

Pattern Recognition XI

Genetic Algorithms

Michal Haindl

Faculty of Information Technology, KTI
Czech Technical University in Prague
Institute of Information Theory and Automation
Czech Academy of Sciences
Prague, Czechia



<http://ro.utia.cz/>



MI/MIE-ROZ 2023-2024

© Haindl

MI-ROZ - 11

1/32

Outline

- 1 Genetic Algorithms
 - Fitness Function
 - Genetic Operators
- 2 GA Requirements
 - Genetic Programming
 - Genetic Clustering
- 3 Genetic Segmentation
- 4 Genetic Search
- 5 Genetic Feature Selection



© Haindl

MI-ROZ - 11

2/32

Genetic Algorithms

optimization techniques based on natural evolution mechanism,
working in binary search spaces

C.R. Darwin (1859 - On the Origin of Species)

Survival of the fittest

- 1 Variation must be copying because selection does not create directly anything, but presupposes a large population to work on.
- 2 Variation must be small-scaled in practice. Species do not appear suddenly.
- 3 Variation is undirected (the blind watchmaker paradigm).

G. Mendel (1865 - Versuche über Pflanzen-Hybriden)



© Haindl

MI-ROZ - 11

3/32

Genetic Algorithms

optimization techniques based on natural evolution mechanism,
working in binary search spaces

C.R. Darwin (1859 - On the Origin of Species)

G. Mendel (1865 - Versuche über Pflanzen-Hybriden)



*12.2.1809 +19.4.1882



*22.7.1822 +6.1.1884

- computer simulation of biological systems (1950s)
- evolutionary strategies - Recheuberg (1973)
- GA University of Michigan - J. Holland (1975)



© Haindl

MI-ROZ - 11

3/32

Genetic Algorithms

optimization techniques based on natural evolution mechanism,
working in binary search spaces

C.R. Darwin (1859 - On the Origin of Species)

G. Mendel (1865 - Versuche über Pflanzen-Hybriden)

differences from other optimization techniques:

- GA uses a coding (a finite length string **chromosome** over a finite alphabet, e.g., binary strings) of the parameter set, not the parameters themselves, the design of the representation is an important part of the GA
- GA search from a large **population of possible solutions**, the search for the optimum is driven from many points in the search space → better chance to find the global optimum



Genetic Algorithms 2

differences

- GA uses objective function information, no derivatives or other auxiliary knowledge, there is no guarantee that this function is relevant to the solved problem, the value of the evaluation function is called **fitness** in GA
- GA uses probabilistic transition rules, they support good **strings** with higher fitness and remove poor strings with lower fitness, the best strings are allowed survive with a higher probability
- survival and death is achieved applying three basic operations **reproduction, crossover, mutation**



Genetic Algorithms 2

- combine directed and stochastic search
- $\binom{n + 2^d - 1}{2^d - 1}$ number of different populations (n chromosomes, d bits)
- ⊕ implicit parallelism
- ⊕ robust to dynamic changes
- ⊕ resistant to be trapped in local optima
- ⊕ wider solution space
- ⊕ easy modification for different problems
- ⊕ no restrictions for fitness function
- ⊕ discontinuities have little effect on overall optimization performance
- ⊕ no gradient knowledge required
- ⊕ good for multi-modal problems, large, poorly understood search spaces



Genetic Algorithms 2

- ⊖ do not guarantee the global optimum
- ⊖ cannot use gradients
- ⊖ problems with constraints or incorporating problem specific information
- ⊖ definition of representation for the problem
- ⊖ heuristic parameters (population size, initialization, op. probabilities, fitness function, number of bits, ...)
- ⊖ no effective terminator
- ⊖ require large number of fitness function evaluations
- ⊖ not effective for smooth unimodal functions
- ⊖ multidimensional high-precision problems
(e.g., 10^2 variables, 10^3 range, 6 digit precision $\Rightarrow$ solution $V = [v_1, \dots, v_{100}]$
vector length 20 $\rightarrow 20 \times 100 = 2000$, $2^{2000} \approx 10^{600}$ search space)



Genetic Algorithms - Encoding

binary 110100011010

octal 15723314

hexadecimal 5FC7DBA

value *back, left, right, forward, downward*

tree tree of some objects (functions and commands of a programming language)

other arrays, lists,...



Population

- The initial population generation:
 - random initialization,
 - using any prior information available,
 - some known good solutions,
 - uniform cover of the whole solution space.
- The population size:
 - large more efficient to reach global optimum,
 - requires much more computational cost, memory and time.



Fitness Function

$$f(\text{chromosome}_i)$$

chromosome - a feasible solution

<i>i</i> th generation	$f(\alpha_{j,i})$	rank
1100101101100111	20	4.
1110001110100010	35	2.
0001111001101101	17	5.
1010101111000100	22	3.
1110011000000010	37	1.
1001111001111100	13	6.



Fitness Function

$$f(\text{chromosome}_i)$$

chromosome - a feasible solution

<i>i</i> th generation	$f(\alpha_{j,i})$	rank
1100101101100111	20	4.
1110001110100010	35	2.
0001111001101101	17	5.
1010101111000100	22	3.
1110011000000010	37	1.
1001111001111100	13	6.



