



HYPER PARAMETER TUNING

Using genetic algorithm



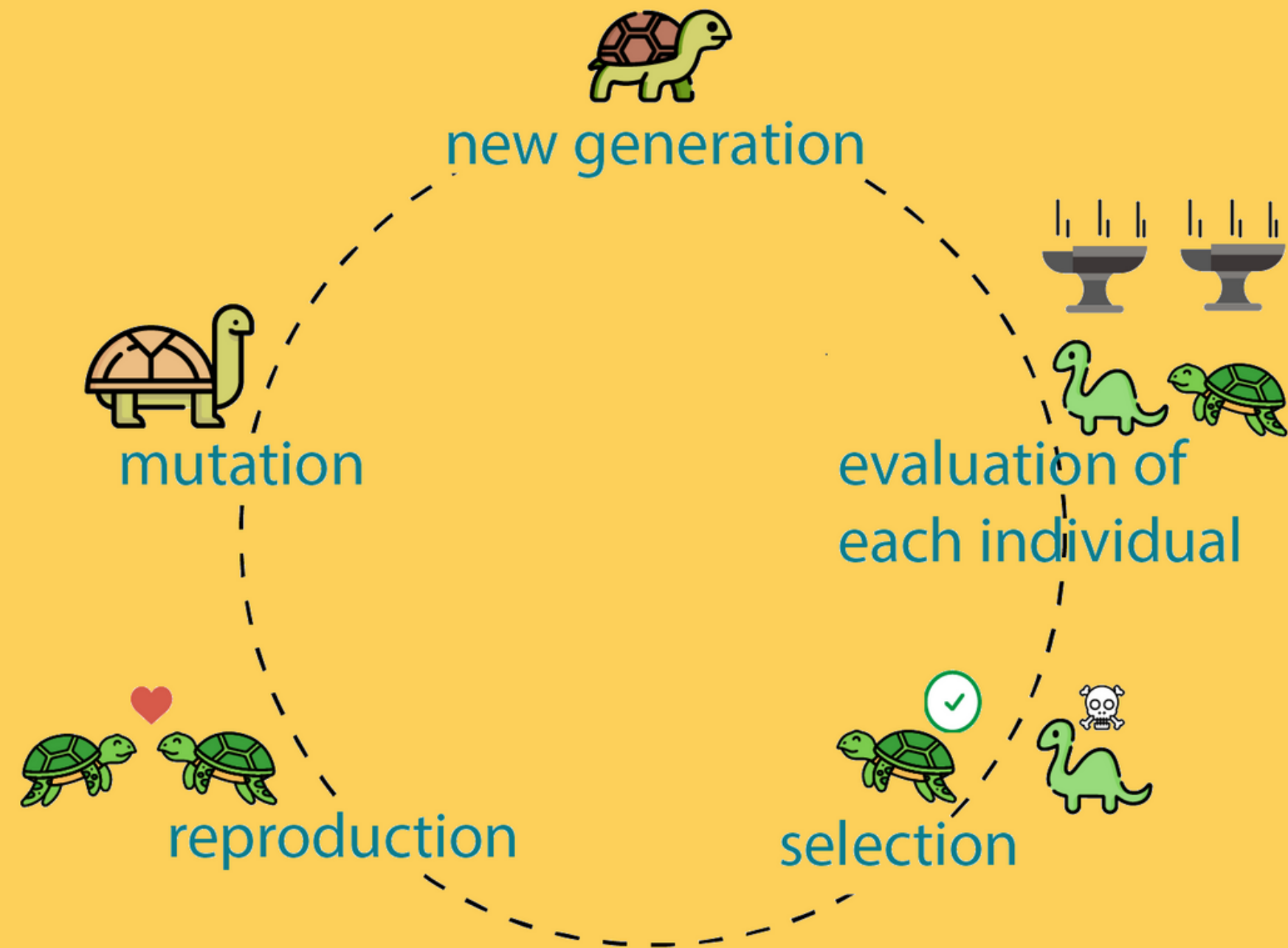
SUMMARY

- Genetic algorithms
- Hyper-parameters tuning
- Limitations (+ solutions)
- Implementation
- Some results

GENETIC ALGORITHM

GLOSSARY

Genes
Genomes
Population
Generations
Fitness



GENETIC OPERATIONS

FULL MUTATION

Creates a fully random new genome

PARTIAL MUTATION

Copies a genome from the population and changes randomly one of his genes

CROSSOVER

Performs a uniform crossover between 2 parents:
picks random genes from parent 1 and 2

