THE UNIVERSITY of TENNESSEE KNOXVILLE

Lecture 06: Vector Spaces and Genetic Algorithms

Sergei V. Kalinin

Universality matters!

Computer vision: Deep convolutional networks

Radiation effects in materials

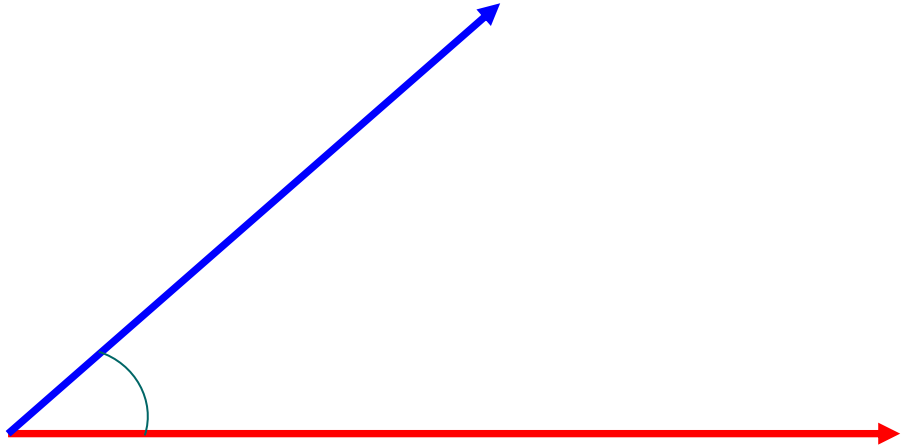
Medical imaging and radiology

Autonomous driving and robotics

Vector – quantity represented by arrow with both direction and magnitude!

https://jordanfernandezdespicableme.blogspot.com/

Vector spaces

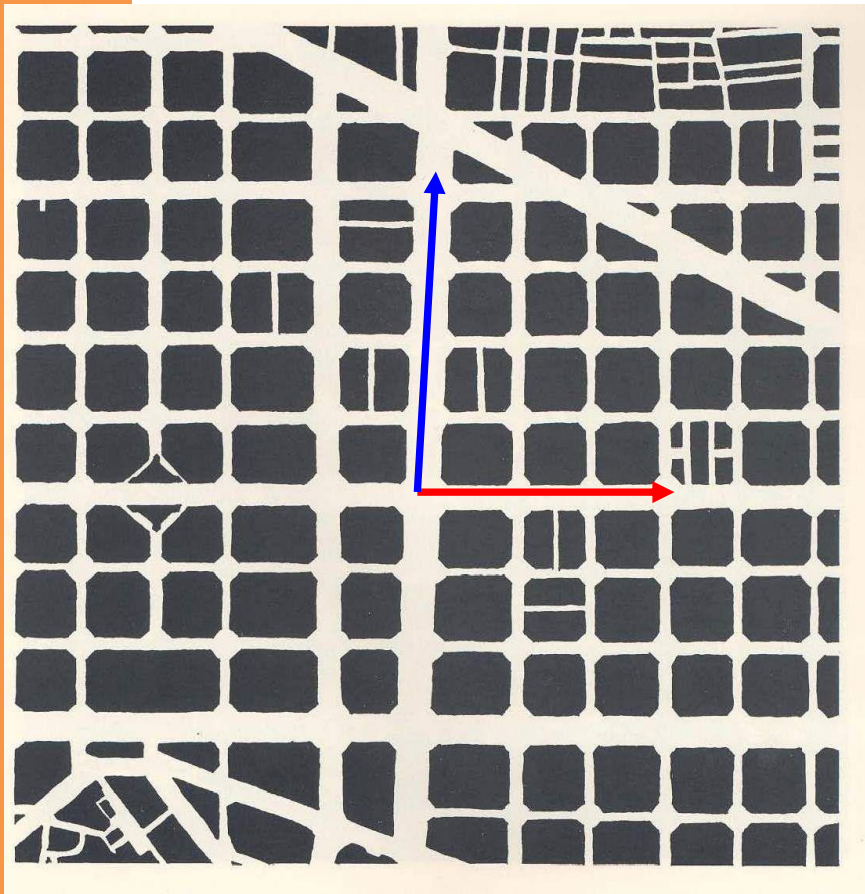


Norm: Defines the length or magnitude of a vector. A vector space with a norm is called a normed vector space. The norm allows for defining the concept of distance and angle between vectors.

Inner Product: An operation that takes two vectors and returns a scalar, often used to define the angle between vectors and their orthogonality. A vector space with an inner product is called an inner product space.

Metric: A function that defines the distance between two vectors. A vector space equipped with a metric becomes a metric space, which is fundamental in analysis and topology.

Examples of vector spaces



- $\mathbf{R^n}$ (The set of all n-tuples of real numbers), e.g. R^2 (all pairs of real numbers), R^3 (all triplets of real numbers), etc. Vector addition and scalar multiplication are defined in the usual way.

- **The set of all *m* x *n* matrices:** Matrices can be added together and multiplied by scalars, and these operations satisfy vector space axioms.

- **The set of all continuous functions:** The set of functions from an interval [a, b] in R^1 to R^1, where addition and scalar multiplication are defined pointwise.

- **The set of all polynomials:** Polynomials of any degree, including the zero polynomial, form a vector space with regular polynomial addition and scalar multiplication.

- **Complex numbers**: With standard addition and multiplication by real or complex scalars.

Examples of not vector spaces

- **The set of natural numbers (N):** This set does not form a vector space as it lacks an additive inverse for each element (e.g., there's no natural number that you can add to 3 to get 0).
- **The set of integers (Z) under usual addition and multiplication:** Similar to natural numbers, integers don't form a vector space as scalar multiplication does not always yield an integer.
- **The set of nonzero real numbers:** This set is not closed under addition; adding two nonzero numbers can yield zero (e.g., 1 + (-1) = 0), which is not in the set.
- **The set of all positive functions:** Functions that are always positive do not form a vector space as the additive inverse (negative of a function) would not be positive.
- **The set of all invertible n x n matrices:** While these matrices form a group under multiplication, they do not form a vector space under matrix addition because the sum of two invertible matrices might not be invertible (e.g., two matrices with determinant 1 could sum to a matrix with determinant 0).

What should we look at practically?

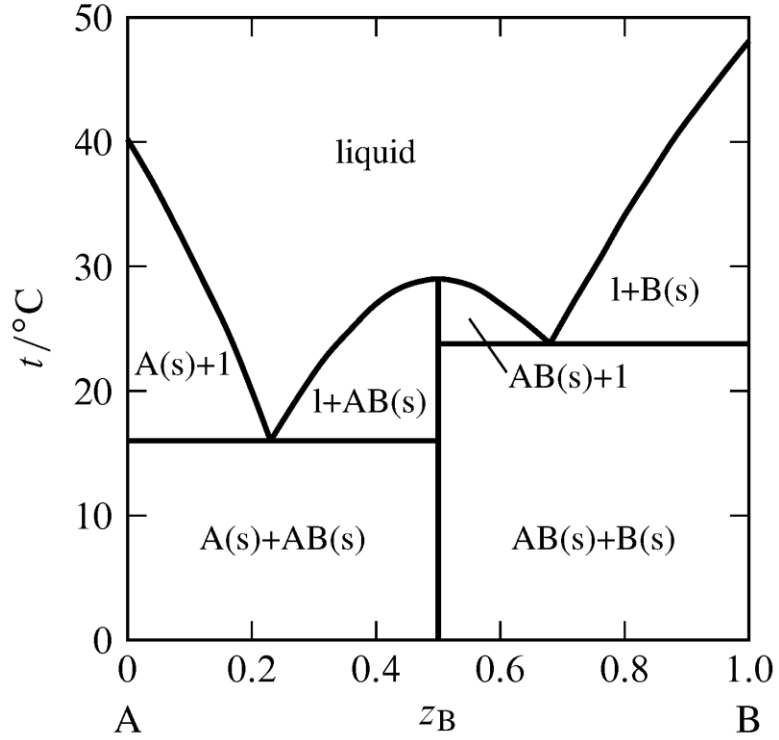
These are important:

