Computational Maths 2

Introduction to Numerical Optimization

Beniamin Bogosel

Universitatea Aurel Vlaicu Arad

Optimization in dimension 1

- Genetic Algoritms

Optimization in dimension 1

- Genetic Algoritms

Introduction

https://www.youtube.com/watch?v=gVEWaOtEASM

- Mimic evolutionary behavior
- Many processes are random, but the level of randomization is controlled.
- More efficient than random search or exhaustive algorithms
- Can find global minima compared to local minima for gradient based algorithms
- Can find minimizers even in discrete settings where we don't have continuity, linearity or other features that we can exploit

Structure of Genetic Algorithms

- fitness function to optimize: analogue of objective function
- population of chromosomes: a set of variables that correspond to inputs for the objective function
- selection of which chromosomes will reproduce
- crossover to produce next generations of chromosomes
- random mutations for chromosomes in the new generation

Fitness function

$\star$ This is the function to be minimized
Example:

$$\min_{x \in \mathbb{R}} x^2 + \sin(x).$$

The fitness function is: $f(x) = x^2 + \sin x$.

Chromosomes

⋆ Chromosomes represent the optimization variables.
⋆ For example, if $x \in \mathbb{R}^n$, $x = (x_1, ..., x_n)$ then the chromosome is simply: $(x_1, ..., x_n)$
⋆ Alternatively: we can work in binary, convert variables and consider binary bits as "chromosomes"

$$x \mapsto 100110111011110100110101$$

Population, Selection

⋆ a fixed number N of variables are kept in memory for a given **generation**
⋆ each member of the population is a variable for which the **objective function** can be evaluated
⋆ from each generation we may wish to keep only a number of individuals which give the best objective function to **create new individuals**

Some options

- keep the best N_0 individuals, use them and possibly other individuals to create new ones
- Define a probability function which indicates how likely it is to use the current individual to create **new ones**

Crossover operator

Question: starting from two individuals x, y how can we construct new ones?

- Simplest option: convex combinations $\alpha x + (1-\alpha)y$ where α is chosen randomly (drawback: we cannot explore the space outside the given population)
- More intricate options: binary crossover

$$x = 10101110101101110110101$$
$$y = 01010011011011101011011$$

  choose one or multiple **crossing points** and flip the bits between x, y alternatively at these crossing points

- Any other idea of passing information between x and y
- Sometimes, the structure of x and y needs to be preserved (e.g. Traveling Salesman problem; see projects list)

Mutation operator

$\star$ keeping best individuals and using crossover between them is not enough sometimes to explore all the space.
$\star$ introducing random mutations may help explore the space of admissible variables even further

Examples:

- add a **random number** to the current one (same for multiple variables)
- flip one or more bits in the binary representation

Genetic algorithm pseudocode

⋆ Initialize the population P_0 of size N, choose a number of **generations** (analogue of iterations)

⋆ For each iteration do:

- Compute the fitness/objective function for each member of the population
- Apply the **Selection operator**: decide which individuals to keep, which to use in the crossover
- Keep best $N_0 < N$ individuals; create new ones using crossover and mutation operators
- Apply the **Crossover operator** for the selected individuals; for each new individual introduce a **Mutation** with a certain probability $p \in (0, 1)$.
- Keep creating new individuals until we have a new population of size N
- Go to the first step.

Practical examples in 1D

$f(x) = x^2$, $f(x) = 0.3|x| + \sin(x)$

$\star$ Crossover1: convex hull, Mutation 1: add random number

$\star$ Crossover2: switch bits in binary, Mutation 2: flip random bits in binary

See Notebook!

Traveling Salesman

Problem: Assume there is a traveling salesman which needs to visit n cities via the shortest route possible. He visits every city once then returns to the initial one.

- Cities may be labeled City1, City2, ..., City n
- We search for an optimal path between these cities
- Chromosomes, or optimization variables, are permutations of integers from 1 to n
- Iterating through all permutations cost $n!$ which is huge.
- $10! = 3628800$, $20! = 2432902008176640000 \sim 10^{18}$
- assume the cost for a path between c_i and c_j is $d(c_i, c_j)$, the distance between cities c_i, c_j

Crossover for permutations?

⋆ simple swapping: not ok

Parents	Normal Crossover	Offspring (faulty)
3 5 1 2 4	3 5|1 2 4	3 5 5 3 2
1 4 5 3 2	1 4|5 3 2	1 4 1 2 4

⋆ cycle crossover:

Parents	Offspring (step 1)	Offspring (step 2)	Offspring (step 3)	Offspring (step 4)
4 1 5|3|2 6	4 1 5 **2**|2|6	4|1|5 **2 1** 6	|4|**4** 5 **2 1** 6	**3 4** 5 **2 1** 6
3 4 6|2|1 5	3 4 6 **3**|1|5	3|4|6 **3 2** 5	|3|**1** 6 **3 2** 5	**4 1** 6 **3 2** 5

- pick random starting position i
- swap cities in x, y on position i
- if $x(i)$ is double then swap again on the position of its double
- repeat until there's no city which repeats

How about mutations?

- ⋆ random transformation which preserves the permutation character
  - injective, surjective: bijective!

- ⋆ pick two cities in the cycle x and **swap them**
- ⋆ the resulting chromosome is again a cycle.
- ⋆ can you imagine more complex mutations?