Genetic Algorithms and Genetic Programming

Overview

1. Genetics Review
2. General Genetic Algorithm
3. Implementation Examples

Genetics Review

An *'organism'* or *'individual'* is a form of life such as a plant or animal. Chromosomes carry biological information or characteristics of an individual. During reproduction, there is several things that happen to form the child individuals. There is a *'crossing over'* where genetic material is exchanged and there is a small, random chance of a mutation. *'Natual selection'* is the process by which organisms with traits that are better suited to their environment reproduce in higher numbers, increasing the prescense of those traits.

In a Genetic Algorithm (GA), *'individuals'* are canidate solutions. Each solution has *'chromosomes'* or a unique identifier that represents its corresponding canidate solution. *'Natural selection'* occurs through a fitness function that determines the overall fitness of the canidate solution. *'Crossing over'* happens by exchanging information between the identifier for the solution. *'Mutations'* happen by a bitflip, or some other change to the information representing a solution.

General Genetic Algorithm

Algorithm 1 Generic Genetic Algorithm

```
i = 0                                                        ▷ Generation counter
initialize population P_0                                    ▷ Decide on sample size; n
repeat
    Evaluate fitness of each indivitual in population P_i ;  ▷ Fitness function, f(x_i)
    Select individuals for reproduction based on fitness ;
    Perform crossover and mutation on the selected individuals ;
    i++;
until terminal condition is met ;      ▷ A certain number of generations or average fitness
```

The probability of selecting a sample, x_i for reproduction, with n being the sample size, is defined by:

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} f(x_j)}$$

Implementation Example

Suppose we have this *fitness function* and the goal is to *maximize* it.

$$f(x) = \sin\left(\frac{x\pi}{256}\right) \text{ with the interval } 0 \leq x \leq 255 \ \& \ x \in \mathbb{Z}.$$

We can represent canidate solutions with 8 bits; a byte. Since the goal is to maximize this function, and the values are locked between $0-1$, we will just use the value of the function to represent fitness for a canidate solution.

The first step is to select a population size, which will be 8. Then, to randomly select individuals from the population, we will generate random numbers in our interval, $0-255$.

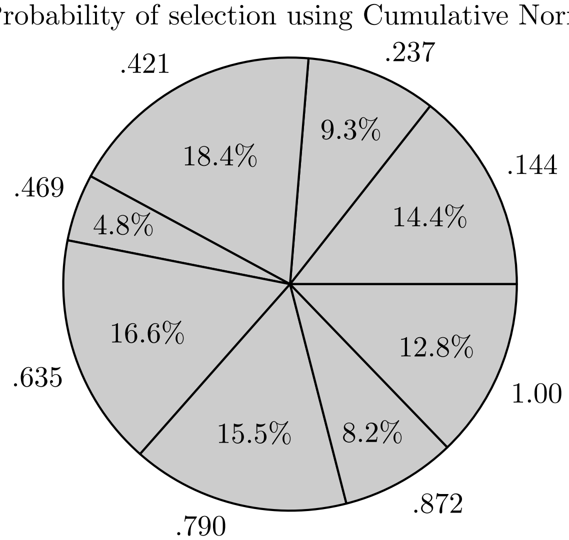
Table 1: Normed Fitness

x	Representation	$f(x)$	Normed $f(x)$	Cumulative $f(x)$
189	1 0 1 1 1 1 0 1	.733	.144	.144
216	1 1 0 1 1 0 0 0	.471	.093	.237
99	0 1 1 0 0 0 1 1	.937	.184	.421
236	1 1 1 0 1 1 0 0	.243	.048	.469
174	1 0 1 0 1 1 1 0	.845	.166	.635
74	0 0 1 0 0 0 1 1	.788	.155	.790
35	0 0 1 0 0 0 1 1	.416	.082	.872
53	0 0 1 1 0 1 0 1	.650	.128	1.000

To start selecting samples, find each sample's *Cumulative Normed Fitness* which will allow for selection using a random number generator between 0-1. The *Cumulative Normed Fitness* should always add up to 1.0, after normalizing fitness.

$$\sum_{i=1}^{n} P(x_i) = 1.0$$

Probability of selection using Cumulative Norm



To select an individual, generate a random number between $0 - 1$. Then find the two numbers that surround your selection. The individual who is being represented by the larger of the two is selected for reproduction.

Table 2: Parent-Child Crossover Table

Selected Numbers	Parent	Children
$3 \cdots 6$	0 1 1 ‖ **0 0 0** ‖ 1 1	0 1 1 ‖ **1 0 1** ‖ 1 1
$3 \cdots 6$	0 0 1 ‖ **1 0 1** ‖ 0 1	0 0 1 ‖ **0 0 0** ‖ 0 1
$1 \cdots 5$	1 ‖ **0 1 0 1** ‖ 0 0 1	1 ‖ **1 1 1 0** ‖ 0 0 1
$1 \cdots 5$	0 ‖ **1 1 1 0** ‖ 1 0 1	0 ‖ **0 1 0 1** ‖ 1 0 1

In this example, two random numbers between zero and the bitsize are chosen, and the two parents exchange the bits between the two selected points.

Table 3: Mutation Table

Individual	Mutation	New x
0 1 1 **1** 0 1 1 1	0 1 1 **0** 0 1 1 1	103
0 0 1 0 0 0 0 1	0 0 1 0 0 0 0 1	33
1 1 1 1 0 0 0 **1**	1 1 1 1 0 0 0 **0**	240
0 **0** 1 0 1 1 0 1	0 **1** 1 0 1 1 0 1	109

Each individual has a very low percentage to mutate. For this example, a bit flip will be used as a mutation. After mutation occurs, the newly generated individuals are the 'next generation' and the reproduction process starts again.