Fitness Function

$$\text{goodness_of_the_solution} = f(\text{chromosome}_i)$$

$f()$ depends on

- application
- selected genetic representation



Fitness Function

probabilistic fitness - Boltzmann (Gibbs) distribution

$$P(i) = \frac{\exp\{\frac{f_i}{T}\}}{E\left\{\exp\{\frac{f_i}{T}\}\right\}}$$

f_i i-th chromosome fitness

T ↘ control parameter (temperature)

$E\{.\}$ over the current generation



Genetic Operators - Reproduction

strings with the highest fitness are copied into the next generation strings

the selection process is probabilistic, the probability of copying is given by its relative fitness in the current population

some strings with high fitness can be copied more than once
the total cardinality of population set remains constant

$$\{0, 1, 1, 1, 0, 0, 0, 0\} \rightarrow \{0, 1, 1, 1, 0, 0, 0, 0\}$$



Crossover

- Exchange features of two individuals to produce two offspring.
- Selected mates may have good properties to survive in next generation.
 - best individuals based upon their fitness values
 - random selection
 - rank selection (select randomly 2 chromosomes a choose the better, repeat that step)
- Thus we expect that exchanging features may produce other good individuals.



Crossover

mating of new generation string to produce new strings by swapping all characters between the beginning of the string pairs and the randomly chosen border position

$$\begin{Bmatrix} \{0, 0, 0, 0, 0, 0, 0, 0\} \\ \{1, 1, 1, 1, 1, 1, 1, 1\} \end{Bmatrix} \rightarrow \begin{Bmatrix} \{0, 0, 0, 0, 0, 0, 0, 0\} \\ \{1, 1, 1, 1, 1, 1, 1, 1\} \end{Bmatrix}$$



Crossover

mating of new generation string to produce new strings by swapping all characters between the beginning of the string pairs and the randomly chosen border position

$$\begin{Bmatrix} \{0, 0, 0, 0, 0, 0, 0, 0\} \\ \{1, 1, 1, 1, 1, 1, 1, 1\} \end{Bmatrix} \rightarrow \begin{Bmatrix} \{1, 1, 1, 1, 1, 0, 0, 0\} \\ \{0, 0, 0, 0, 0, 1, 1, 1\} \end{Bmatrix}$$



Crossover

mating of new generation string to produce new strings by swapping all characters between the beginning of the string pairs and the randomly chosen border position

- all newly reproduced strings are subject to crossover
- probabilistic parameter represents the number of pairs processed by crossover $P_{crossover} = \begin{cases} 0, & \text{no crossover;} \\ 1, & \text{all chromosomes.} \end{cases}$
- implicit parallelism - substrings (schemata) can represent correct solution even if string as a whole is wrong, if the schemata can be manipulated independently, the optimal solution can be located faster than if all characters are handled independently



Crossover

Single-point Crossover

$$\begin{Bmatrix} \{0, 0, 0, 0, 0, 0, 0, 0\} \\ \{1, 1, 1, 1, 1, 1, 1, 1\} \end{Bmatrix} \rightarrow \begin{Bmatrix} \{1, 1, 1, 1, 1, 0, 0, 0\} \\ \{0, 0, 0, 0, 0, 1, 1, 1\} \end{Bmatrix}$$



Crossover

Single-point Crossover

$$\left\{ \begin{array}{l} \{0, 0, 0, 0, 0|0, 0, 0\} \\ \{1, 1, 1, 1, 1|1, 1, 1\} \end{array} \right\} \rightarrow \left\{ \begin{array}{l} \{1, 1, 1, 1, 1, 0, 0, 0\} \\ \{0, 0, 0, 0, 0, 1, 1, 1\} \end{array} \right\}$$

Two-point Crossover

$$\left\{ \begin{array}{l} \{0, 0|0, 0, 0|0, 0, 0\} \\ \{1, 1|1, 1, 1|1, 1, 1\} \end{array} \right\} \rightarrow \left\{ \begin{array}{l} \{1, 1, 0, 0, 0, 1, 1, 1\} \\ \{0, 0, 1, 1, 1, 0, 0, 0\} \end{array} \right\}$$



Crossover

Single-point Crossover

$$\left\{ \begin{array}{l} \{0, 0, 0, 0, 0|0, 0, 0\} \\ \{1, 1, 1, 1, 1|1, 1, 1\} \end{array} \right\} \rightarrow \left\{ \begin{array}{l} \{1, 1, 1, 1, 1, 0, 0, 0\} \\ \{0, 0, 0, 0, 0, 1, 1, 1\} \end{array} \right\}$$

Two-point Crossover

$$\left\{ \begin{array}{l} \{0, 0|0, 0, 0|0, 0, 0\} \\ \{1, 1|1, 1, 1|1, 1, 1\} \end{array} \right\} \rightarrow \left\{ \begin{array}{l} \{1, 1, 0, 0, 0, 1, 1, 1\} \\ \{0, 0, 1, 1, 1, 0, 0, 0\} \end{array} \right\}$$

