

Genetic algorithm

Tuesday, September 19, 2017 01:05

For the glory of God

Introduction

- Genetic Algorithm (GA) is inspired by an analogy with natural selection in Darwin's theory of evolution.
- The invention of Genetic Algorithm is credited to John Holland (professor at University of Michigan)
- Steps in Genetic algorithm are as following :

Initialization, Selection, Reproduction (crossover and mutation), Fitness evaluation, replacement

Genetic Algorithm

a) Initialization

(within that population, certain individuals are selected for reproduction)

- First of all, a population of individuals must be generated to initialize the problem.

A family of possible solutions Many design points are represented as individuals in a population in GA. ↓
Then, how many individuals do we need to make healthy population? → size of binary string

In general, the more points, the better diversity. A rule of thumb is $2n < m < 4n$; where n = Binary digit i.e. $n=4$ for 0010

- The population size depends on the nature of problem, but typically contains several hundreds and thousands of possible solutions.

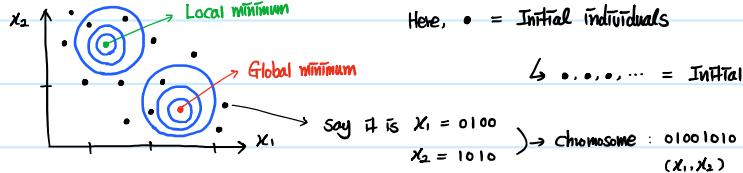
↳ Often, the initial population is generated randomly but occasionally, the solution may be seeded in the areas.

The initial population of m individuals is generated by randomly sampling bits in each chromosome.

e.g. 1000, 0100 \Rightarrow Concatenating \Rightarrow 10000100

Concatenating the binary numbers for each variable FORMS a single string of n binary digits known as the chromosome for the individual.

e.g.



- If the problem involves continuous variables, these should be discretized over a desired range and to a desired resolution as binary numbers.

↳ It is preferable to represent the binary numbers in Gray code.

b) Selection

- Once we have initial population, the next step is to evaluate individuals in the population using the fitness function.

The formula that is used to calculate the fitness on the basis of chromosome values is called as Fitness function.

(Better the solution, higher the fitness score)

→ To write the fitness function, it is necessary to know the detailed description of the problem. (problem dependent)

e.g. let's say we want to minimize $Z = x^2 + y^3$, if $x=2 / y=1$, then $Z=5$, say fitness is 5.

Once we finish to evaluate all individuals in the population, there are two common forms for selecting individuals for reproduction:

Selection

(Initial population)

They will be survived



• Once you have a design, how do you choose which gonna be reproduced?

• The first way is Proportional Selection. The general idea behind it can be stated as; The expected number of selections of an individual for reproduction is proportional to its fitness.

⇒ If something has ^{good} healthy fitness, it could have many offsprings. (∴ Reproduced)

In other words, fitness is intended to be maximized.

• However, we might have some problems if we just try to relate the fitness to objective function because our nature objective function is **non-linear** which causes some problems on proportional selection.

e.g. $\text{f}(x)$

↳ It may be related to the local minimum.

• For this reason, generally speaking,

- the measure of fitness should monotonically increase with $f(x)$ if the optimization goal is to maximize

- the measure of fitness should monotonically increase with $-f(x)$ if the optimization goal is to minimize

→ based on fitness

- A common form of proportional selection is called roulette wheel

selection, which is not totally random method but it is;

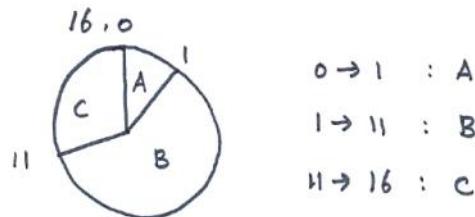
→ Each candidate is given a slice of an imaginary roulette wheel with the size of the slice proportional to its fitness.

A random number is then generated to select a patient

- For example, let's say

fitness 1, 10, 5

- we have three individuals among a population A, B, C with respectively.



0 → 1 : A

1 → 11 : B

11 → 16 : C

- Let's say that two random numbers between 0 and 16 were generated.

(12 and 3)

→ They will be reproduced to

- Then, the two patients will be **C** ($12 \in 11 \rightarrow 16$) and **B**. have kids.

- What if we have only one patient (say two random number 4, 8) B?

: It could be worked in the algorithm; however, this is just simple

case. In most situations, we will have a lot of individuals.