HYPER- PARAMETERS OPTIMIZATION

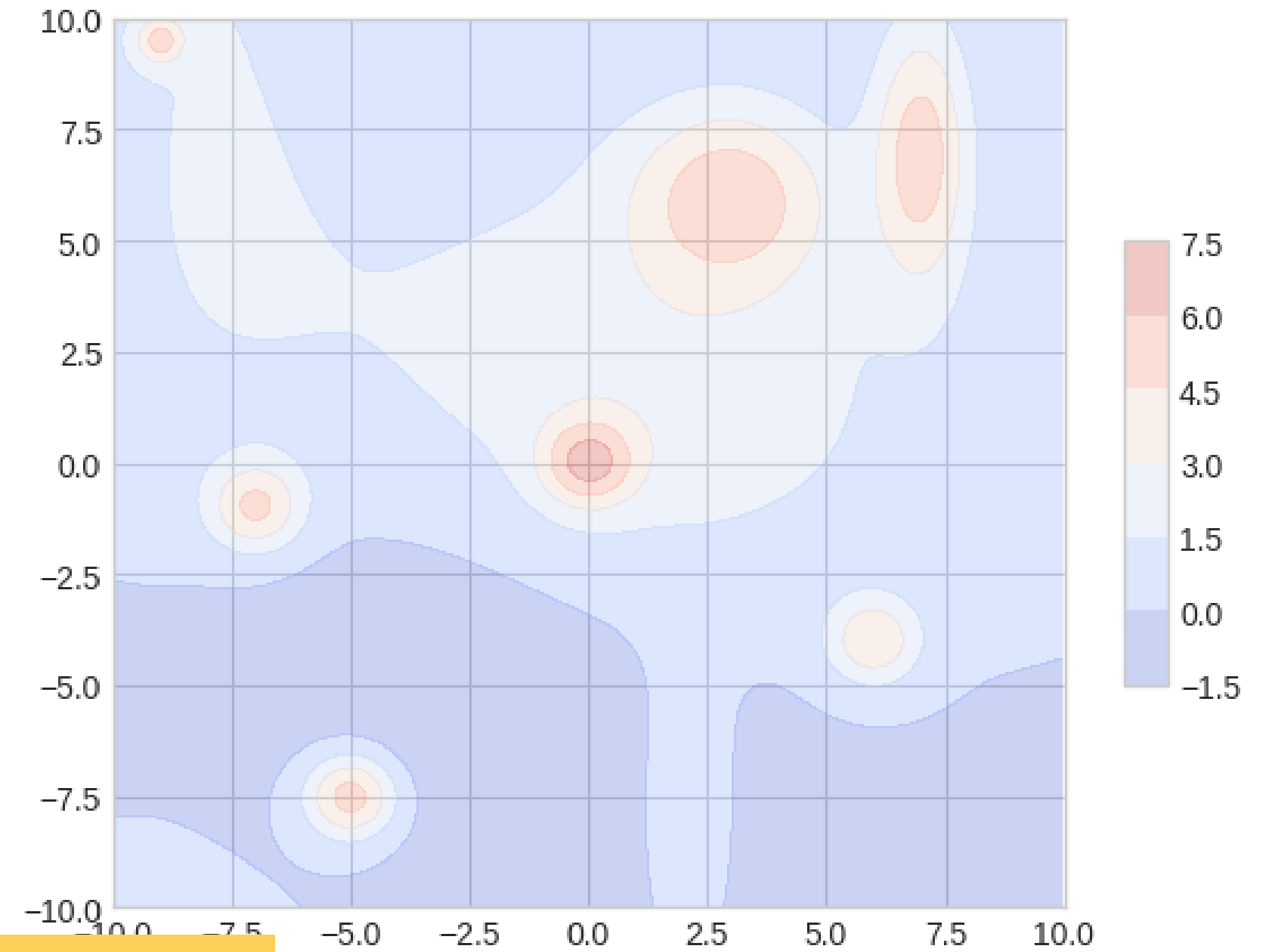
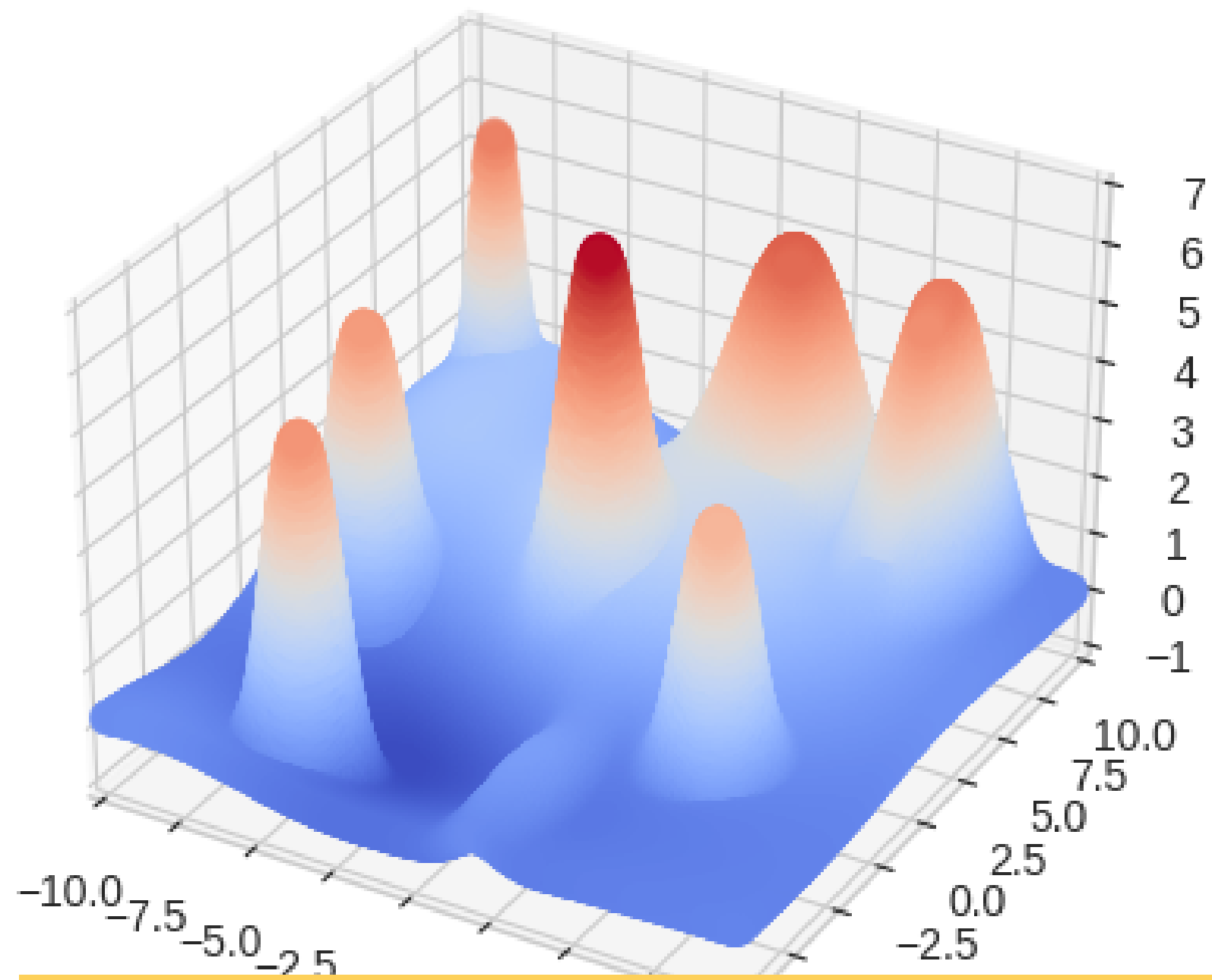


