Intro to Evolutionary Algorithms

Shahil Rauf

12/18/2017

Intro

Hello, and welcome to my write-up of a practice exercise I engaged in. This is basically here to serve as a guide through my code, and an intro to the very basics of evolutionary algorithms. There is nothing exceptional about this, as it is the most bare fundamentals but feel free to replicate, or improve upon this as you see fit. As it is the only problem that this tries to solve is to generate the largest number up to a certain point(255). Enjoy!

1 What is an evolutionary algorithm?

Evolutionary algorithms are a way to solve the best answer to a problem, by using the methods replicated in nature. As a result, much of the terminology is derived from biology and similar areas of study. The first step is to understand how evolution works. First off, you have your **initial population** of creatures with their **genomes**. These creatures compete to survive, and only the **fit** survive through **natural selection**. Those go on to reproduce and form the next generation of **offspring**. This process uses a large amount of random variation through processes to arrive at an efficient or correct answer.

1.1 Initial Population

The purpose of an initial population is to give you somewhere to start. This should be randomly generated to give a certain amount of starting variation. The example used in my code can be seen in the $gen_number()$ function. It accepts an input x which is the number to generate. It is called on line 78, and it is not a part of the loop below it as it is only there to generate the initial population. It is not to be used ever again as it is completely random. There is no guiding principle to it and therefore, it isn't in the loop. I have found that a larger initial population doesn't have much impact. The real impact comes from the difference between the initial population size, and the population size after selection occurs. There is a sweet spot between too high and too low, so feel free to mess around with those values.

Also if you proceed to look at the code for that function, you will find that there are some peculiar methods there. For example, there are two arrays consistently in use, and that's the *numbers* and *integers* arrays. The numbers array is there to contain the genome, whereas the integers array is to make it easier to debug, and compare numbers. It was not necessary and can be removed, however it makes it easier to leave it there.

Another peculiar thing about the initial startup code is that I use the *num_to_bin*() function. This is because binary is much easier to make random than integers. The conversion to strings was just so that they could be a consistent length, since python would only show the relevant bits. The importance of all binary numbers being the same length comes into play with the *mutate*() function.

After executing this function, you have an initial population. You can alter the code to display the initial population however it currently does not do so.

1.2 Genomes

At this point, you must be saying that everything makes sense, but why do we generate these numbers as they are? Well, the the binary numbers are the actual genome of the "creatures". These serve as their genes and are subject to mutations and are the building blocks of the "creature's" performance. We use the genome as it is in real life to also determine the offspring. In reality, the genome is a series of Adenine-Thymine and Guanine-Cytosine pairs. In this simulation, they are simply 1s or 0s. The importance of Genomes is in two scenarios: Offspring and Mutation

1.2.1 Offspring

In biological scenarios, there is a complicated process that decides how a baby is born, what the dominant and recessive genes are, etc. For this simulation, I made it a little simpler, and also a fair bit more random. Inspecting the code on line 40 and 41 will show you that the parents of offspring are chosen completely randomly. It is possible for the parent to "mate" with its own child and produce an identical child. This is usually not too big of a deal however. Due to the algorithm for selection, it gets dealt with fairly easily. It also becomes much less likely with a larger population post-selection.

From these two parents, it goes through each bit, and chooses one of the parent traits. If both parents had 1 for a trait, there is a 100% for the child to have that trait. Conversely, if the parents had 0 for a particular trait, the child has a 0% chance of acquiring that trait from their parents.

1.2.2 Mutation

With a lack of mutation, it would be possible for me to never see a creature that has maximum potential. For example, if it weren't for mutations, new genetic material would never see the light of day. If all "creatures" had a 0 for their leftmost bit, they would never in a million years change it to a 1 without a random mutation. So the mutation process is mandatory.

The implementation of mutations is also extremely random. There is a variable that defines mutations rate, which is simply how many mutations happen per generations. Each mutation is on a randomly chosen "creature". This creature has a random gene chosen, any individual bit, and has it changed randomly to either a 0 or 1. Due to this, it is absolutely possible for an animal to have no change in their genome, however it is fixed with a higher mutation rate. Too high makes it hard to have a consistent level of growth however so there is a sweet spot here as well.

1.3 Selection

The entire process up until now has been random. The creation of "creatures", the creation of offspring, the application of mutations, etc. Everything has been entirely random and you would get nowhere without the process of selection. This is where you get to make your divine intervention. You have to make your process of a fitness testing, and that guides your natural selection.

For this program, I already had a pre-planned goal in mind. As it was more to gain familiarity, it was to see if I could play God to some degree. As a result, I made this in order to try to have a creature that had a genome of all 1s. This is why for the natural selection process, I merely deleted the smallest number, and repeated that process until enough numbers had been deleted.

2 Conclusion

This was overall a very educational experience and I learned a lot. I don't quite know where to go from here, however I am fairly sure that most of the aspects of a basic evolutionary algorithm have been covered. I know this is simply the most basic of basics, however it is a starting point and I am interested in possibly developing something more complex. Let me know if you have any questions or responses. Thank you!