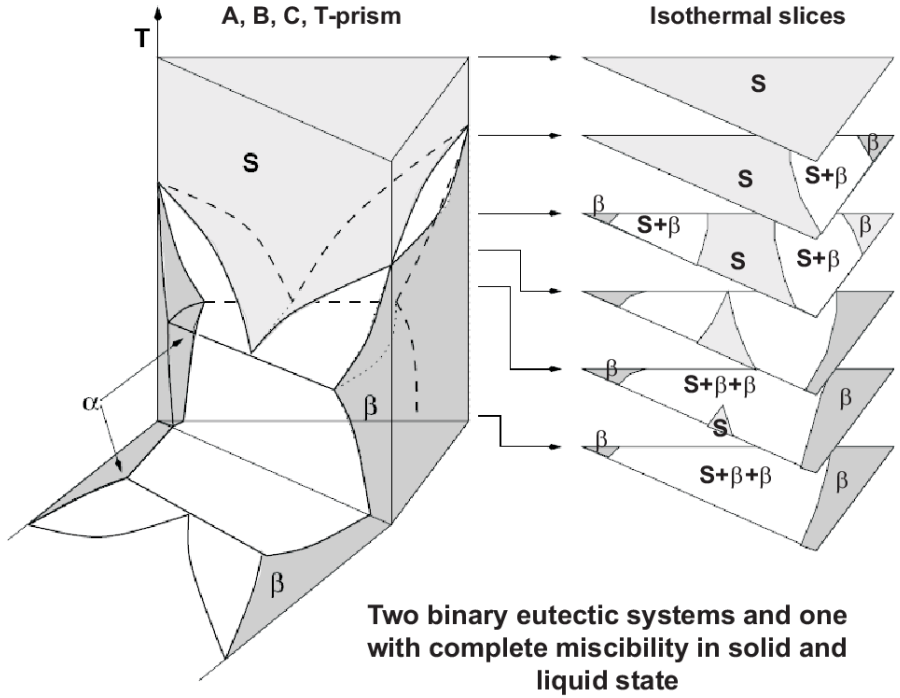
- Dimensionality
- Metrics (Distance Function)
- Continuity and Differentiability

These may be important:

- Symmetry
- Topological Properties
- Inner Product (Hilbert Spaces)
- Manifold Structure

Phase diagrams





Two binary eutectic systems and one with complete miscibility in solid and liquid state

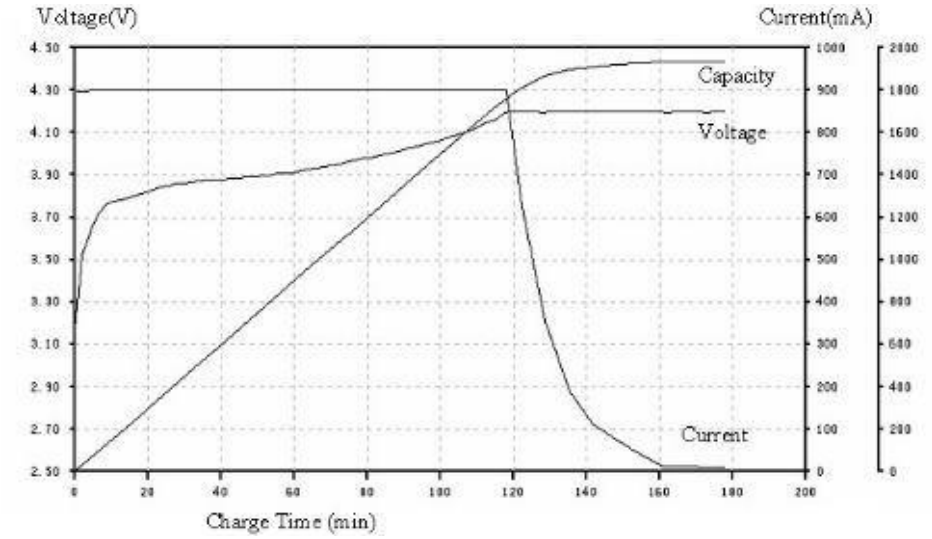
- What is the dimensionality of the parameter space?
- Is it differentiable?
- What is the function?
- Is it continuous?
- Is it differentiable

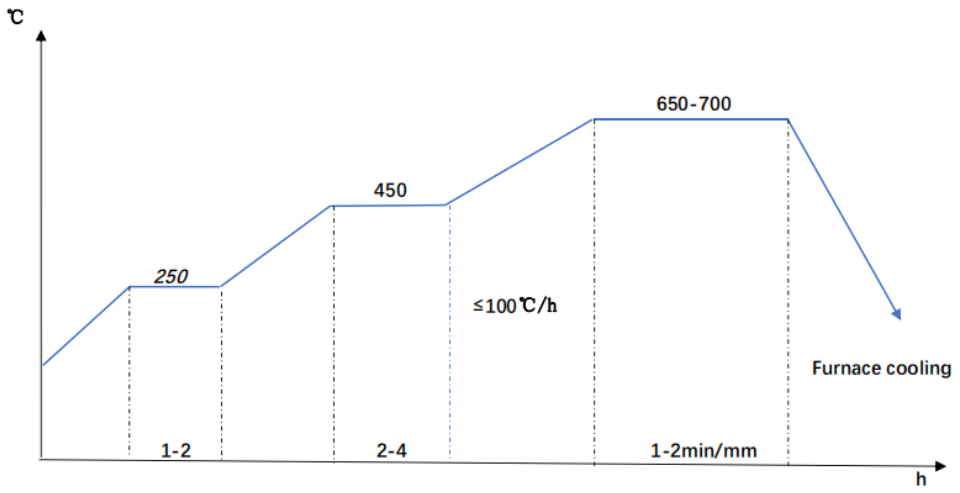
13.2: Phase Diagrams- Binary Systems - Chemistry LibreTexts

