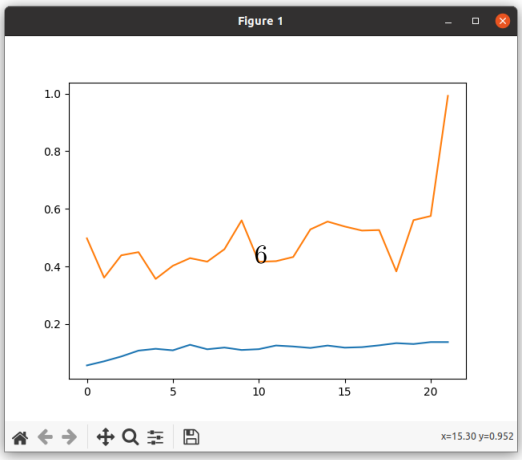


Figure 1
1.0
0.8
0.6
0.4
0.2
0
5
10
15
20
x=15.30 y=0.952

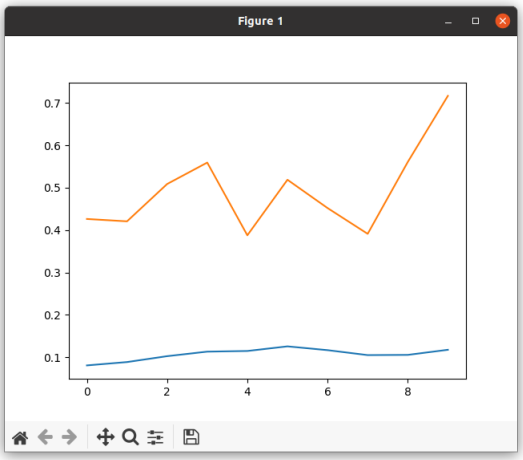


Figure 1
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
2
4
6
8

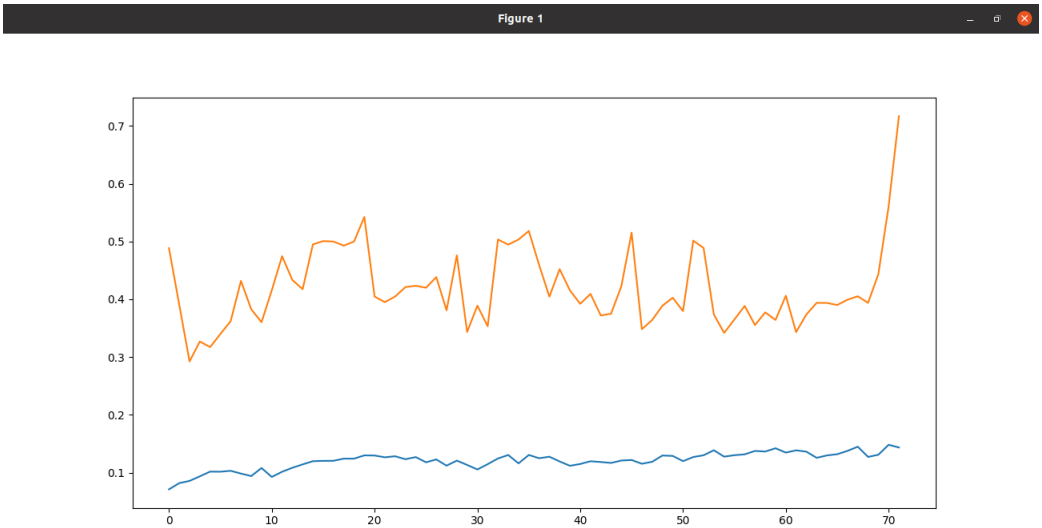
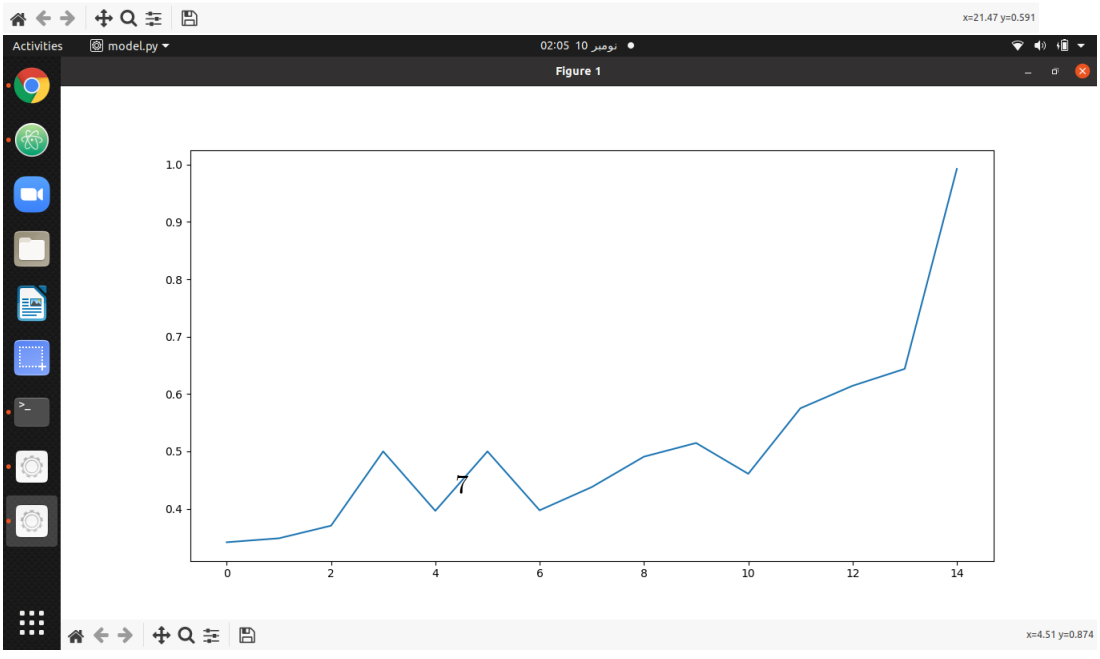
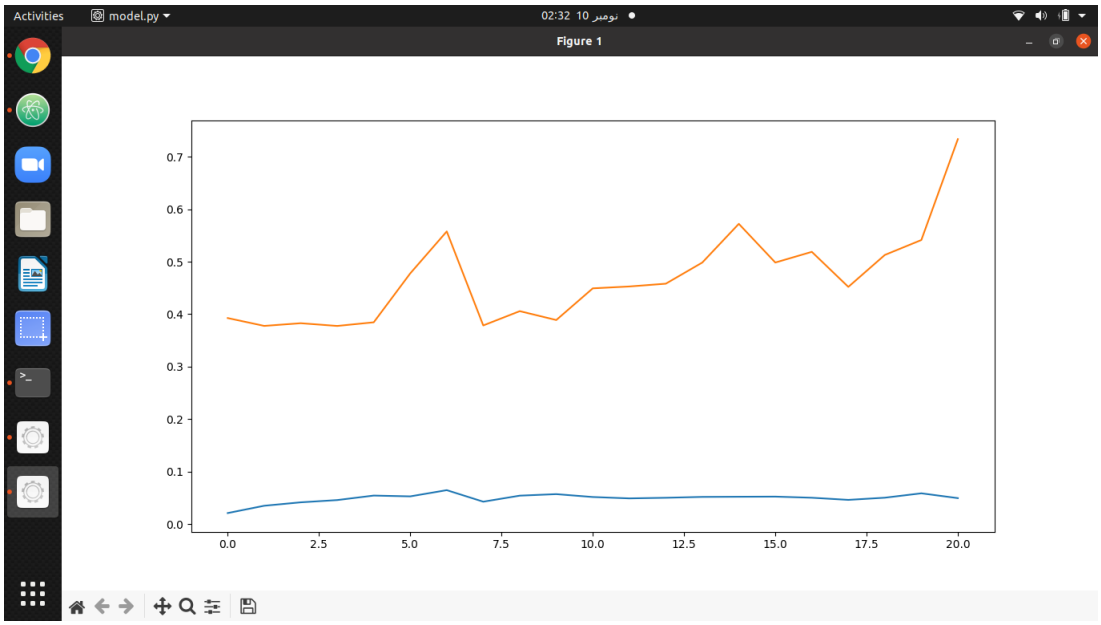


Figure 1
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50
60
70


x=21.47 y=0.591
Activities
model.py
02:05 10 نومبر
Figure 1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
2
4
6
8
10
12
14
x=4.51 y=0.874



0.2.10 Finding

From here, i concluded and its also according to my understanding that in my case, graph rising will not be that smooth as there are many face and they have so much resemblance with each other.

for example, if we come ac-cross a face other than target, so by nature of algorithm, mutation rates changes (very low value changes) so we will start emerging more and more near to it. but we know that's not the target so what will happen is that, we can converge to that face to a particular extend let take 0.6 (coo relation). so it cant go above 0.6 as its not the target so we need to re spawn at another place so starting evolving around it.

by above mentioned procedure, we will have rising patches in our graph and each rising patch denotes some face around which, we converge but we moved on to other faces and finally got the desired face.

0.3 Recommendation:

- I chosen the initial population of 250, as it was more fesible to me which having a look at graph trends, 50 makes it very squeezed
- I set the mutation rate 50percent left or right w.r.t coorelation value. Above 0 will have change in right 50 percent of binary string(least change) below 0 will have change in left 50 percent(more change)