GENE = HYPER-PARAMETER

GENOME = HYPER-PARAMETERS
CONFIGURATION

FITNESS FUNCTION = TRAIN AND EVALUATE
THE MODEL WITH THE HYPER-PARAMETERS
CONFIGURATION

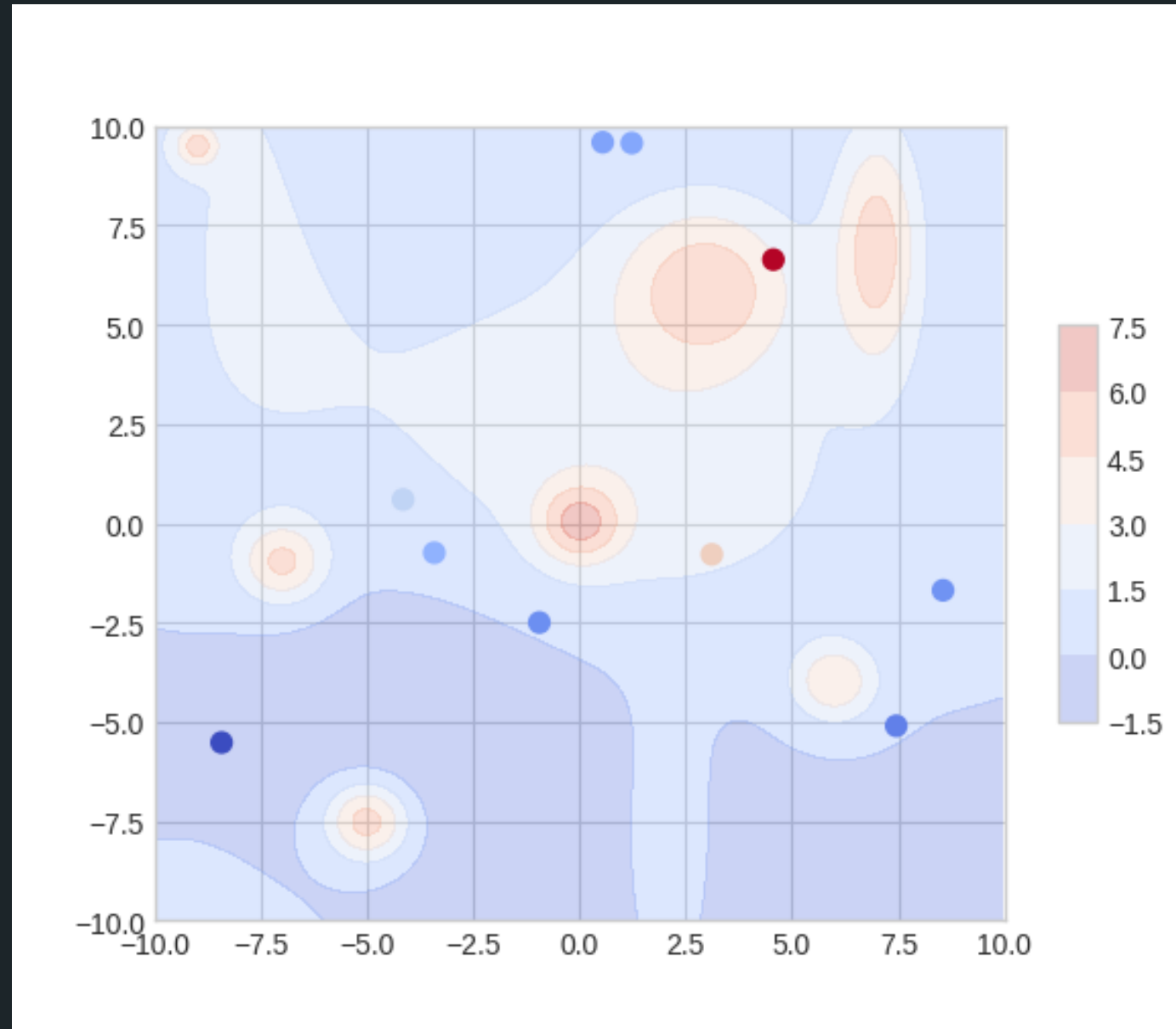
RUN THE GA TO FIND THE BEST-FITTING