https://www.tf.uni-kiel.de/matwis/amat/td_kin_ii/kap_1/backbone/r_se17.html

Processing

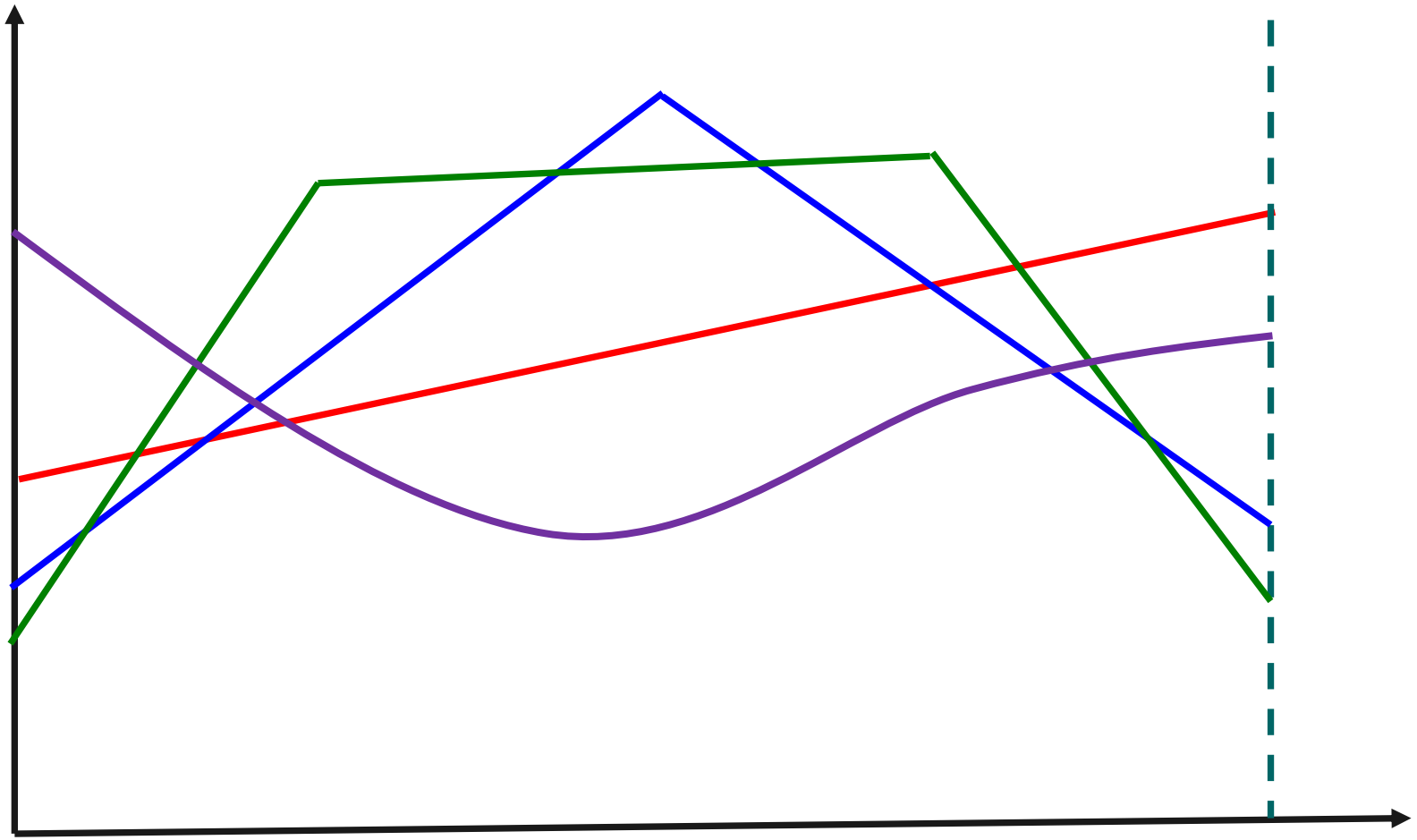
- Making steel – can be complicated and took a lot of time optimize
- Battery charging – fairly simple now, but obvious economic impact
- Annealing hybrid perovskite thin films
- Poling ferroelectric materials
- … and so on





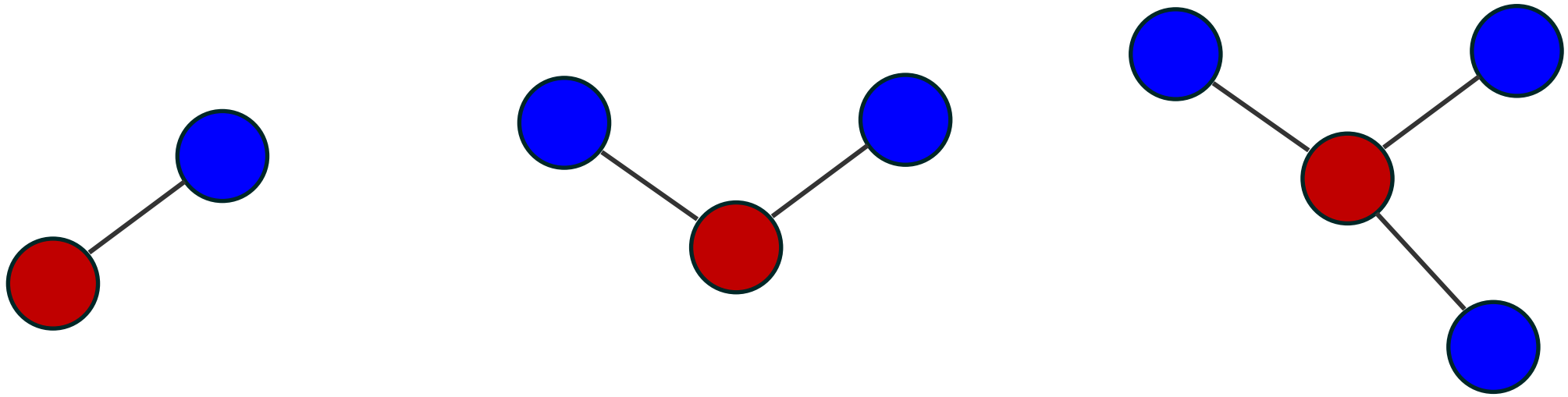
- What is the dimensionality of the parameter space?
- Is it differentiable?
- What is the function?
- Is it continuous?
- Is it differentiable

Space of continuous functions



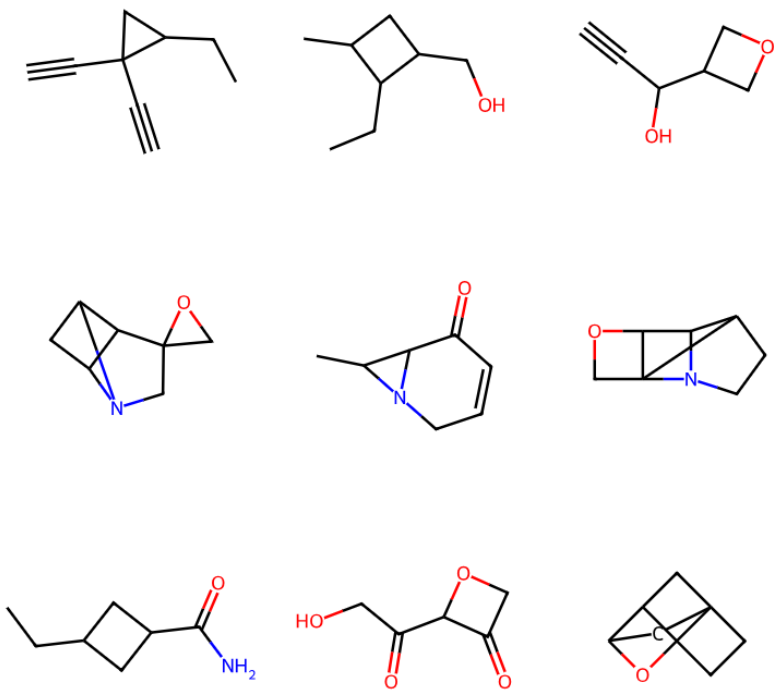
However: if all our functions are linear combinations of several basis functions,
$f = a_1 f_1(x) + a_2 f_2(x) + \ldots + a_n f_n(x)$, when dimensionality is n-1!

