Genetic Algorithm

Prasangsha Ganguly

December 29, 2020

Optimizing a Function Using Genetic Algorithm

Genetic Algorithm is a powerful heuristic optimization tool. A set of genes or feasible solutions of the problem constitute the set of population chromosomes. The first step is to initialize the population. Then the objective value of the population is evaluated. After that, using multiple modifications of the chromosomes by *crossover* and *mutation*, the objective value is improved. The following algorithm provides a framework of the genetic algorithm.

Algorithm 1: GA

```
    begin
        /*Initialize the time and the population $P(t)$*/ ;
1       $t \leftarrow 0$;
2       Initialize($P(t)$);
3       Evaluate($P(t)$);
4       while not termination do
5           $t \leftarrow t + 1$;
6           /* Select a subset of the population from the last iteration population set*/;
7           $P(t) \leftarrow Selection(P(t-1))$;
8           Crossover($P(t)$);
9           Mutation($P(t)$);
10          Evaluate($P(t)$);
```

GA can be used to optimize a function. Consider, the following function:

$$z = (4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (-4 + 4y^2)y^2$$

Here, a six hump camelback function is given where both x and y lie in the range $[-5, 5]$. The objective is to minimize z.

First *Encoding*, the x and y, which are real variables have to be discretized. We encode x and y as 8 bit strings. Hence, there are $2^8 - 1$, i.e., 255 possible values of x and y. Now the range of x i.e., $[-5, 5]$ is divided into 255 equally spaced values. So, 00000000 represents -5 and 11111111 represents 5. The difference between two consecutive values of the encoded strings is $\frac{5-(-5)}{255} = 0.039$.

For any chromosome if encoded as $< c_7c_6c_5c_4c_3c_2c_1c_0 >$, then the value corresponding to that chromosome is $((\sum_{i=0}^{7} c_i * 2^i) * 0.039) - 5$. That is the (decimal number corresponding to the binary string - 5). For example, $< 00000001 >$ is -4.961.

Second $Initialize(P(t))$. We create a random population of chromosomes or encoded strings of x and y out of the 255 possible values. Then we $Evaluate(P(t))$. We identify the z values for the chromosome pairs (x_i, y_j) for all the possible pairs. Hence, if the population of x has p_x values and the population of y has p_y values, then we are going to evaluate $p_x * p_y$ pairs of (x, y) values for their corresponding objective. After, the z values are obtained, the candidate solutions (x, y) are ranked according to the z value with lower is the better because we are minimizing the objective.

Third $Selection(P(t + 1))$. Now, from the previous population, we have to select the best candidates with higher probability and worse candidates with lower probability. Essentially, we are making copies of the previous population but some candidates (worse) of the previous population will not have a copy in the new population and some candidates (better) will have more than one copy in the new population. This can be implemented as, we can create a roulette wheel with the areas proportional to the fitness function. That is the best candidate gets maximum area and worst candidate gets smallest area. We roll the wheel $n = p_x * p_y$ times and identify which candidate is selected. By this way, we create n new candidates where a better candidate of previous population has more number of copies compared to worse candidates of the previous population.

Fourth $Crossover(P(t))$. This is a mating of two chromosomes to create two new chromosomes. The two children chromosome gets a part from their mother and the other par from their father. Say we have the mother chromosome encoded as $< m_1m_2m_3m_4m_5m_6m_7m_8 >$, and the father encoded as $< f_1f_2f_3f_4f_5f_6f_7f_8 >$. Then, we randomly select a point of crossover say, 5. Then, the first child gets first 5 bits from their mother and last 3 bits from their father. Whereas, the second child gets first 5 bits from their father and last 3 bits from their mother. So, $child_1$ is $< m_1m_2m_3m_4m_5f_6f_7f_8 >$ and the $child_2$ is $< f_1f_2f_3f_4f_5m_6m_7m_8 >$.

Fifth, $Mutation(P(t))$. Often the population miss a gene, or say, how new species emerge from a population? To address this, we have random and rare mutation. Say, in a population, for all the parents encoding, the third bit is a 0. Now, no matter what cross over we do, the children will always have third bit as 0, because all the parents have 0 in the third bit. To address this problem, we randomly change a bit in the candidates but we do this rarely.

Now, we get a new set of chromosomes, we can consider this whole set as my population. Or, do another thing. Say in $P(t - 1)$ we have chromosomes with fitness values sorted as $\{c_1, c_2, ...c_n\}$ with fitness values $\{0.99, 0.67, ..., 0.01\}$. Now, after we do crossover and mutation, we obtain $\{d_1, d_2, ..., d_n\}$ with fitness values $\{0.88, 0.76, ..., 0.2\}$. We loose c_1 with very high fitness. But we don't want that. So what we could do is, keep the best chromosome of the parent population and replace $n - 1$ chromosomes of parent by the top $n - 1$ child chromosomes. That is instead of considering $\{d_1, d_2, ..., d_n\}$ as the new population, we consider, $\{c_1, d_1, ..., d_{n-1}\}$ as the new population.

Essentially, we put them in a loop with a termination criteria on number of iterations say.