Uniform Crossover

$$\left\{ \begin{array}{l} \{0, 1, 1, 0, 0, 1, 0, 1\} \\ \{0|0, 0|0, 0|0|0|0\} \\ \{1|1, 1|1, 1|1|1|1\} \end{array} \right\} \rightarrow \left\{ \begin{array}{l} \text{crossover template} \\ \{0, 1, 1, 0, 0, 1, 0, 1\} \\ \{1, 0, 0, 1, 1, 0, 1, 0\} \end{array} \right\}$$



Crossover

Single-point Crossover

$$\left\{ \begin{array}{l} \{0, 0, 0, 0, 0|0, 0, 0\} \\ \{1, 1, 1, 1, 1|1, 1, 1\} \end{array} \right\} \rightarrow \left\{ \begin{array}{l} \{1, 1, 1, 1, 1, 0, 0, 0\} \\ \{0, 0, 0, 0, 0, 1, 1, 1\} \end{array} \right\}$$

Two-point Crossover

$$\left\{ \begin{array}{l} \{0, 0|0, 0, 0|0, 0, 0\} \\ \{1, 1|1, 1, 1|1, 1, 1\} \end{array} \right\} \rightarrow \left\{ \begin{array}{l} \{1, 1, 0, 0, 0, 1, 1, 1\} \\ \{0, 0, 1, 1, 1, 0, 0, 0\} \end{array} \right\}$$

Uniform Crossover

$$\left\{ \begin{array}{l} \{0, 1, 1, 0, 0, 1, 0, 1\} \\ \{0|0, 0|0, 0|0|0|0\} \\ \{1|1, 1|1, 1|1|1|1\} \end{array} \right\} \rightarrow \left\{ \begin{array}{l} \text{crossover template} \\ \{0, 1, 1, 0, 0, 1, 0, 1\} \\ \{1, 0, 0, 1, 1, 0, 1, 0\} \end{array} \right\}$$

Multipoint Crossover (N-Point)



Crossover

Tree Parents Crossover

$$\left\{ \begin{array}{l} \{0, 1, 1, 0, 0, 1, 0, 0\} \\ \{1, 1, 1, 1, 1, 1, 1, 0\} \\ \{1, 0, 0, 1, 1, 0, 0, 0\} \end{array} \right\} \rightarrow \{1, 1, 1, 1, 1, 1, 0, 0\}$$

1 three parents p_1, p_2, p_3 are randomly chosen

2

$$p_{1,i} = p_{2,i} \Rightarrow o_i = p_{1,i}$$

$$p_{1,i} \neq p_{2,i} \Rightarrow o_i = p_{3,i}$$



Crossover

Crossover with Reduced Surrogate

constrains crossover to always produce new individuals (restricting the location of crossover points such that crossover points only occur where gene values differ)

$$\begin{Bmatrix} \{0, 1, 1, 0, 1, 0, 0, 0\} \\ \{1, 1, 1, 1, 1, 1, 1, 0\} \end{Bmatrix} \rightarrow \begin{Bmatrix} \{1, 1, 1, 1, 1, 1, 1, 0\} \\ \{0, 1, 1, 0, 1, 0, 0, 0\} \end{Bmatrix}$$



Mutation

randomly changes one character of some string in population from time to time

idea - recovering lost configurations (in crossover / mutation)

$$\{0, 1, 1, 1, 0, 0, 0, 0\} \rightarrow \{0, 1, 1, 1, 0, 0, 1, 0\}$$

flipping:

$$\begin{Bmatrix} \{0, 1, 1, 0, 0, 1, 0, 0\} \\ \{1, 0, 0, 1, 1, 0, 0, 0\} \end{Bmatrix} \rightarrow \begin{Bmatrix} \{1, 1, 1, 1, 1, 1, 0, 0\} \end{Bmatrix}$$

interchanging:

$$\{0, 1, 1, 1, 0, 0, 0, 0\} \rightarrow \{0, 1, 0, 1, 0, 0, 1, 0\}$$



Mutation

$$P_{mutation} \rightarrow 0 \quad \text{should happen with low probability}$$

$$(P_{mutation} = \frac{1}{chromosome_length})$$

- ⊖ $P_{crossover}, P_{mutation} \nearrow 1$ inefficient random search
- ⊖ $P_{crossover}, P_{mutation} \searrow 0$ too long search



Other Operators

inversion (chromosome reversal - seldom useful)

$$\{0, 1, 1, 1, 0, 0, 0, 0\} \rightarrow \{0, 0, 0, 0, 1, 1, 1, 0\}$$

substring inversion

$$\{1, 1, 0, 1, 1, 0, 1, 0\} \rightarrow \{1, 1, 0, 1, 1, 1, 0, 1\}$$

fusion, fission, ...