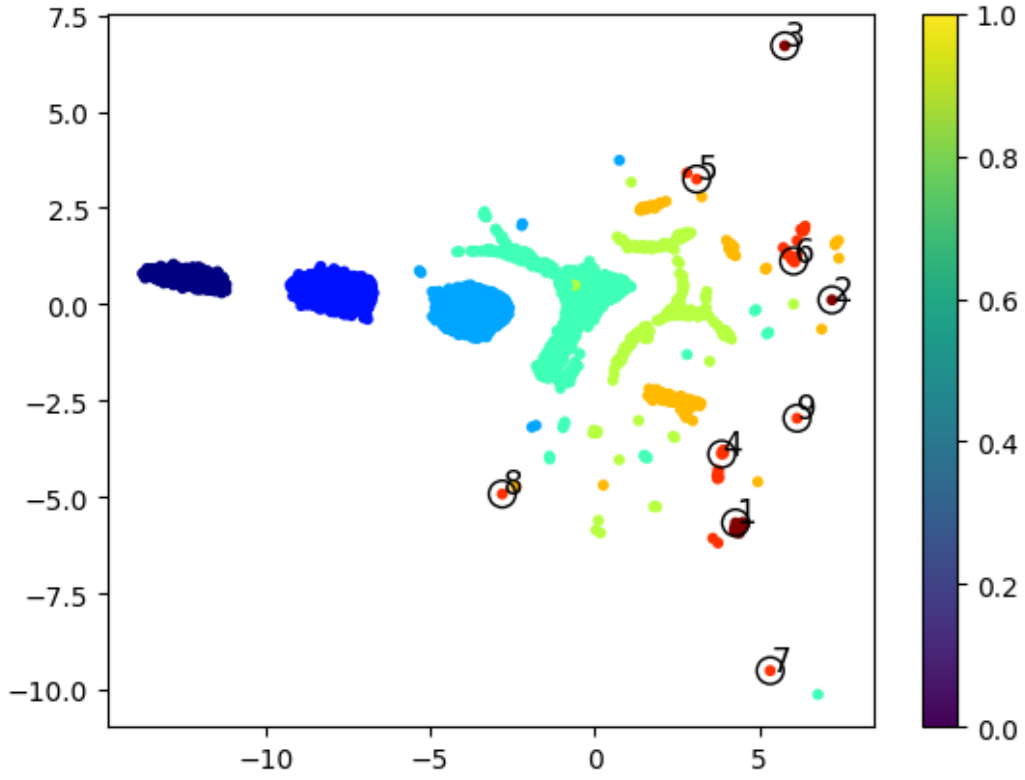
Geometric molecular space



- What is the dimensionality of the parameter space?
- Is it differentiable?
- What is the function?
- Is it continuous?
- Is it differentiable

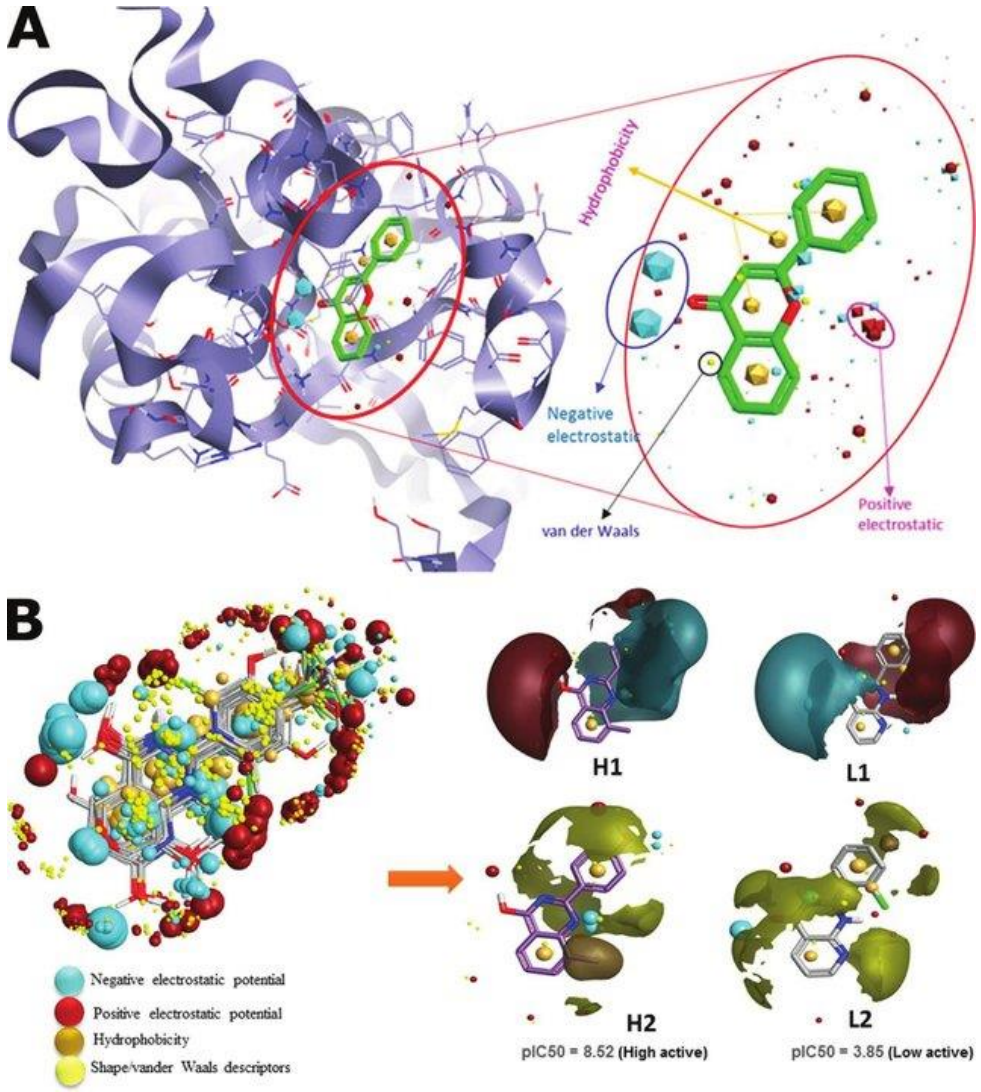
Molecular spaces





- What is the dimensionality of the parameter space?
- Is it differentiable?
- What is the function?
- Is it continuous?
- Is it differentiable

