AutoML: Hyperparameter Optimization

Evolutionary Algorithms

Bernd Bischl   Frank Hutter   Lars Kotthoff
Marius Lindauer   Joaquin Vanschoren

Evolutionary algorithms

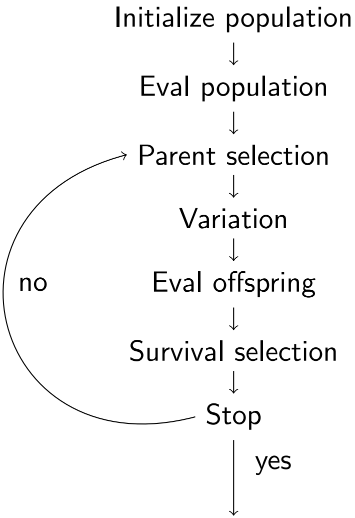
Evolutionary algorithms (EA) are a class of stochastic, metaheuristic optimization techniques whose mode of operation is inspired by the evolution of natural organisms.

History of evolutionary algorithms:

- **Genetic algorithms**: Use binary problem representation, therefore closest to the biological model of evolution.
- **Evolution strategies**: Use direct problem representation, e.g., vector of real numbers.
- **Genetic programming**: Create structures that convert an input into a fixed output (e.g. computer programs); solution candidates are represented as trees.
- **Evolutionary programming**: Similar to GP, but solution candidates are not represented by trees, but by finite state machines.

The boundaries between the terms become increasingly blurred and are often used synonymously.

Structure of an evolutionary algorithm



Notation and Terminology

Symbols	EA Terminology
Solution candidate $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$	Chromosome of an individual
$\boldsymbol{\lambda}_i$	i-th gene of chromosome
Set of candidates $\mathcal{P}$ with $\mu = \lvert\mathcal{P}\rvert$	Population and size
λ	Number of generated offsprings
$c : \boldsymbol{\Lambda} \rightarrow \mathbb{R}$	Fitness function

$$c(\boldsymbol{\lambda}) = \widehat{GE}_{\mathcal{D}_{\text{test}}}\left(\mathcal{I}(\mathcal{D}_{\text{train}}, \boldsymbol{\lambda})\right)$$

Notation clash:

- In EAs the objective function is often denoted $f(x)$.
- As these symbols are used for ML already we use $c(\boldsymbol{\lambda})$ and $\boldsymbol{\lambda}$ instead of f and x.
- Be careful: The offspring size λ is different from the candidate $\boldsymbol{\lambda}$ (bold symbol!).

Step 1: Initialize population

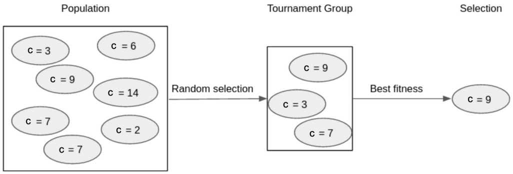
- A evolutionary algorithm is started by generating a initial population $\mathcal{P} = \{\boldsymbol{\lambda}^{(1)}, ..., \boldsymbol{\lambda}^{(\mu)}\}$.
- Usually we sample this uniformly at random.
- We could introduce problem prior knowledge via a smarter init procedure.
- This population is evaluated, i.e., the objective function is computed for every individual in the initial population.
- The initialization can have a large influence on the quality of the found solution, so many EAs employ *restarts* with new randomly generated populations.

Step 2: Parent selection I

In the first step of an iteration, λ parents are chosen, who create offspring in the next step.

Possibilities for selection of parents:

- **Neutral selection:** choose individual with a probability $1/\mu$.
- **Fitness-proportional selection:** draw individuals with probability proportional to their fitness.
- **Tournament Selection:** randomly select k individuals for a "Tournament Group". Of the drawn individuals, the best one (with the highest fitness value) is then chosen. Procedure is performed λ-times.



Step 3: Variation

New individuals are now generated from the parent population. This is done by

- Recombination/Crossover: combine two parents into one offspring.
- Mutation: (locally) change an individual.

Sometimes only one operation is performed.

Recombination for numeric representations

Two individuals $\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}} \in \mathbb{R}^n$ in numerical representation can be recombined as follows:

- **Uniform crossover**: choose gene i with probability p of 1st parent and probability $1-p$ of 2nd parent.
- **Intermediate recombination**: new individual is created from the mean value of two parents $\frac{1}{2}(\boldsymbol{\lambda} + \tilde{\boldsymbol{\lambda}})$
- **Simulated Binary Crossover (SBX)**: generate **two offspring** based on

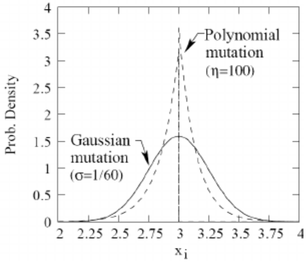
$$\bar{\boldsymbol{\lambda}} \pm \frac{1}{2}\beta(\tilde{\boldsymbol{\lambda}} - \boldsymbol{\lambda})$$

  with $\bar{\boldsymbol{\lambda}} = \frac{1}{2}(\boldsymbol{\lambda} + \tilde{\boldsymbol{\lambda}})$ and β randomly sampled from a certain distribution.