Other Operators

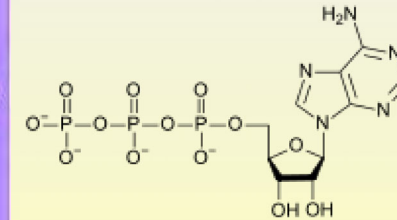
stop criterion

- no a significant (average) fitness improvement through several generations
- maximum generations
- elapsed time
- best individual
- sum / median fitness



Molecular Clock

human DNA mutation speed - $0.5 \cdot 10^{-9}$ mutations per nucleotide/year $\Rightarrow$



AMP - adenosin monophosphate (part of RNA)



Molecular Clock

human DNA mutation speed - $0.5 \cdot 10^{-9}$ mutations per nucleotide/year $\Rightarrow$

≈ 5000 generations - African Eve $100 \cdot 10^3 - 150 \cdot 10^3$ years
non-African ancestor $\approx 60 \cdot 10^3 - 90 \cdot 10^3$ years



Molecular Clock

human DNA mutation speed - $0.5 \cdot 10^{-9}$ mutations per nucleotide/year $\Rightarrow$

≈ 5000 generations - African Eve $100 \cdot 10^3 - 150 \cdot 10^3$ years
non-African ancestor $\approx 60 \cdot 10^3 - 90 \cdot 10^3$ years

whole human genetic variability < single chimp group (15 - 150 individuals)



Genetic Algorithm Requirements

Criteria for GA approach:

Completeness Any solution should have its encoding

Non redundancy Codes and solutions should correspond one to one

Soundness Any code (produced by genetic operators) should have its corresponding solution

Characteristic perseverance Offspring should inherit useful characteristics from parents



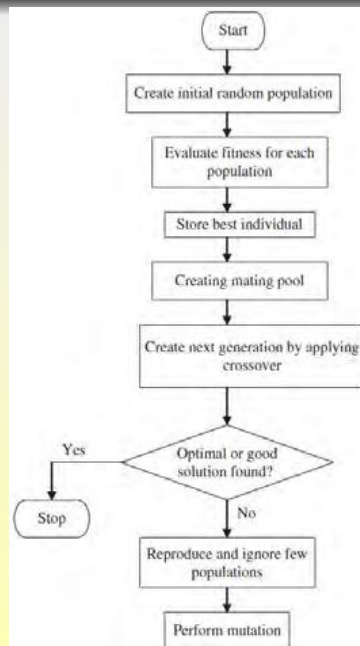
Genetic Algorithm Requirements

GA requirements:

- A genetic representation for potential solutions of the problem.
- Creation of an initial population of potential solutions.
- An evaluation function (environment) rating solutions ("fitness").
- Genetic operators.
- Parameter values (population size, genetic operators probabilities, etc.).



Genetic Algorithm



Genetic Algorithm

- 1 Create a starting population and find the value of their objective functions.
- 2 Reproduction - probabilistically reproduce high fitness strings in the new population, remove low fitness strings.
- 3 Crossover - construct new strings combining reproduced code strings from the previous population.
- 4 Mutation - from time to time change one character of some string at random.
- 5 Rank code strings of the current population according to the value of their objective function (fitness).
- 6 If the maximum achieved string fitness does not increase over several steps stop, the desired optimum is represented by the current string of maximum fitness. Otherwise, repeat from step 2.



Genetic Algorithm

prior knowledge ↓

- optimization function design (heuristics)
- starting population selection



Genetic Programming

automatic programming



Genetic Programming

automatic programming

code snippets → chromosomes

operators:

reproduction a snippet unchanged

crossover mixing two snippets

mutation each element of a snippet can be changed
(compatibly to the snippet syntax!)

insertion replacement of a single snippet element with another
snippet randomly chosen

- ⊕ simple operations (multiplication, division,...)
- ⊖ difficult for syntactically rich languages (C++)



Genetic Clustering

chromosome {1, 2, 4, 4, 5, 5, 5}

Fusion operator (*eliminates redundancy*):

{1, 2, 4, 4, 5, 5, 5} → {4, 2, 4, 4, 5, 5, 5}

Fission operator (*eliminates misclassification*):

{1, 2, 4, 4, 5, 5, 5} → {1, 2, 3, 4, 5, 5, 5}



Genetic Clustering

chromosome $\{1, 2, 4, 4, 5, 5, 5\}$

Fusion operator (*eliminates redundancy*):

$$\{1, 2, 4, 4, 5, 5, 5\} \rightarrow \{4, 2, 4, 4, 5, 5, 5\}$$

Fission operator (*eliminates misclassification*):

$$\{1, 2, 4, 4, 5, 5, 5\} \rightarrow \{1, 2, 3, 4, 5, 5, 5\}$$



Genetic Clustering - Example

Old generation

$$\left\{ \begin{array}{l} \{1, 2, 4, 4, 5, 5, 5\} \\ \{1, 2, 3, 3, 4, 4, 4\} \\ \{1, 2, 2, 3, 4, 5, 5\} \\ \{1, 1, 2, 3, 4, 4, 5\} \\ \{1, 2, 2, 2, 4, 4, 5\} \end{array} \right\} \rightarrow \left\{ \begin{array}{l} \text{crossover} \\ \text{crossover} \\ \text{mutation} \\ \text{fusion} \\ \text{fission} \end{array} \right.$$