Structure–property relationships

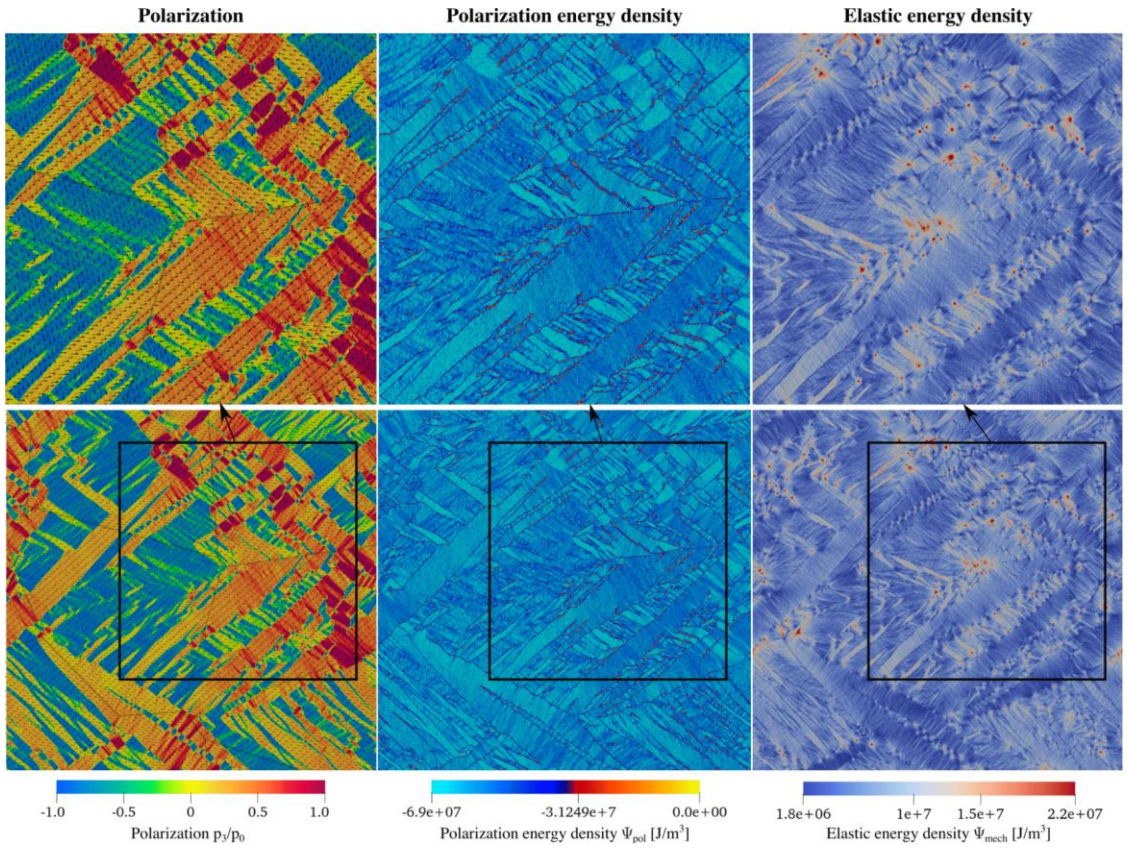


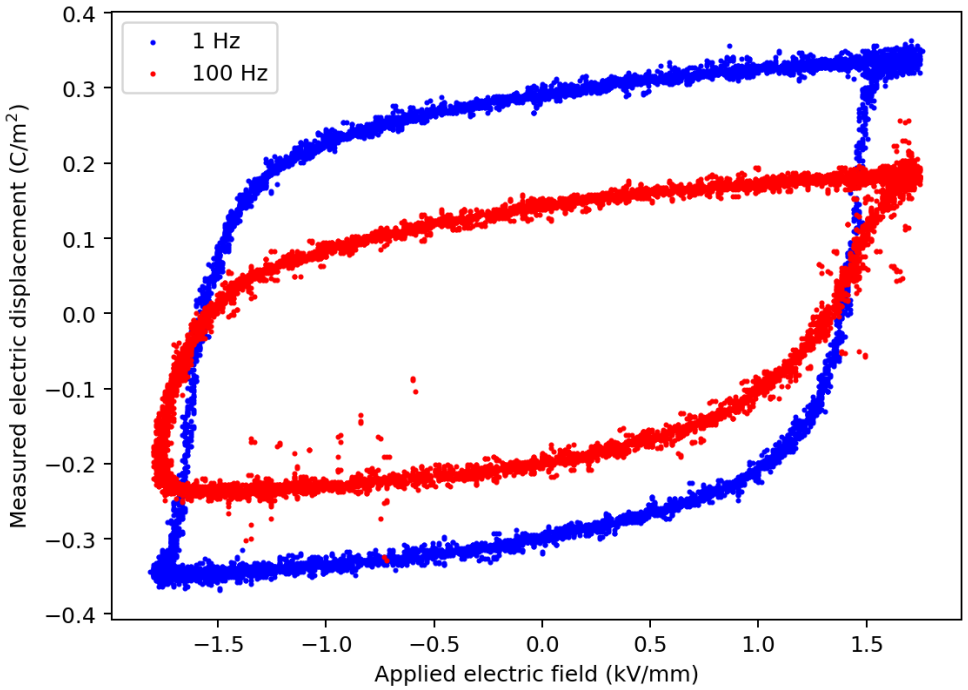
- What is the dimensionality of the parameter space?
- Is it differentiable?
- What is the function?
- Is it continuous?
- Is it differentiable

Hint: activity cliffs in quantitative structure-activity relations (QSAR)

https://www.researchgate.net/publication/342718908_Artificial_Intelligence_and_Machine_Learning_in_Computational_Nanotoxicology_Unlocking_and_Empowering_Nanomedicine/figures?lo=1

Structure–property relationships

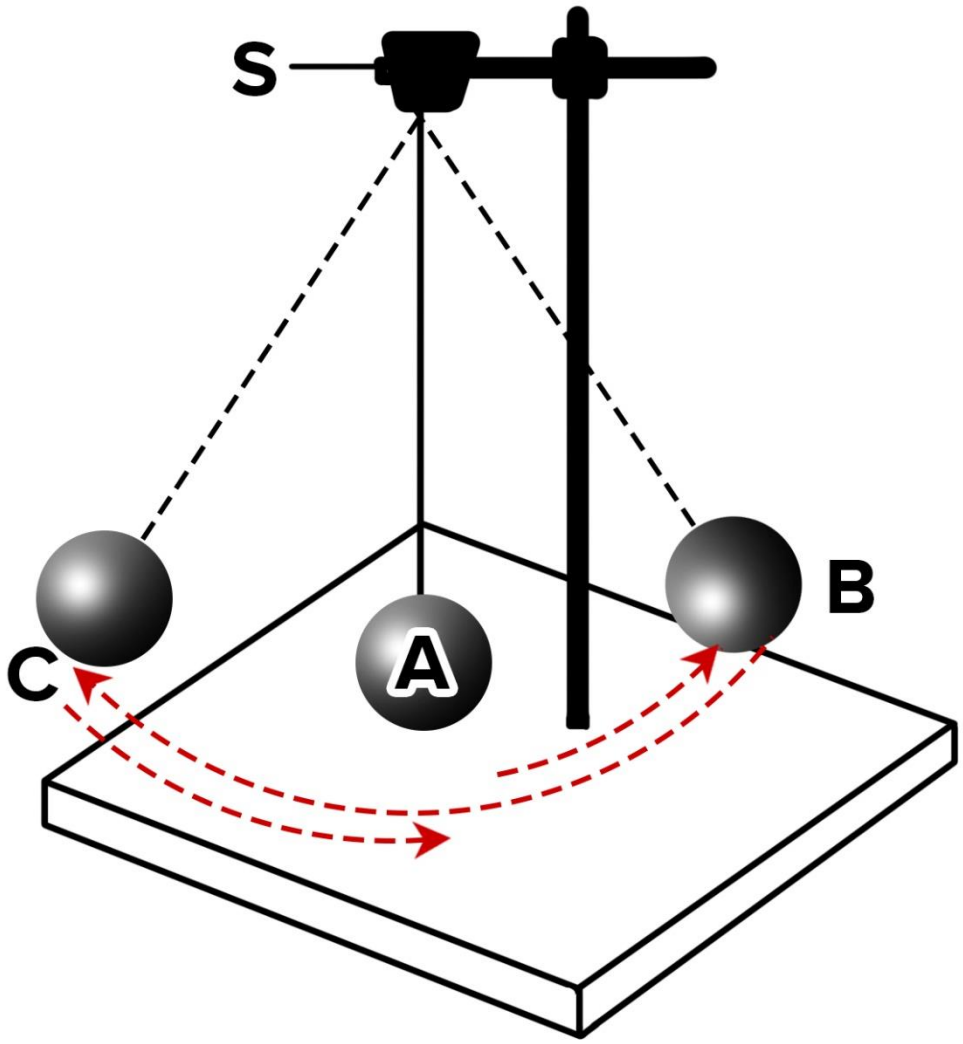




- What is the dimensionality of the parameter space?
- Is it differentiable?
- What is the function?
- Is it continuous?
- Is it differentiable

Back to functions from data

- If the function is known, we can fit it to data
- If we have several possible functions, we can fit all of them and compare the quality of fits
- We can also make some judgements based on the parameter values
- But what if we do not know the functions?



Genetic Algorithms

Genetic Algorithms (GAs) are a part of Evolutionary Computing (EC), which is a rapidly growing area of Artificial Intelligence (AI). It inspired by the process of biological evolution based on Charles Darwin's theory of natural selection, where fitter individuals are more likely to pass on their genes to the next generation.