Mutation for numeric representations [K. Deb and D. Deb. 2014]

Mutation: individuals are changed, for example for $\boldsymbol{\lambda} \in \mathbb{R}^n$

- **Uniform mutation:** choose a random gene $\boldsymbol{\lambda}_i$ and replace it with a value uniformly distributed (within the feasible range).
- **Gauss mutation**: $\tilde{\boldsymbol{\lambda}} = \boldsymbol{\lambda} \pm \sigma \mathcal{N}(0, \boldsymbol{I})$
- **Polynomial mutation:** polynomial distribution instead of normal distribution



Recombination for bit strings

Two individuals $\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}} \in \{0,1\}^n$ encoded as bit strings can be recombined as follows:

- **1-point crossover**: select crossover $k \in \{1, \ldots, n-1\}$ randomly and select the first k bits from 1st parent, the last $n-k$ bits from 2nd parent.

$$\begin{array}{cccc}
1 & 1 & & 1 \\
0 & 0 & & 0 \\
\hline
0 & 1 & \Rightarrow & 1 \\
1 & 1 & & 1 \\
1 & 0 & & 0
\end{array}$$

- **Uniform crossover**: select bit i with probability p of 1st parent and $1-p$ of 2nd parent.

$$\begin{array}{cccc}
1 & 0 & & 1 \\
0 & 0 & & 0 \\
0 & 1 & \Rightarrow & 1 \\
0 & 1 & & 1 \\
1 & 0 & & 1
\end{array}$$

Mutation for bit strings

An individual $\boldsymbol{\lambda} \in \{0,1\}^n$ encoded as a bit string can be mutated as follows:

- **Bitflip**: for each index $k \in \{1, ..., n\}$: bit k is flipped with probability $p \in (0, 1)$.

$$\begin{matrix} 1 & & {\color{red}0} \\ 0 & & 0 \\ 0 & \Rightarrow & 0 \\ 0 & & {\color{red}1} \\ 1 & & 1 \end{matrix}$$

Step 4: Survival selection

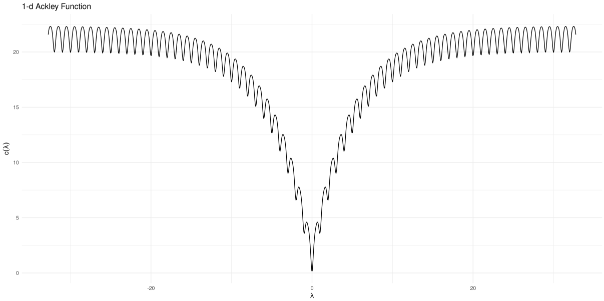
Now individuals are chosen who survive. Two common strategies are:

- (μ, λ)-**selection**: we select from the λ descendants the μ best ($\lambda \geq \mu$ necessary). **But:** best overall individual can get lost!
- $(\mu + \lambda)$-**selection**: μ parents and λ offspring are lumped together and the μ best individuals are chosen. Best individual safely survives.

Example of an evolutionary algorithm I

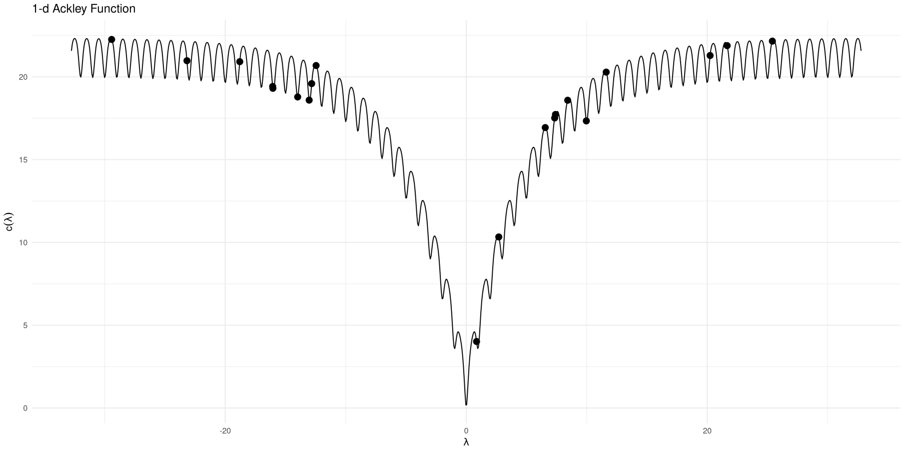
In the following, a (simple) EA is shown on the 1-dim Ackley function, optimized on $[-30, 30]$.

Usually for the optimization of a function $c : \mathbb{R}^n \rightarrow \mathbb{R}$ individuals are coded as real vectors $\boldsymbol{\lambda} \in \mathbb{R}^n$, so here we use simply one real number as chromosome.