Genetic Clustering - Example

New generation

$$\left\{ \begin{array}{l} \{1, 2, 4, 4, 5, 5, 5\} \\ \{1, 2, 3, 3, 4, 4, 4\} \\ \{1, 2, 2, 3, 4, 5, 5\} \\ \{1, 1, 2, 3, 4, 4, 5\} \\ \{1, 2, 2, 2, 4, 4, 5\} \end{array} \right\} \rightarrow \left\{ \begin{array}{ll} \{1, 2, 4, 4, 4, 4, 4\} & \text{crossover} \\ \{1, 2, 3, 3, 5, 5, 5\} & \text{crossover} \\ \{1, 2, 2, 3, 4, 4, 5\} & \text{mutation} \\ \{1, 1, 3, 3, 4, 4, 5\} & \text{fusion} \\ \{1, 2, 2, 3, 4, 4, 5\} & \text{fission} \end{array} \right.$$



Genetic Clustering

K fixed, solution coding:

$$[\vec{\mu}_1, \dots, \vec{\mu}_K] = [\mu_{11}, \dots, \mu_{1I}, \dots, \mu_{K1}, \dots, \mu_{KI}]$$

cost function:

$$J = \sum_{i=1}^N u_{ij} \delta(X_i, \vec{\mu}_j)$$

$$u_{ij} = \begin{cases} 1 & \delta(X_i, \vec{\mu}_j) = \min_{k=1, \dots, K} \delta(X_i, \vec{\mu}_k) \\ 0 & \text{otherwise} \end{cases}$$



Genetic Clustering 2

reproduction more probable solutions are replicated

crossover allowable cuts between different representatives ($\vec{\mu}_j$) only

mutation randomly selects a solution vector coordinate and randomly decides to add a small random number



Genetic Clustering 2

alternative method

- 1 $n \times$ a hard clustering method with different initialisation
→ population
- 2 apply reproduce operator

(hopefully chooses between n local minima of the cost function)



Genetic Segmentation

- a confidence (does not need to fulfil the total probability theorem) in the interpretation ω_i of the object i given its properties X_i

$$c(\omega_i|X_i) = p(\omega_i|X_i)$$

- a confidence in the interpretation ω_i of the object i according to the interpretations ω_j of its neighbours

$$c(\omega_i) = \frac{c(\omega_i|X_i) \sum_{j=1}^{n_i} r(\omega_i, \omega_j) c(\omega_j|X_j)}{n_i}$$

where $r(\omega_i, \omega_j)$ represents the value of a compatibility function of two adjacent objects i, j with labels ω_i, ω_j , n_i is the number of adjacent objects to i

- a whole image interpretation confidence

$$c_{\text{image}} = \frac{\sum_{j=1}^n c(\omega_j)}{n}$$



Genetic Segmentation

- a confidence (does not need to fulfil the total probability theorem) in the interpretation ω_i of the object i given its properties X_i

$$c(\omega_i|X_i) = p(\omega_i|X_i)$$

- a confidence in the interpretation ω_i of the object i according to the interpretations ω_j of its neighbours

$$c(\omega_i) = \frac{c(\omega_i|X_i) \sum_{j=1}^{n_i} r(\omega_i, \omega_j) c(\omega_j|X_j)}{n_i}$$

where $r(\omega_i, \omega_j)$ represents the value of a compatibility function of two adjacent objects i, j with labels ω_i, ω_j , n_i is the number of adjacent objects to i

- a whole image interpretation confidence

$$c_{\text{image}} = \frac{\sum_{j=1}^n c(\omega_j)}{n}$$



Genetic Segmentation

- a confidence (does not need to fulfil the total probability theorem) in the interpretation ω_i of the object i given its properties X_i

$$c(\omega_i|X_i) = p(\omega_i|X_i)$$

- a confidence in the interpretation ω_i of the object i according to the interpretations ω_j of its neighbours

$$c(\omega_i) = \frac{c(\omega_i|X_i) \sum_{j=1}^{n_i} r(\omega_i, \omega_j) c(\omega_j|X_j)}{n_i}$$

where $r(\omega_i, \omega_j)$ represents the value of a compatibility function of two adjacent objects i, j with labels ω_i, ω_j , n_i is the number of adjacent objects to i

- a whole image interpretation confidence

$$C_{image} = \frac{\sum_{j=1}^n c(\omega_j)}{n}$$



Genetic Segmentation 2

- 1 Initialize segmentation into primary regions, define a correspondence between each region and the related position of its label in the code string generated by a GA.
- 2 Construct a primary region adjacency graph.
- 3 Randomly pick the starting population, possibly using a priori information.
- 4 Collapse a region adjacency graph for each code string of the current population. Using the current region adjacency graphs, compute the value of the optimization segmentation function for each code string.
- 5 If the maximum of the optimization criterion does not increase significantly in several steps, stop.
- 6 Let the GA generate a new population of segmentation and interpretation hypothesis. Go to 4.
- 7 The code string with the maximum confidence represents the best segmentation hypothesis.



Genetic Segmentation 2

- 1 Initialize segmentation into primary regions, define a correspondence between each region and the related position of its label in the code string generated by a GA.
- 2 Construct a primary region adjacency graph.
- 3 Randomly pick the starting population, possibly using a priori information.
- 4 Collapse a region adjacency graph for each code string of the current population. Using the current region adjacency graphs, compute the value of the optimization segmentation function for each code string.
- 5 If the maximum of the optimization criterion does not increase significantly in several steps, stop.
- 6 Let the GA generate a new population of segmentation and interpretation hypothesis. Go to 4.
- 7 The code string with the maximum confidence represents the best segmentation hypothesis.