CONFIGURATION



DEMO VISUALIZATIONS

LIMITATION

INITIAL SPACE EXPLORATION

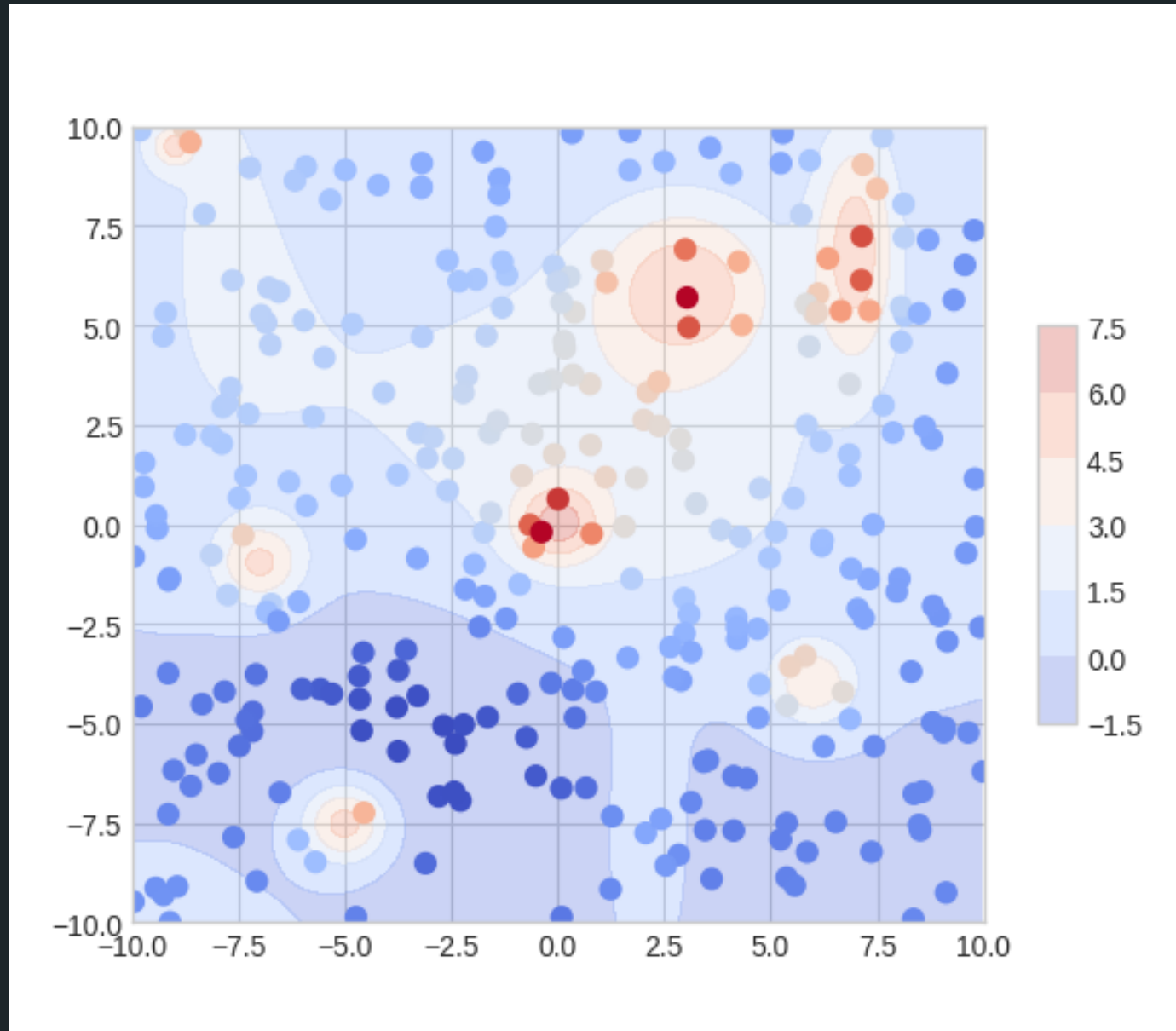


unsufficiently explored space

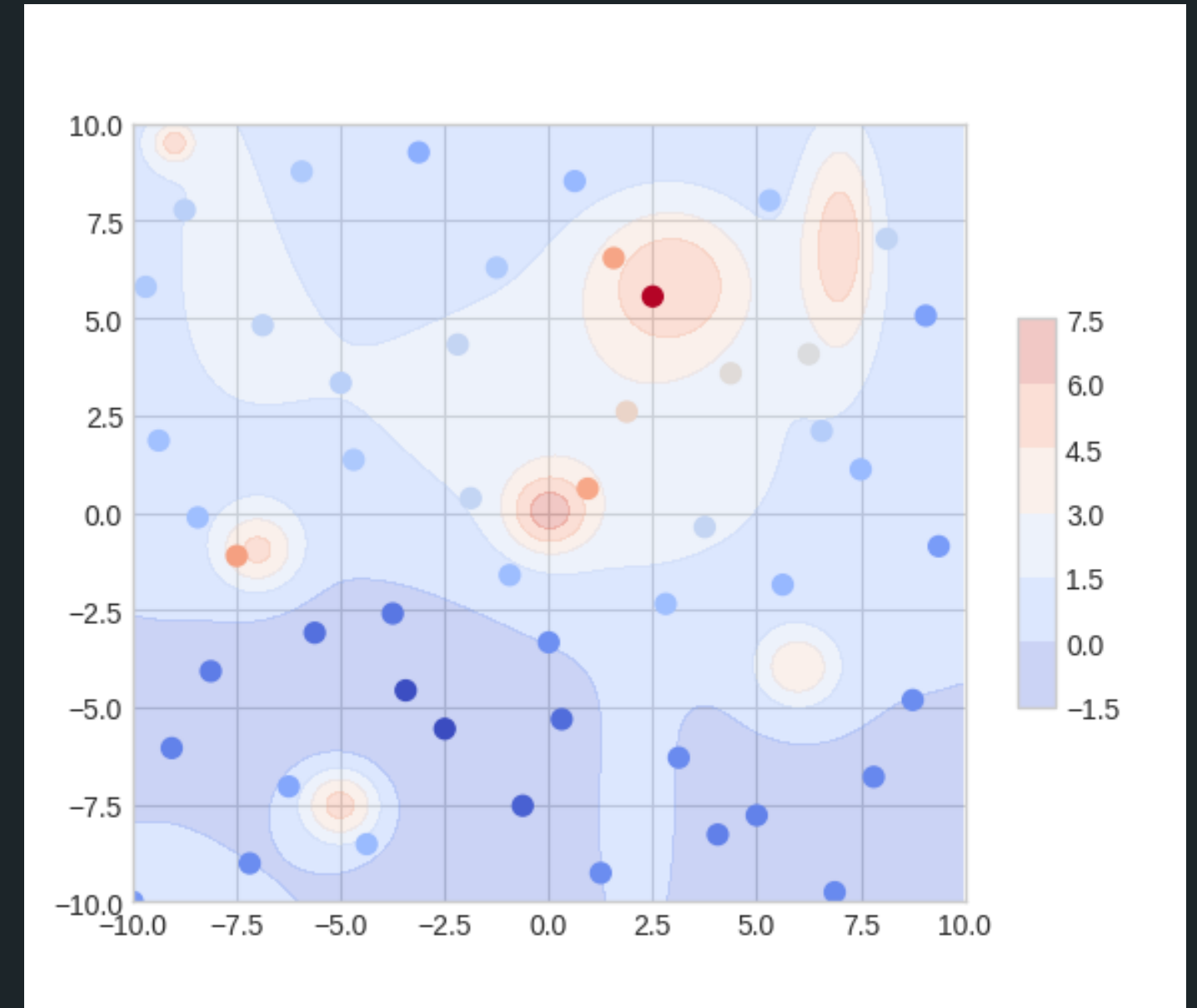
POSSIBLE CAUSES