History of Genetic Algorithms

- The GA, developed by John Holland and his collaborators in the 1960s and 1970s.
- As early as 1962, John Holland's work on adaptive systems laid the foundation for later developments.
- By the 1975, the publication of the book "*Adaptation in Natural and Artificial Systems*", by Holland and his students and colleagues.
- The GA got popular in the late 1980s by was being applied to a broad range of subjects that are not easy to solve using other techniques.
- In 1992, John Koza has used genetic algorithm to evolve programs to perform certain tasks. He called his method "*genetic programming*" (GP)[3].

https://towardsdatascience.com/an-introduction-to-genetic-algorithms-c07a81032547

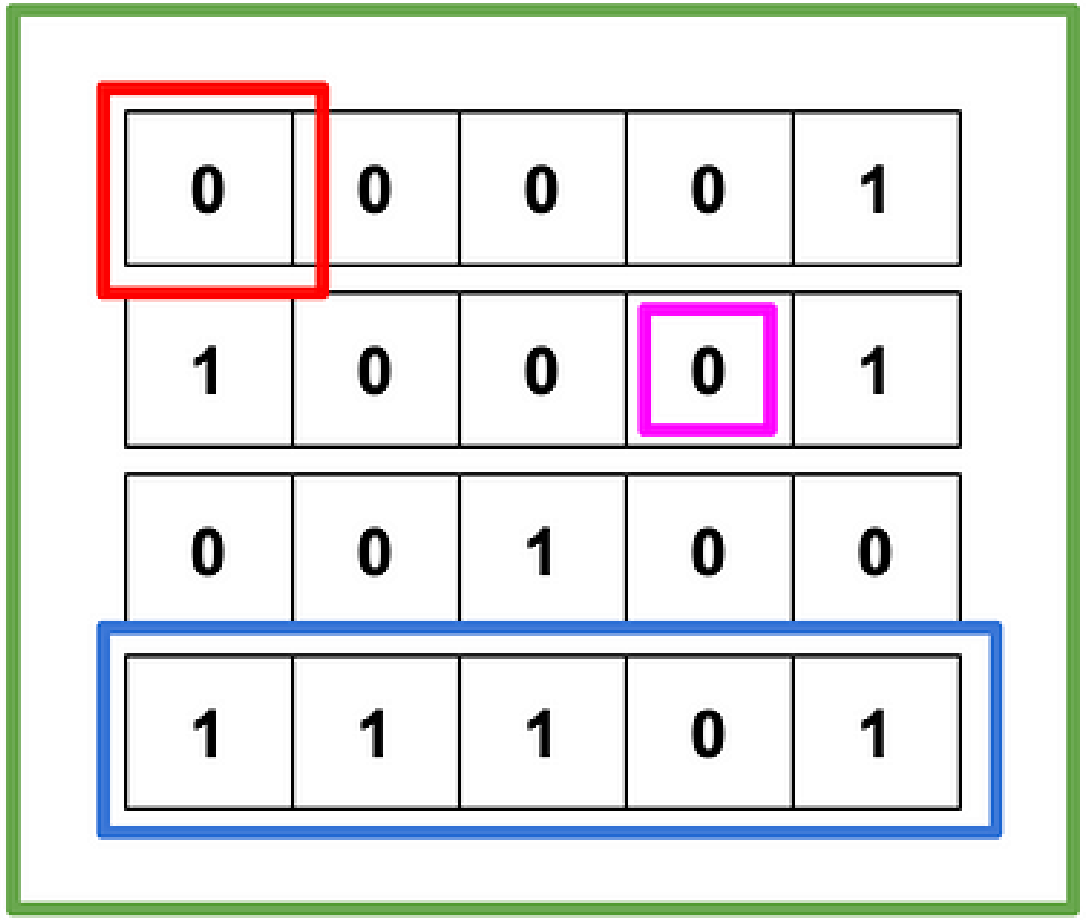
Genetic Algorithms in Nature

- Each cell of a living thing contains chromosomes — strings of DNA.
- Each chromosome contains a set of genes — blocks of DNA
- Each gene determines some aspect of the organism (like eye color)
- A collection of genes is sometimes called a genotype
- A collection of aspects (like eye color) is sometimes called a phenotype
- Reproduction (crossover) involves recombination of genes from parents and then small amounts of mutation (errors) in copying
- The fitness of an organism is how much it can reproduce before it dies
- Evolution based on “survival of the fittest”

https://towardsdatascience.com/from-biology-to-computing-an-introduction-to-genetic-algorithms-b39476743483

https://towardsdatascience.com/an-introduction-to-genetic-algorithms-c07a81032547

Key element: representation

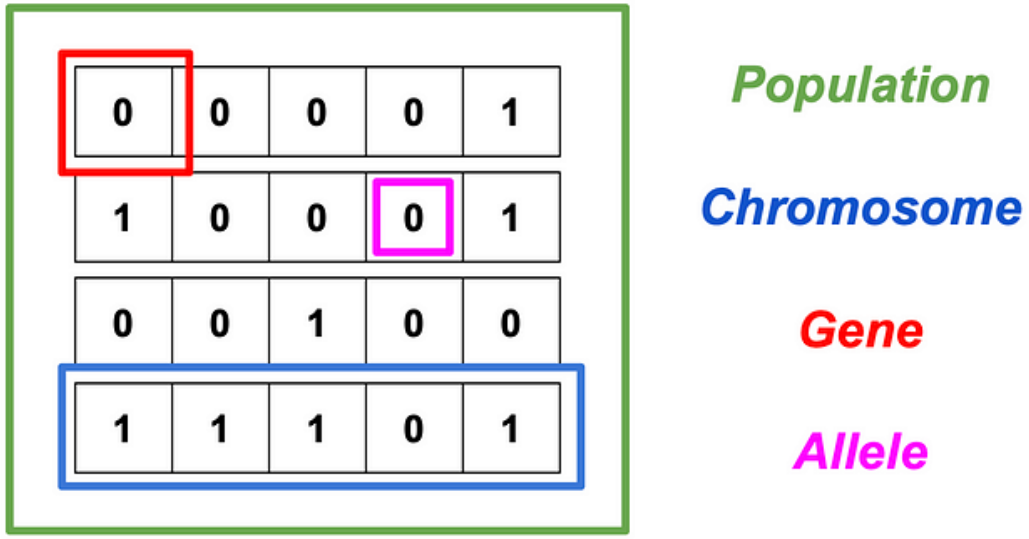


https://towardsdatascience.com/an-introduction-to-genetic-algorithms-c07a81032547

Key element: representation

Traveling Salesman Problem (TSP): Each chromosome is a sequence (array) of city numbers, representing a specific route. For example, [2, 5, 3, 1, 4] might represent a route that starts at city 2, then goes to 5, 3, 1, and ends at 4.

Optimizing Neural Network Architectures: Each chromosome could be a string or array representing the layers in the network, where each gene represents the type of layer (e.g., convolutional, pooling), its size, activation function, etc.