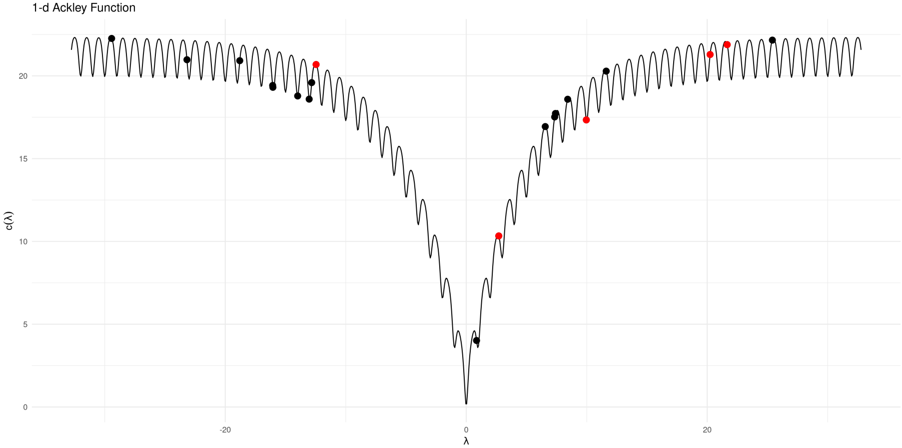
Example of an evolutionary algorithm II

Randomly init population with size $\mu = 20$.



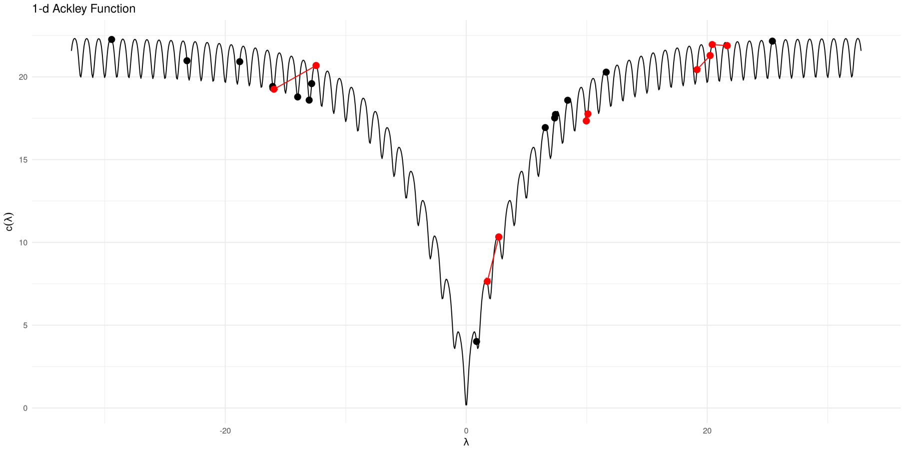
Example of an evolutionary algorithm III

We choose $\lambda = 5$ offspring by neutral selection (red individuals).



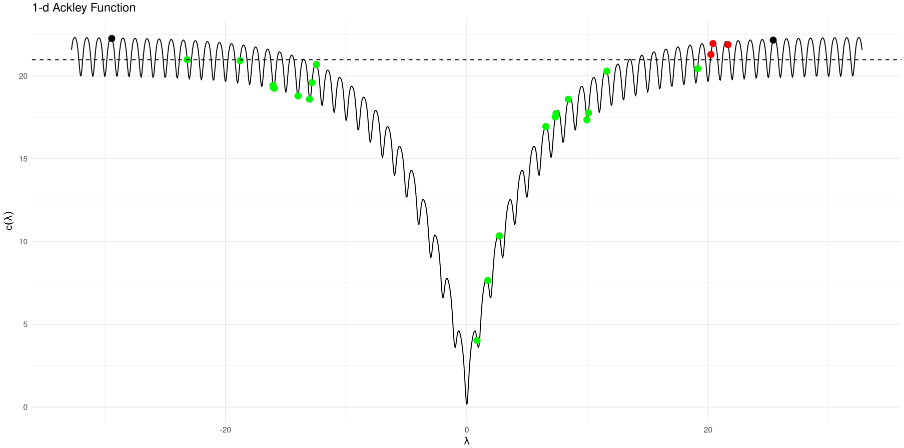
Example of an evolutionary algorithm IV

We use a Gauss mutation with $\sigma = 2$ and do not apply a recombination.



Example of an evolutionary algorithm V

We use a $(\mu + \lambda)$ selection. The selected individuals are green.



Evolutionary Algorithms

Advantages

- Conceptually simple, yet powerful enough to solve complex problems (including HPO)
- All parameter types possible in general
- Highly parallelizable (depends on μ)
- Allows customization via specific variation operators

Disadvantages

- Less theory available (for realistic, complex EAs)
- Can be hard to get balance between exploration and exploitation right
- Can have quite a few control parameters, hard to set them correctly
- Customization necessary for complex problems
- Not perfectly suited for expensive problems like HPO, as quite a higher number of evaluations is usually needed for appropriate convergence / progress