- not enough genomes
- too large bounds
- too many dimensions

INITIAL SPACE EXPLORATION



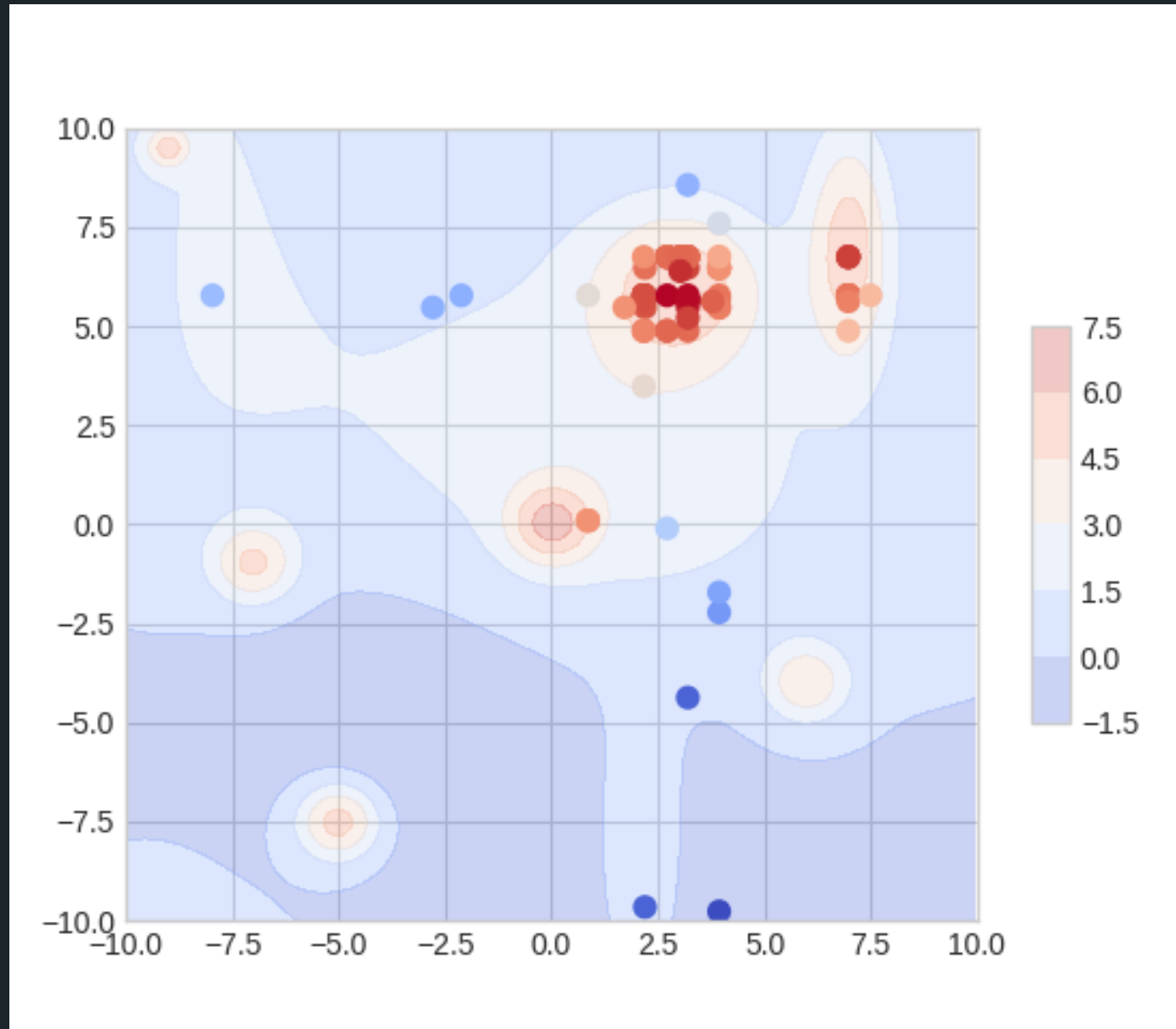
well explored space: numerous genomes



well explored space: low discrepancy sequence