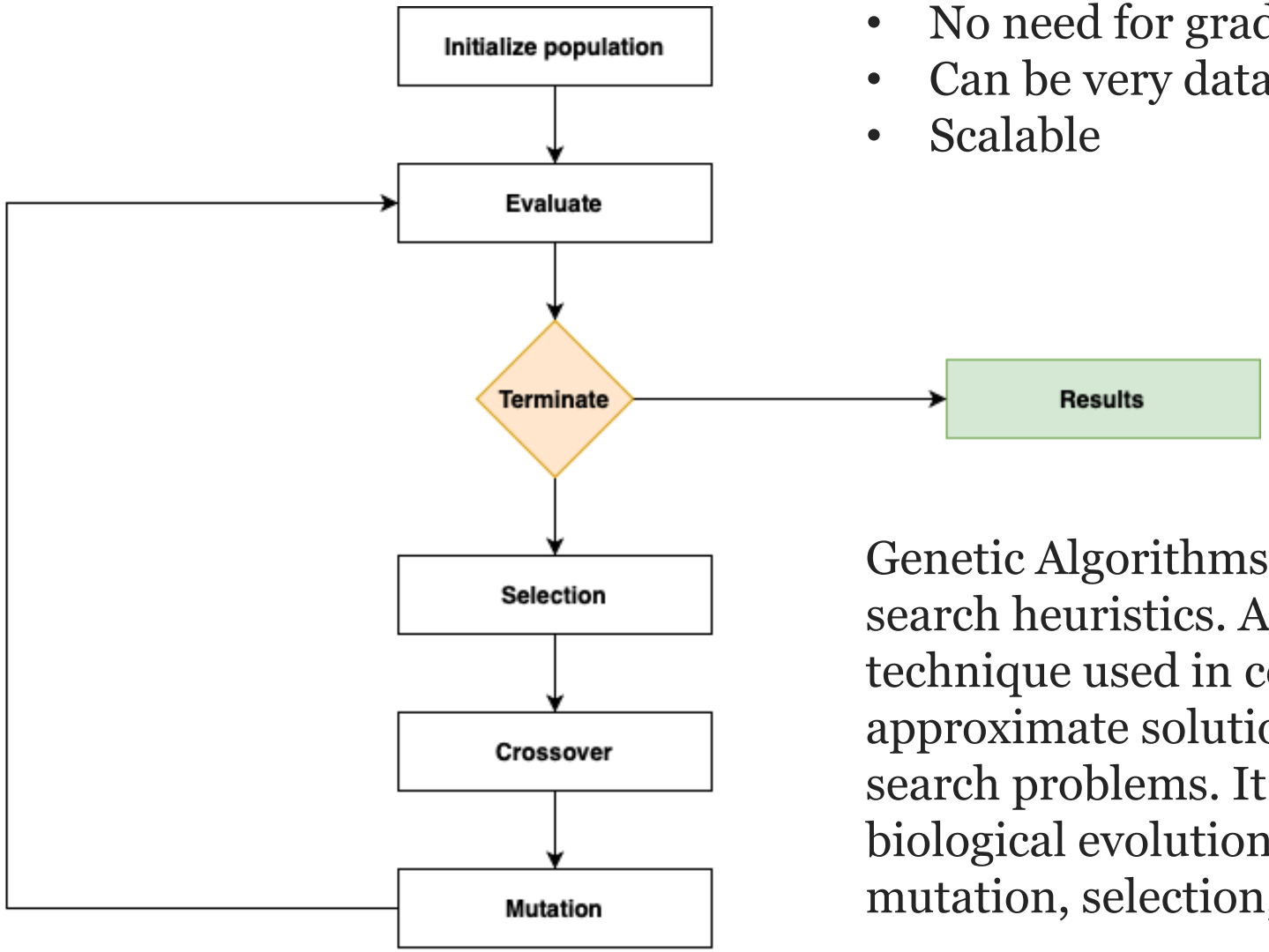
Portfolio Optimization: Each chromosome can be an array where each element represents the proportion of total funds allocated to a specific investment.

Genetic Programming: Chromosomes are tree structures representing computer programs. Nodes and branches of the tree represent operations and operands in the program.

Molecular design: representations are SMILES and SELFIES

https://towardsdatascience.com/an-introduction-to-genetic-algorithms-c07a81032547

General principle of GA



- No need for gradients
- Can be very data intensive
- Scalable

Genetic Algorithms are categorized as global search heuristics. A genetic algorithm is a search technique used in computing to find true or approximate solutions to optimization and search problems. It uses techniques inspired by biological evolution such as inheritance, mutation, selection, and crossover.

https://towardsdatascience.com/an-introduction-to-genetic-algorithms-c07a81032547

General principle of GA

Initialize population: genetic algorithms begin by initializing a **Population** of candidate solutions. This is typically done randomly to provide even coverage of the entire search space. A candidate solution is a C**hromosome** that is characterized by a set of parameters known as **Genes**.

Evaluate: next, the population is evaluated by assigning a fitness value to each individual in the population. In this stage we would often want to take note of the current fittest solution, and the average fitness of the population.

After evaluation, the algorithm decides whether it should terminate the search depending on the termination conditions set. Usually this will be because the algorithm has reached a fixed number of generations or an adequate solution has been found.

When the termination condition is finally met, the algorithm will break out of the loop and typically return its finial search results back to the user.

General principle of GA

- **Selection:** if the termination condition is not met, the population goes through a selection stage in which individuals from the population are selected based on their fitness score, the higher the fitness, the better chance an individual has of being selected.
- Two pairs of selected individuals called **parents.**
- **Crossover:** the next stage is to apply crossover and mutation to the selected individuals. This stage is where new individuals (**children**) are created for the next generation.
- **Mutation:** at this point the new population goes back to the evaluation step and the process starts again. We call each cycle of this loop a generation.

Selection methods

After we have our population, we need to select solutions (parents) to take forward to produce offspring for the next generation. The parents are selected based on their fitness to ensure the 'best' genes are passed on.

- **Elitism:** Choose a certain number of solutions in the current population with the best fitness function.

- **Roulette Wheel:** Sample from a probability distribution of the solutions (chromosomes) in the current population where each solution's probability, p_s, is proportional to its fitness score, f_s:

$$p_s = \frac{f_s}{\sum_{s=1}^{N} f_s}$$

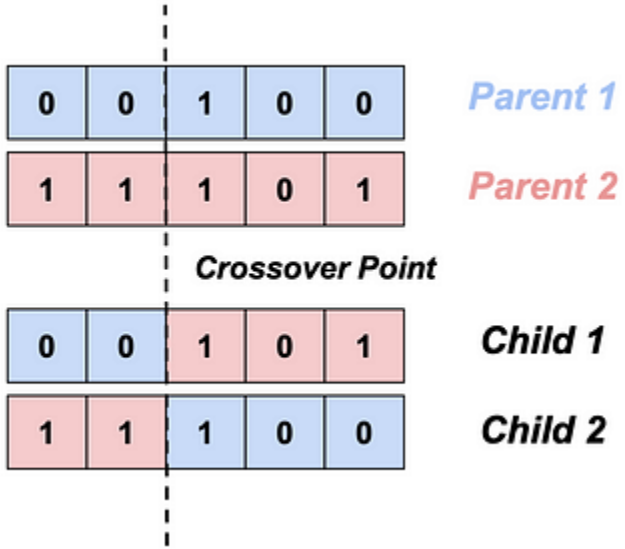
- **Tournament:** This involves selecting solutions at random and running tournaments with these subset solutions where the winner is the one with the best fitness score.

There are many other methods for the selection process and different variants of the techniques listed above. You can also combine selection procedures to generate hybrid methods as well. There is no 'one size fits all' and it's best to experiment with various types.

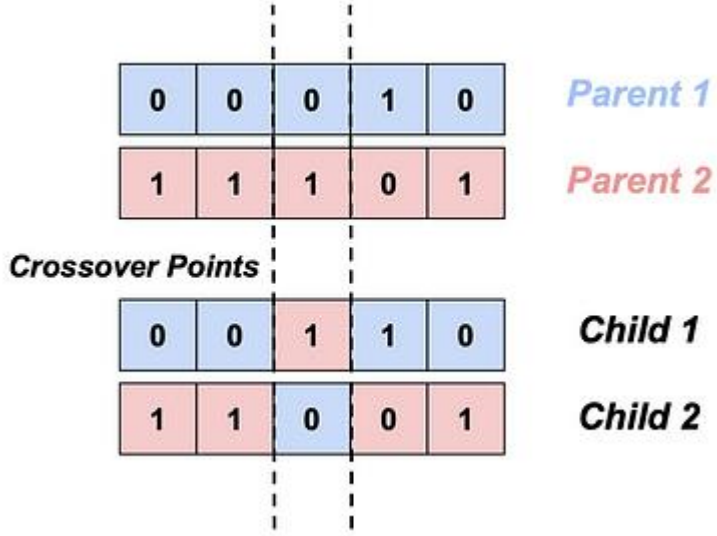
https://towardsdatascience.com/from-biology-to-computing-an-introduction-to-genetic-algorithms-b39476743483

Cross-over

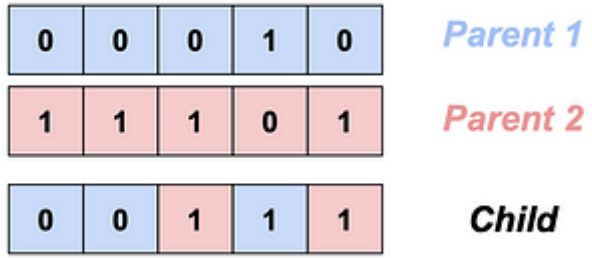
One-point crossover: *Swap the genes from a selected point on the parents' chromosomes:*



Two-point crossover: *Swap the genes from two selected points on the parents' chromosomes:*