There will be still two patients.

→ Next page

→ 2017 Spring (professor Ligutsey) keep going on Selection:

- In most cases, Proportional selection is working well; however, it sometimes encounters issues when there are very large or small differences in the fitness function between individuals.
- For example,
 - If large differences, a candidate would dominate the pool so that it diminishes the ability to explore, that isn't what GA wants.
 - If small differences, the roulette wheel becomes nearly evenly spaced so that it hinder explores
- Sometimes those problems can be resolved when we use "monotone-preserving" transformations, which follow the trends but avoid the problem, or normalization to overcome sensitivity issue.
- The second way is "Tournament selection" which overcomes some of the difficulties of Proportional selection. As its name implies, tournament selection holds tournament between different individuals to determine which will be reproduced.

→ "head to head fitness"

- There are two general types of Tournament selection "competition"

① Deterministic tournament (It still random but..)

- The outcome is completely predictive based on the fitness function

↳ Better fitness function always wins to others (Every time)

- Tournament size → ~~Individual chosen~~ → Tournament → highest win
The K individuals participate in a tournament.

② Stochastic tournament

K (arbitrary) individuals are chosen at random from the population.
⇒ This is the tournament size.

Stochastic tournament selection

Let's say

How it works for a binary tournament ($k = 2$):

- ❖ 2 individuals are chosen at random from the population.
- ❖ Select the most fit individual with a probability $0.5 \leq p \leq 1$
- ❖ Choose the second most fit individual with probability $(1 - p)$

- Even though both proportional and tournament selection, there is no guarantee that the individuals in the population with the best fitness will be selected as parents.

↳ There are several strategies for implementing Elitism, which is the concept of copying the best individuals from one generation to subsequent ones.
↳ either before or after reproduction

- Anyway, now we have selected the design where we can reproduce it.

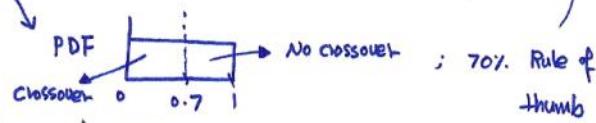
Elitism involves copying a small portion of the fittest candidates, unchanged, into the next generation.

This can sometimes have a dramatic impact on performance by ensuring that the GA doesn't waste time to re-discover the discarded one.

c) Crossover

Crossover

- We're ready to reproduce the generation. How can we combine the design information to produce new designs which are our offspring designs? (Enabler: Crossover)
 - ↳ This is ultimately how we optimize.
- Crossover is the genetic recombination of two parent individuals.
 - After selecting two parents, a random number is generated to determine if crossover occurs. If no crossover occurs, both parents simply become children.
 - There are three common types of crossover to implement to GA
- ① One point crossover
 - Selecting two parents
 - A random integer k between 1 and $l-1$: where $l = \text{chromosome length}$
 - ↳ In this case, $l = 8$ and let's say $k=4$ (1 to 7)
 - Crossover is definitely happened because any integer should be in the range. (Hence, parents and childs)



→ Definitely gray code

Parent A : 01110011
Parent B : 11001100 ($l=8$)

A random integer k between 1 and $l-1$: where $l = \text{chromosome length}$

↳ In this case, $l = 8$ and let's say $k=4$ (1 to 7)

Crossover is definitely happened because any integer should be in the range. (Hence, parents and childs)

(卷四)

- It is good ; however, Just swapping the left / right bits causes less diversity in design Exploration. For this reason second-point (two-point) crossover is introduced.

② Two point crossover

Samplers is good but on the edges,
it's hard for chromosome to a bit swap

- It is similar to one-point crossover except two random number between 1 and $\ell - 1$ are generated.
 - The first random number represents which to start the swap
 - The second number represents how many bits to swap from the start
 - For example, let's say 1st random number = 7 and 2nd = 3

Patients A : 00110011, Patients B : 11001100

Start

how many bits to swap

↓

Patients A : 11 1100 10 Patients B : 00 0011 01

Child. Child.