(Halton sequence here)

LIMITATION

CONVERGENCE TOWARDS A LOCAL OPTIMUM



genomes converging towards a local optimum

POSSIBLE CAUSES

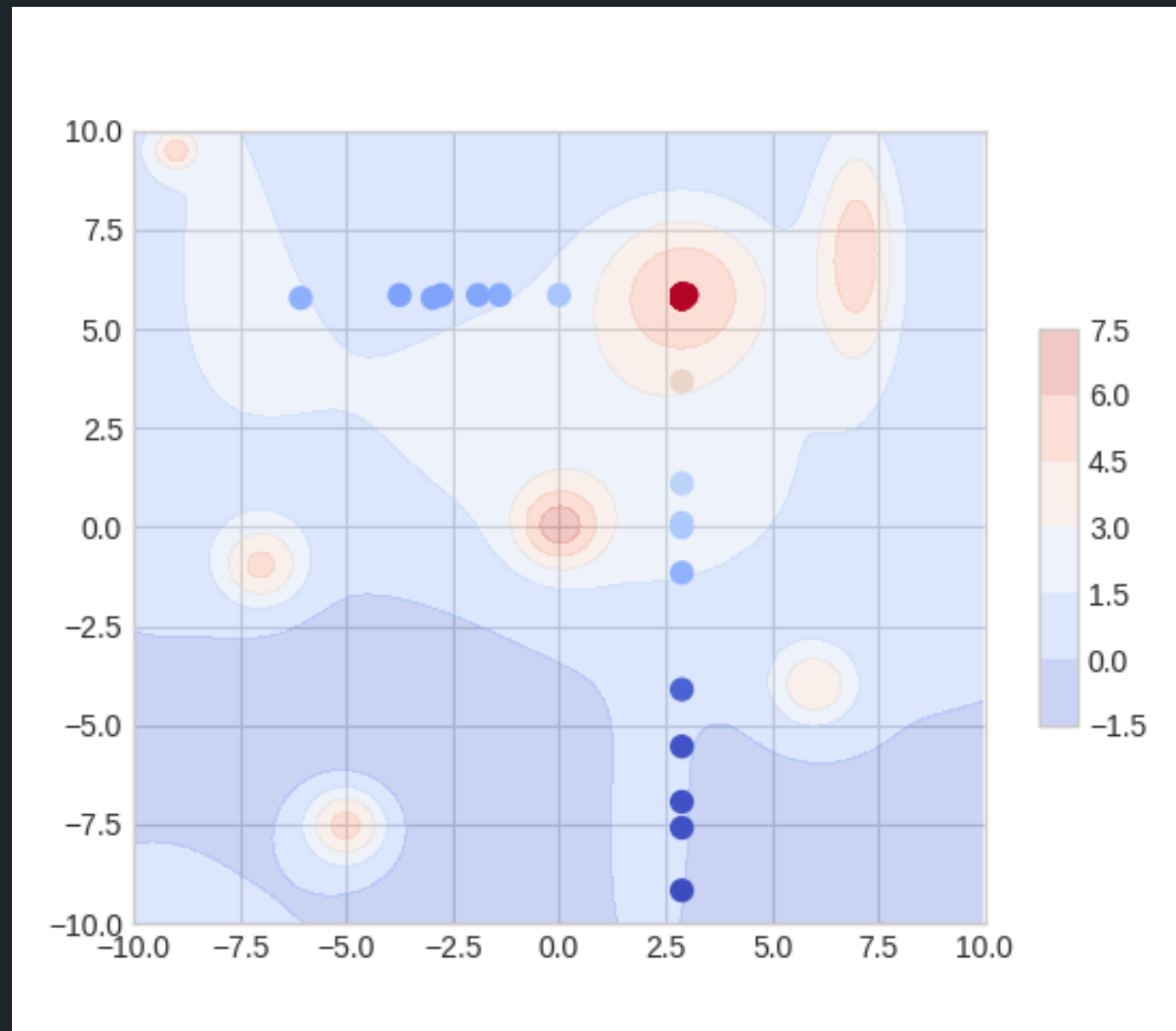
- not enough genomes
- not enough mutation