Genetic Segmentation 2

- 1 Initialize segmentation into primary regions, define a correspondence between each region and the related position of its label in the code string generated by a GA.
- 2 Construct a primary region adjacency graph.
- 3 Randomly pick the starting population, possibly using a priori information.
- 4 Collapse a region adjacency graph for each code string of the current population. Using the current region adjacency graphs, compute the value of the optimization segmentation function for each code string.
- 5 If the maximum of the optimization criterion does not increase significantly in several steps, stop.
- 6 Let the GA generate a new population of segmentation and interpretation hypothesis. Go to 4.
- 7 The code string with the maximum confidence represents the best segmentation hypothesis.



Genetic Segmentation 2

- 1 Initialize segmentation into primary regions, define a correspondence between each region and the related position of its label in the code string generated by a GA.
- 2 Construct a primary region adjacency graph.
- 3 Randomly pick the starting population, possibly using a priori information.
- 4 Collapse a region adjacency graph for each code string of the current population. Using the current region adjacency graphs, compute the value of the optimization segmentation function for each code string.
- 5 If the maximum of the optimization criterion does not increase significantly in several steps, stop.
- 6 Let the GA generate a new population of segmentation and interpretation hypothesis. Go to 4.
- 7 The code string with the maximum confidence represents the best segmentation hypothesis.



Genetic Segmentation 2

- 1 Initialize segmentation into primary regions, define a correspondence between each region and the related position of its label in the code string generated by a GA.
- 2 Construct a primary region adjacency graph.
- 3 Randomly pick the starting population, possibly using a priori information.
- 4 Collapse a region adjacency graph for each code string of the current population. Using the current region adjacency graphs, compute the value of the optimization segmentation function for each code string.
- 5 If the maximum of the optimization criterion does not increase significantly in several steps, stop.
- 6 Let the GA generate a new population of segmentation and interpretation hypothesis. Go to 4.
- 7 The code string with the maximum confidence represents the best segmentation hypothesis.



Genetic Segmentation 2

- 1 Initialize segmentation into primary regions, define a correspondence between each region and the related position of its label in the code string generated by a GA.
- 2 Construct a primary region adjacency graph.
- 3 Randomly pick the starting population, possibly using a priori information.
- 4 Collapse a region adjacency graph for each code string of the current population. Using the current region adjacency graphs, compute the value of the optimization segmentation function for each code string.
- 5 If the maximum of the optimization criterion does not increase significantly in several steps, stop.
- 6 Let the GA generate a new population of segmentation and interpretation hypothesis. Go to 4.
- 7 The code string with the maximum confidence represents the best segmentation hypothesis.



Genetic Segmentation 2

- 1 Initialize segmentation into primary regions, define a correspondence between each region and the related position of its label in the code string generated by a GA.
- 2 Construct a primary region adjacency graph.
- 3 Randomly pick the starting population, possibly using a priori information.
- 4 Collapse a region adjacency graph for each code string of the current population. Using the current region adjacency graphs, compute the value of the optimization segmentation function for each code string.
- 5 If the maximum of the optimization criterion does not increase significantly in several steps, stop.
- 6 Let the GA generate a new population of segmentation and interpretation hypothesis. Go to 4.
- 7 The code string with the maximum confidence represents the best segmentation hypothesis.



Segmentation Example

ball on a lawn

prior information

- ball is circular
- ball is inside a green lawn

$$c(\omega_i = B|X_i) = compactness(i)$$

$$c(\omega_i = L|X_i) = greenness(i)$$

 $c(B | circular) = 1$ confidence in a perfect ball

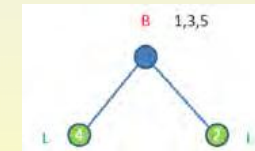
 $c(L | green) = 1$ confidence in a perfect lawn

 $r(B \text{ inside } L) = 1$ all other combinations $r() = 0$


UTIA

Segmentation Example

ball on a lawn

1) starting hypothesis ($BLBLB^{(0)}$, $LLLBL^{(0)}$)

UTIA

Segmentation Example

ball on a lawn

2) random crossover

 $BL|BLB^{(0)}$ $c_{image} = 0.00$
 $LL|LBL^{(0)}$ $c_{image} = 0.12$

↓

 $LLBLB^{(1)}$ $c_{image} = 0.20$
 $BLLBL^{(x)}$ $c_{image} = 0.00$


UTIA

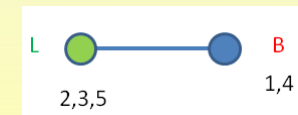
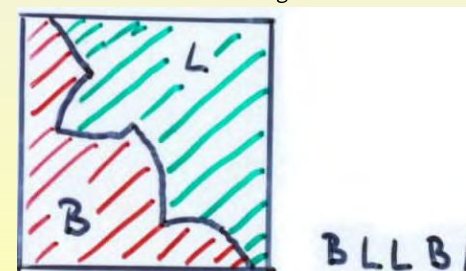
Segmentation Example

ball on a lawn

3) 2.,3. are reproduced, 1.,4. die, crossover

 $LLL|BL^{(1)}$ $c_{image} = 0.12$
 $LLB|LB^{(1)}$ $c_{image} = 0.20$

↓

 $LLLLB^{(x)}$ $c_{image} = 0.14$
 $LLBBL^{(2)}$ $c_{image} = 0.18$
die ($c_{image} = 0.0$)