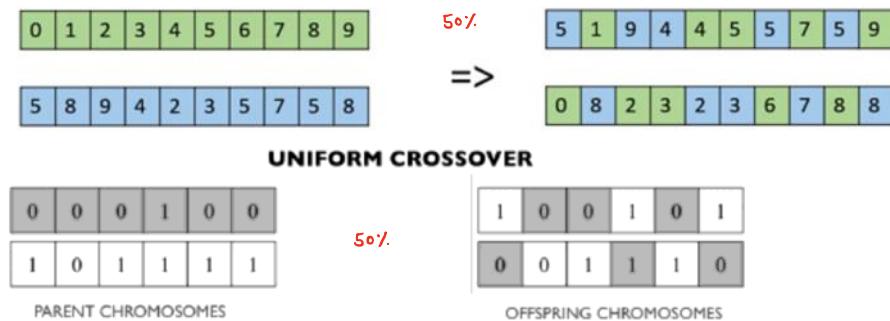
- Two-point crossover is less sensitive to the ordering of individual design variable in the overall chromosome

③ Uniform crossover

- It is based on the fact that each bit is given a probability to swap.
- The typical probability is $p = 0.5$ (It could be changed)

Uniform Crossover

In a uniform crossover, we don't divide the chromosome into segments, rather we treat each gene separately. In this, we essentially flip a coin for each chromosome to decide whether or not it'll be included in the off-spring. We can also bias the coin to one parent, to have more genetic material in the child from that parent.



Mutation

- Mutation is something different compared with crossover.



It's just changing the bit



try to combine two things

- What if you have too much mutation? What is a problem?

If it is too much, it will be too randomly.

- Then, why do we need to consider Mutation in Genetic Algorithm?

First of all, please keep in mind that we will have a bunch

• If it is too much, it will be too random.

- Then, why do we need to consider Mutation in Genetic Algorithm?
 - : First of all, please keep in mind that we will have a bunch of individuals. After doing crossover, there might be someone who dominates the pool.
 - In other words, we want to preserve the variability.
(I don't want to cut off my variability)
- In order to maintain genetic diversity, mutation is added in the form of randomly flipping a few of the bits in the children's chromosome
 - e.g. ❖ Similar to crossover, give each child a small probability to mutate (about 10%). If a child is chosen to mutate, randomly flip a user-defined number of bits in the child's chromosome. The locations within the string for the bit flips are chosen randomly.

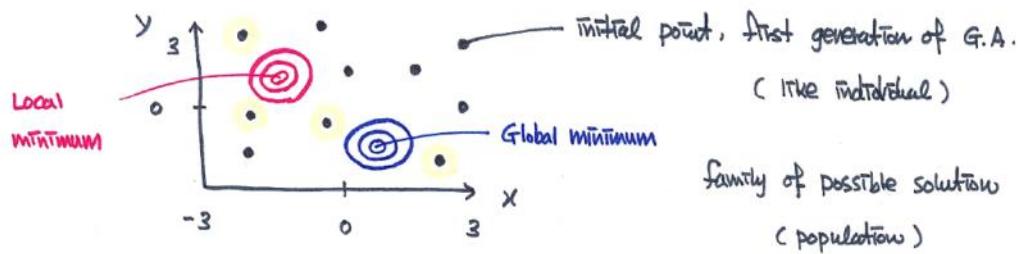
Replacement

- Let's say now we have parent design and child design.
- We might need to have a rule which design is going to be survived for the next generation.
- There are two ways to replace it for next generations:
 - Get rid of all parent design (or kill them) and only consider child design as a new parent (Like Biology)
 - Select the best parent / child individuals and form the subsequent population.

Overview of Genetic Algorithm

Genetic Algorithm overview (Youtube)

1. Randomly generate initial individuals and a population

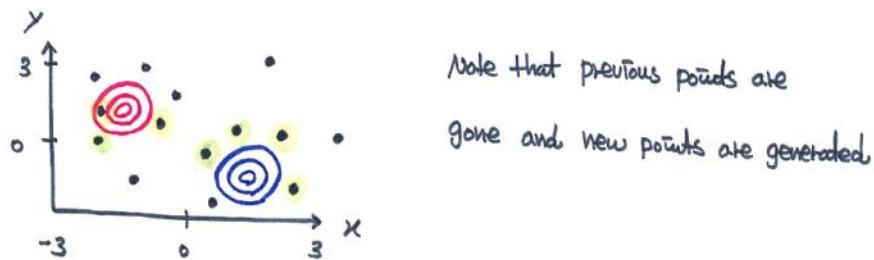


2. Evaluate initial population and select a few good solutions

Yellow : Selected because of for reproduction

Nearest to the optimum (or survived because they had good fitness)

3. Generate new population (like mutation) and Evaluate



4. Repeat the process for good solutions for reproduction

Yellow : selected

5. Continue process until stopping criteria are met.