POSSIBLE SOLUTIONS

- same as previous
- add mutation (full and partial)
- penalize values too close to each other

LIMITATION

NO CLEAR STOP CRITERION MOST OF THE TIME

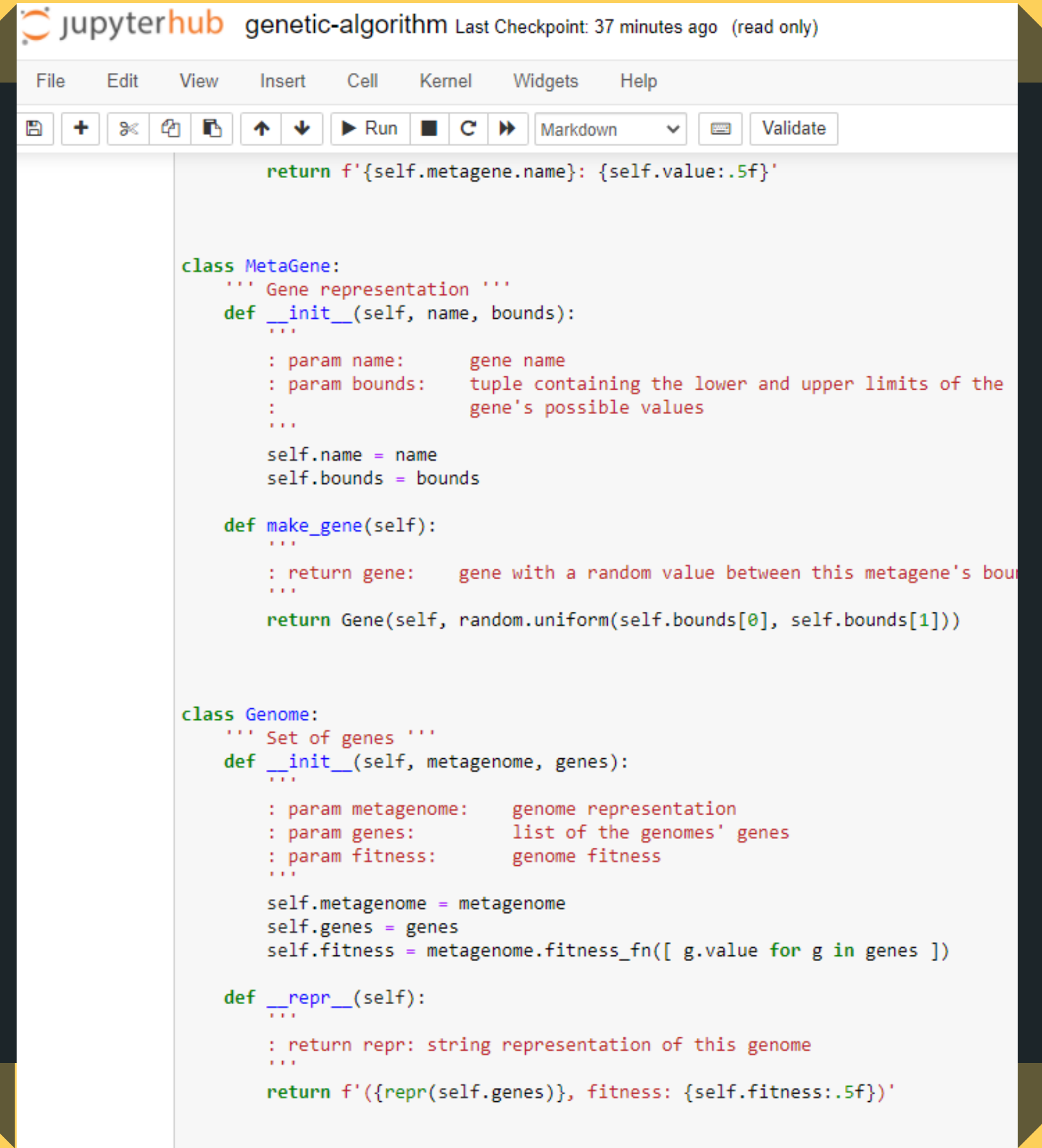


global optimum not reached

POSSIBLE SOLUTIONS

- variance of the best genomes
- not always applicable, and never certain

JUPYTER NOTEBOOK



```
return f'{self.metagene.name}: {self.value:.5f}'

class MetaGene:
    """ Gene representation """
    def __init__(self, name, bounds):
        """
        : param name:      gene name
        : param bounds:    tuple containing the lower and upper limits of the
        :                  gene's possible values
        """
        self.name = name
        self.bounds = bounds

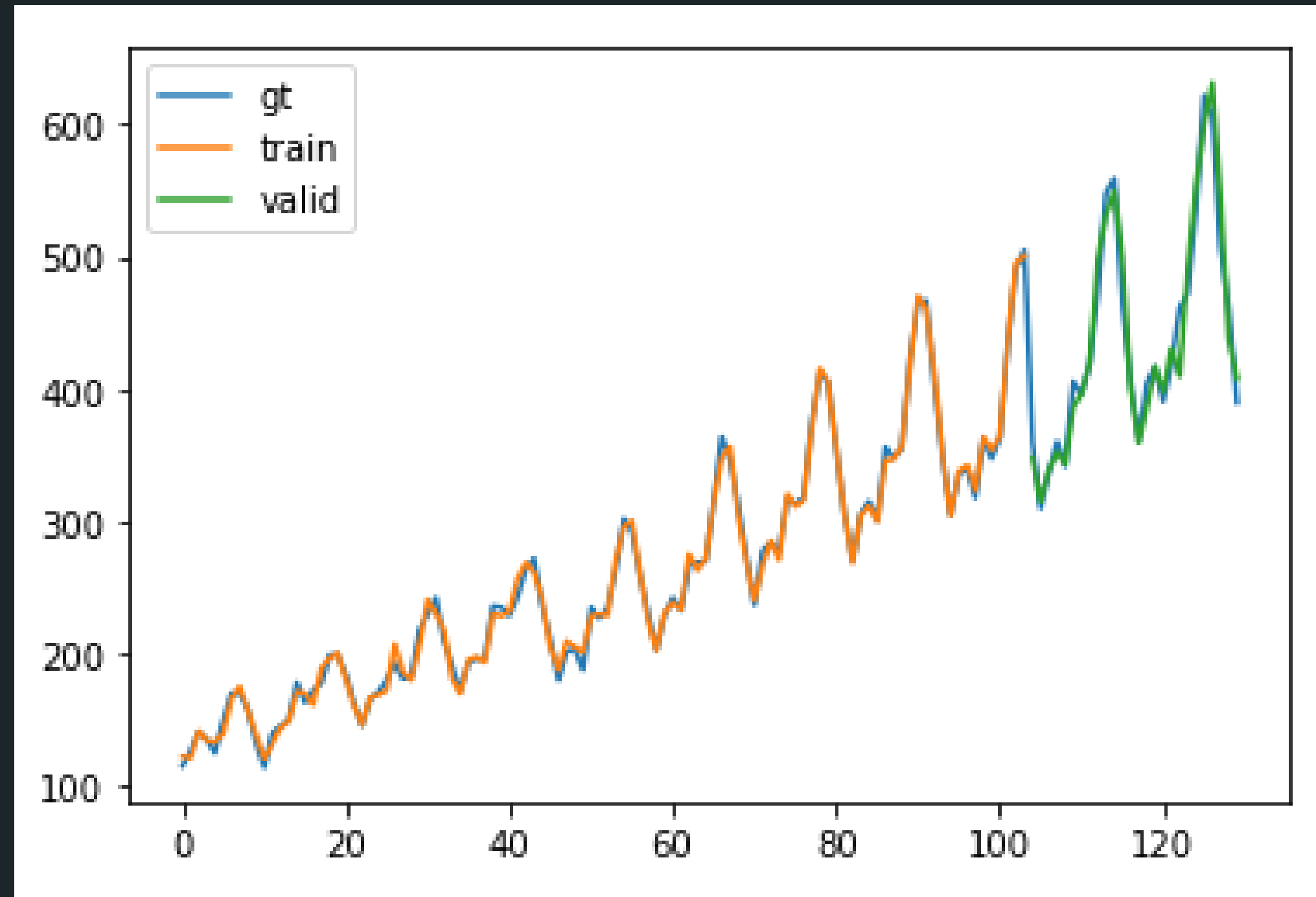
    def make_gene(self):
        """
        : return gene:     gene with a random value between this metagene's bound
        """
        return Gene(self, random.uniform(self.bounds[0], self.bounds[1]))

class Genome:
    """ Set of genes """
    def __init__(self, metagenome, genes):
        """
        : param metagenome: genome representation
        : param genes:      list of the genomes' genes
        : param fitness:    genome fitness