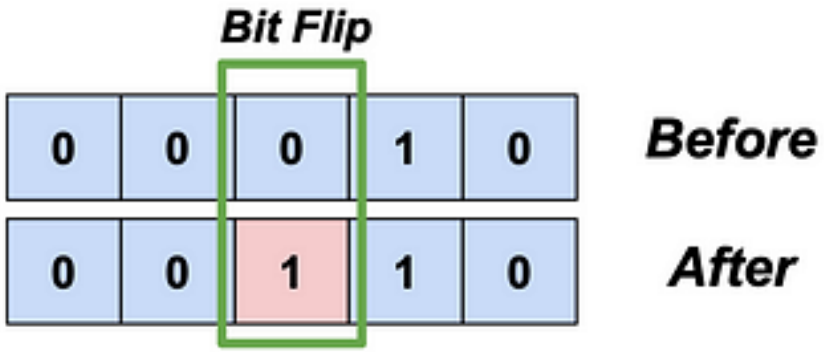
•**Uniform crossover:** *Each gene is randomly selected from the corresponding genes from the two parents:*



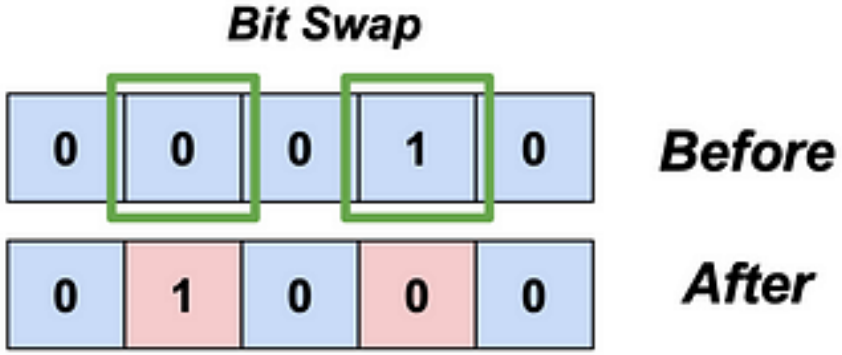
https://towardsdatascience.com/from-biology-to-computing-an-introduction-to-genetic-algorithms-b39476743483

Mutation

Bit flip: *Iterate through the genes in a chromosome and with each pass randomly flip a bit with a small probability:*



Swap: *Select two genes, with low probability, in the chromosome and swap them:*

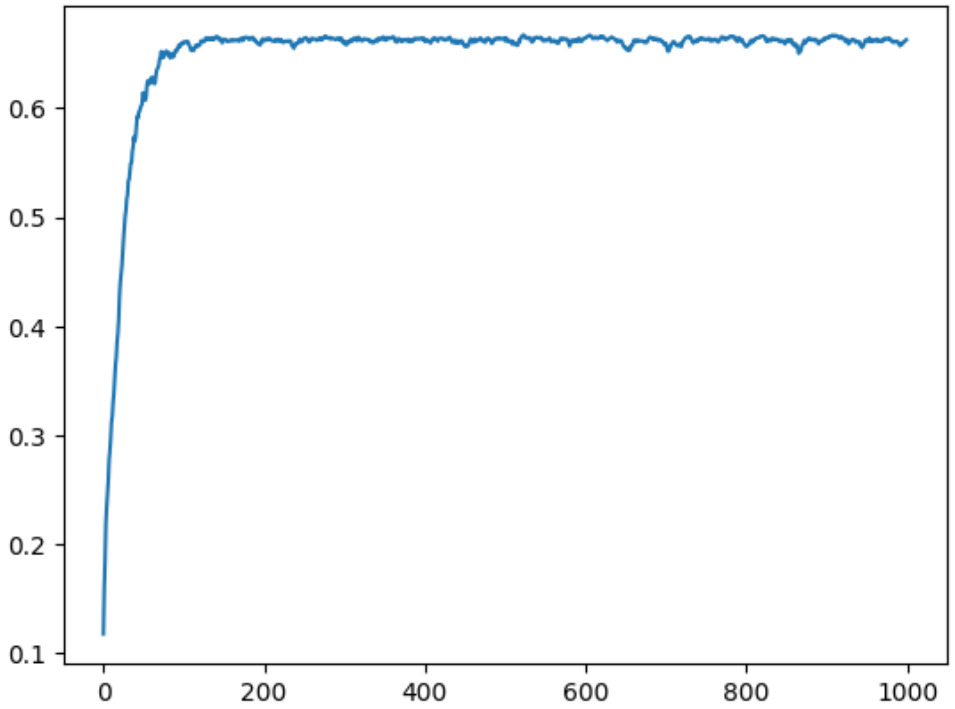


https://towardsdatascience.com/from-biology-to-computing-an-introduction-to-genetic-algorithms-b39476743483

Termination

- A certain number of generations reached
- A desired fitness is achieved
- Computational resource exhausted
- Fitness score has plateaued

```
RCFMcsMTDo,,ds
iJrjvRnbMPraOP
psQjeuocixTcfz
.msh,!QraH!KAu
pZ QzsjTdUVAPx
qfZgqrFG.IILhW
S,iQKcNiGAUJW!
tfFGWZa vqyUwM
YO.vNv.aCRnUEM
!huDlOebruGmys
```




```
HellgoloWrld
Helllo  Wold
Hell o  Wrd!
Hell olcWor!
Helloo oWol!
Hell o  Wrr!
Hell,o  Wol!
Helloo  aol!
Helloo  Wol!
Hell o oWol!
Hell,ol Wold
Hell o oWol!
Helllo YWod!
Helleo  Wold
```

https://towardsdatascience.com/from-biology-to-computing-an-introduction-to-genetic-algorithms-b39476743483

Overall:

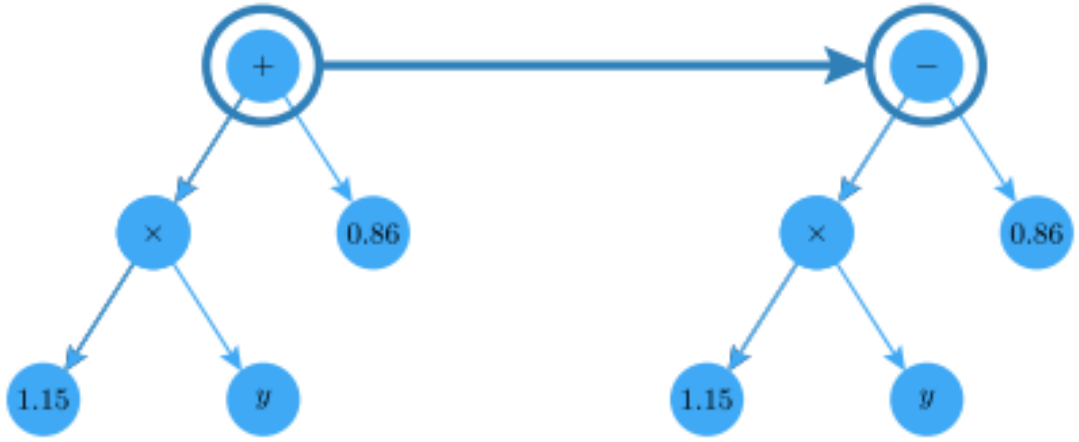
- Scalability is not as good as other optimization algorithms, hence long computing time
- Difficulty to adequately fine-tune hyperparameters such as mutation and selection can lead to non-convergence or useless results
- Doesn't guarantee finding the global optimum, however, this is the trade-off of all meta-heuristic methods
- An expensive and complex fitness function can also lead to long computation duration

Great method when we can represent solution as a string.

We can even tune neural networks using GA

https://towardsdatascience.com/genetic-algorithm-6aefd897f1ac

Functions can be represented as trees of operations!



$1.15y + 0.86$ $1.15y - 0.86$

https://arxiv.org/pdf/2305.01582.pdf