UTIA

Segmentation Example

ball on a lawn

4) another crossover

$LLBL|B^{(2)}$ $C_{image} = 0.20$

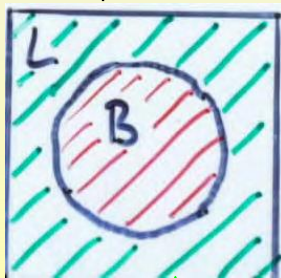
$LLBB|L^{(2)}$ $C_{image} = 0.18$

↓

$LLBLL^{(3)}$ $C_{image} = 0.10$

$LLBBB^{(3)}$ $C_{image} = 1.00$

stop



Genetic Feature Selection

• $\{0, 1, 1, 0, 1, 1, 1, \dots, 1\}$ $\begin{cases} 0, & \text{feature absent;} \\ 1, & \text{feature used.} \end{cases}$

fitness function evaluates **selected** features

- ⊖ computational cost of the search
- ⊖ local minima (a non-monotonic fitness function)
- ⊖ no guarantee to produce a good result



Data - BarkNet 1.0



BarkNet 1.0 23 000 bark samples from 23 classes, 3 cellphones Nexus 5, Samsung Galaxy S5, Samsung Galaxy S7, camera Panasonic Lumix DMC-TS5

Betula alleghaniensis, *Betula papyrifera*, *Quercus rubra*, *Picea glauca*, *Picea mariana*, *Picea abies*, *Picea rubens*,

Acer platanoides, *Acer rubrum*, *Acer saccharum*, *Fraxinus americana*, *Fagus grandifolia*, *Larix laricina*, *Ulmus*

americana, *Ostrya virginiana*, *Populus grandidentata*, *Populus tremuloides*, *Pinus strobus*, *Pinus rigida*, *Pinus*

resinosa, *Tsuga canadensis*, *Abies balsamea*, *Thuja occidentalis*.



Results - BarkNet 2DSCAR Method

	BOJ	BOP	CHR	EPB	EPN	EPO	EPR	ERB	ERR	ERS	FRA	sensitivity
BOJ	1169	34	11	3	3	4	3	0	7	1	0	93,15%
CHR	5	2	2438	5	1	5	8	0	60	55	12	89,50 %
EPB	8	2	7	495	3	53	6	0	2	6	0	83,05 %
EPN	2	0	1	3	795	5	6	0	3	7	4	89,83 %
EPO	13	3	25	64	5	1191	7	0	13	12	3	89,95 %
EPR	2	4	12	1	6	5	617	0	5	5	8	83,38 %
ERB	0	0	0	0	0	0	0	69	0	0	0	98,57 %
ERR	1	1	39	0	1	2	6	0	1407	52	36	83,95 %
ERS	2	2	40	1	0	1	7	0	59	1706	47	85,34 %
FRA	0	1	4	0	3	0	3	0	14	37	1289	87,57 %
HEG	2	2	8	5	0	2	3	0	23	3	1	92,50 %
MEL	2	1	4	5	40	7	20	0	1	5	2	96,80 %
ORA	2	0	15	0	1	0	2	0	21	32	24	88,92 %
OSV	0	1	10	0	0	0	2	0	17	12	9	95,42 %
PEG	1	1	0	0	0	0	0	0	0	0	0	93,75 %
PET	2	3	17	1	0	5	3	0	2	8	1	97,40 %
PIB	3	1	7	1	3	7	17	0	10	25	22	89,83 %
PID	0	0	1	0	1	1	4	0	0	2	3	82,93 %
PIR	2	0	6	4	5	19	1	1	2	11	1	90,60 %
PRU	3	2	72	1	3	12	17	0	25	16	8	93,31 %
SAB	12	5	2	5	15	5	5	0	1	4	1	90,13 %
THO	0	1	3	0	0	0	1	0	3	0	0	93,03 %
precision	93,00%	95,98%	92,10%	83,76%	90,86%	85,25%	84,87%	100,00%	88,27%	89,46%	94,43 %	Accuracy 90,31%