        """
        self.metagenome = metagenome
        self.genes = genes
        self.fitness = metagenome.fitness_fn([ g.value for g in genes ])

    def __repr__(self):
        """
        : return repr: string representation of this genome
        """
        return f'({repr(self.genes)}, fitness: {self.fitness:.5f})'
```

RESULTS

LECTURE EXAMPLE - L6.3 RNN



OUR RESULTS

Batch size	32
Hidden layers size	100
Loss function	MSE
Number of hidden layers	1
Random state	?
RNN architecture	LSTM
Window size	12
Best R2 score	0.9399

RESULTS WITHOUT
HYPER PAREMETER
TUNING

Batch size	8
Hidden layers size	80
Loss function	Huber
Number of hidden layers	2
Random state	15
RNN architecture	GRU
Window size	12
Best R2 score	0.9636
Generations	40

RESULTS WITH
HYPER PAREMETER
TUNING



REFERENCES

LOW-DISCREPANCY SEQUENCES (HALTON SEQUENCE)

<http://extremelearning.com.au/unreasonable-effectiveness-of-quasirandom-sequences/>

VARIANCE AS A STOPPING CRITERION FOR GENETIC ALGORITHMS

<https://www.isical.ac.in/~sankar/paper/Bhandari-2012.pdf>

QUESTIONS ?

THANKS FOR YOUR ATTENTION

GITHUB REPOSITORY

<https://github.com/thuas-imp-2021/Learning-Lab>

CONTACT

Adrien: A.Lucbert@student.hhs.nl

Michael: M.T.Weij@student.hhs.nl