Results - BarkNet 2DSCAR Method

	HEG	MEL	ORA	OSV	PEG	PET	PIB	PID	PIR	PRU	SAB	THO	sensitivity
BOJ	3	0	3	0	1	5	1	0	0	1	6	2	93,15%
CHR	11	6	10	4	0	3	2	1	2	8	3	6	89,50 %
EPB	2	1	1	0	0	1	0	0	1	0	3	0	83,05 %
EPN	0	20	2	1	0	0	3	1	3	0	19	0	89,83 %
EPO	2	3	7	0	2	6	10	0	17	4	10	0	89,95 %
EPR	1	8	1	2	0	1	16	1	1	11	18	2	83,38 %
ERB	0	0	0	0	0	0	0	0	0	0	0	0	98,57 %
ERR	14	1	11	2	0	1	5	0	2	4	3	6	83,95 %
ERS	2	0	14	3	0	1	8	1	4	4	2	3	85,34 %
FRA	0	0	4	2	0	0	5	0	0	3	0	0	87,57 %
HEG	777	0	0	0	0	0	3	1	0	2	1	3	92,50 %
MEL	1	1814	2	0	0	0	11	2	3	2	13	1	96,80 %
ORA	0	0	682	5	0	0	0	1	1	0	0	1	88,92 %
OSV	0	1	11	584	0	2	3	0	2	5	0	1	95,42 %
PEG	0	0	0	0	60	0	0	0	0	1	1	1	93,75 %
PET	4	3	2	0	0	1010	0	0	0	4	0	1	97,40 %
PIB	2	4	3	0	0	0	919	3	4	5	6	5	89,83 %
PID	0	0	0	0	0	0	7	102	11	0	0	0	82,93 %
PIR	0	8	13	2	0	0	11	9	540	10	1	3	90,60 %
PRU	4	1	0	6	0	4	12	1	4	920	3	15	93,31 %
SAB	7	4	0	0	0	0	4	0	0	0	831	1	90,13 %
THO	1	0	0	1	1	0	2	0	0	1	0	694	93,03 %
precision	92,94%	93,70%	86,66%	88,48%	92,31%	94,75%	87,77%	77,27%	83,20%	81,49%	92,13%	98,02%	Accuracy 90,31%



Results - BarkNet 2DSCAR Method

87,7% $\rightarrow \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8, \theta_9, \theta_{10}, \theta_{11}, \theta_{12}, \theta_{13}, \theta_{14}, \theta_{15}, \theta_{16},$
 $\theta_{17}, \theta_{18}, \theta_{19}, \theta_{20}, \theta_{21}, \theta_{22}, \theta_{23}, \theta_{24}, \theta_{25}, \theta_{26}, \theta_{27}, \theta_{28}, \theta_{29}, \theta_{30},$
 $\theta_{31}, \theta_{32}, \theta_{33}$

88,6% $\rightarrow \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8, \theta_9, \theta_{10}, \theta_{11}, \theta_{12}, \theta_{13}, \theta_{14}, \theta_{15}, \theta_{16},$
 $\theta_{17}, \theta_{18}, \theta_{19}, \theta_{20}, \theta_{21}, \theta_{22}, \theta_{23}, \theta_{24}, \theta_{25}, \theta_{26}, \theta_{27}, \theta_{28}, \theta_{29}, \theta_{30},$
 $\theta_{31}, \theta_{32}, \theta_{33}$

90,3% $\rightarrow \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8, \theta_9, \theta_{10}, \theta_{11}, \theta_{12}, \theta_{13}, \theta_{14}, \theta_{15}, \theta_{16},$
 $\theta_{17}, \theta_{18}, \theta_{19}, \theta_{20}, \theta_{21}, \theta_{22}, \theta_{23}, \theta_{24}, \theta_{25}, \theta_{26}, \theta_{27}, \theta_{28}, \theta_{29}, \theta_{30},$
 $\theta_{31}, \theta_{32}, \theta_{33}$



Results - BarkTex 3DSCAR Method

(BP - Betula pendula, FS - Fagus silvatica, PA - Picea abies, PS - Pinus silvestris, QR - Quercus robur, RP -

Robinia pseudacacia)

	BP	FS	PA	PS	QR	RP	Sensitivity [%]
Betula pendula	64	0	0	2	2	0	94,1
Fagus silvatica	0	68	0	0	0	0	100,0
Picea abies	3	0	62	0	3	0	91,2
Pinus silvestris	0	0	1	67	0	0	98,5
Quercus robur	1	2	7	9	48	1	70,6
Robinia pseudacacia	1	0	0	1	1	65	95,6
Precision [%]	92,8	97,1	88,6	84,8	88,9	98,5	Accuracy 91,7



2D/3DSCAR Method

tree	precision [%]		sensitivity [%]	
	2DSCAR	3DSCAR	2DSCAR	3DSCAR
Populus grandidentata (osika)	92	87	94	91
Acer platanoides (javor)	100	96	83	76
Pinus rigida (borovice tuhá)	77	68	83	76

Classification results for three bark species which have too limited training sets in BarkNet for the alternative ResNet classifier.



Results

Dataset [%]	2D SCAR	3D SCAR	Boudra 2017	Fiel 2011	Sulc 2013	Hoang7 2017	Palm 2004	Porebski 2013	Sandid 2016	Sandid 2016b	Carpentier 2018
AFF	89.1	88.0	60.5	69.7	96.5	-	-	-	-	-	-
BarkTex	91.7	88.0	84.6	-	-	81.4	84.7	81.4	82.1	89.6	-
Trunk12	92.9	80.2	62.8	-	-	-	-	-	-	-	-
BarkNet	90.4	85.0	-	-	-	-	-	-	-	-	(64) 85

Comparison with the state-of-the-art '-' denotes lack of results in the particular article on the given dataset.

Remeš, V. - Haindl, M.: Bark recognition using novel rotationally invariant multispectral textural features, Pattern Recognition Letters, Elsevier, 2019, vol.125, pp. 612 - 617, ISSN 0167-8655.

