Evolutionary Computation (continued)

GA Quick Recap

- Typically applied to:
  - discrete optimization
- Attributed features:
  - not too fast
  - good heuristic for combinatorial optimization problems
- Special Features:
  - Traditionally emphasizes combining information from good parents (crossover)
  - many variants, e.g., reproduction models, operators

Evolution Strategies (ES)

Slides adapted from Chapter 4

book Eiben, Smith

ES

- Developed: Germany in the 1970's
- Early names: I. Rechenberg, H.-P. Schwefel
- Typically applied to:
  - numerical optimisation
- Attributed features:
  - fast
  - good optimizer for real-valued optimisation
- Special:
  - self-adaptation of (mutation) parameters standard

Introductory example

- Task: minimimise $f : R^n \rightarrow R$
- Algorithm: "two-membered ES" using
  - Vectors from R^n directly as chromosomes
  - Population size 1
  - Only mutation creating one child
  - Greedy selection

Introductory example: pseudocde

- Set t = 0
- Create initial point $x^t = \langle x_1{}^t,\ldots,x_n{}^t \rangle$
- REPEAT UNTIL (*TERMIN.COND* satisfied) DO
- Draw z_i from a Normal distr. for all i = 1,…,n
- $y_i{}^t = x_i{}^t + z_i$
- IF $f(x^t) < f(y^t)$ THEN $x^{t+1} = x^t$
  - ELSE $x^{t+1} = y^t$
  - FI
  - Set t = t+1
- OD

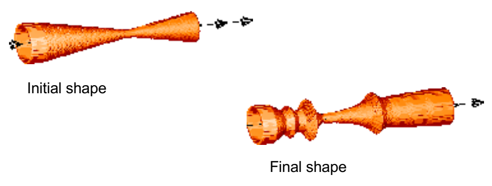
Introductory example: mutation mechanism

- z values drawn from Normal distribution $N(\xi,\sigma)$
  - mean ξ is set to 0
  - variation σ is called mutation step size
  - This mimics the evolutionary process that small changes occur more often than larger ones
- σ is varied on the fly by the "1/5 success rule": resets σ after every k iterations by
  - $\sigma = \sigma / c$ if $p_s > 1/5$ (wider search of the space)
  - $\sigma = \sigma \cdot c$ if $p_s < 1/5$ (concentrate more around current solutions)
  - $\sigma = \sigma$ if $p_s = 1/5$

  where p_s is the % of successful mutations (i.e. child fitter than parent) over a number of trials, $0.8 \leq c < 1$

An historical example: the jet nozzle experiment

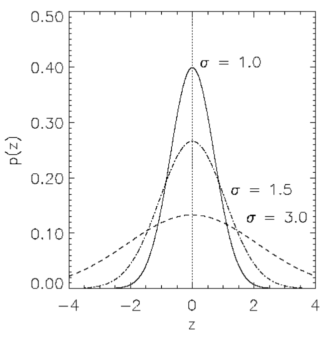
Task: to optimize the shape of a jet nozzle (a pipe or tube of varying cross sectional area, can be used to direct or modify the flow of a fluid)
Approach: random mutations to shape + selection



http://ls11-www.cs.tu-dortmund.de/people/schwefel/EADemos/

Genetic operators: mutations depend on σ

The one dimensional case



Representation

- Chromosomes consist of three parts:
  - Object input variables: $x_1,\ldots,x_n$
  - Strategy parameters:
    - Mutation step sizes: $\sigma_1,\ldots,\sigma_n$
    - Rotation angles: $\alpha_1,\ldots,\alpha_k$
- Not every component is always present
- Full size: $\langle x_1,\ldots,x_n, \sigma_1,\ldots,\sigma_n, \alpha_1,\ldots, \alpha_k \rangle$
- where $k = n(n-1)/2$ (no. of i,j pairs)

Mutation

- Main mechanism: changing value by adding random noise drawn from Normal distribution
- $x'_i = x_i + N(0,\sigma)$
- Key idea:
  - σ is part of the chromosome $\langle x_1,\ldots,x_n, \sigma \rangle$
  - σ is also mutated into σ' (see later how)
- Thus: mutation step size σ is coevolving with the solution x

Mutate σ first

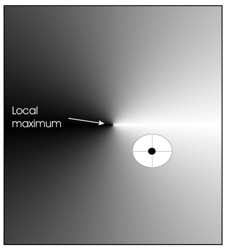
- Net mutation effect: $\langle x, \sigma \rangle \rightarrow \langle x', \sigma' \rangle$
- Order is important:
  - first $\sigma \rightarrow \sigma'$ (see later how)
  - then $x \rightarrow x' = x + N(0,\sigma')$
- Rationale: new $\langle x', \sigma' \rangle$ is evaluated twice
  - Primary: x' is good if $f(x')$ is good
  - Secondary: σ' is good if the x' it created is good
- This happens indirectly
- Reversing mutation order this would not work

Mutation case 1: Uncorrelated mutation with one σ

- Chromosomes: $\langle x_1,\ldots,x_n, \sigma \rangle$
- $\sigma' = \sigma \cdot \exp(\tau \cdot N(0,1))$
- $x'_i = x_i + \sigma' \cdot N(0,1)$
- τ "learning rate" parameter
- Typically $\tau \propto 1/ n^{½}$
- And we have a boundary rule $\sigma' < \varepsilon_0 \Rightarrow \sigma' = \varepsilon_0$

13

Mutants with uncorrelated global mutation 2D example



Circle: equal prob to move along x- and y-axis

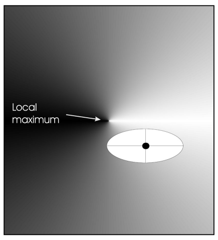
14

Mutation case 2: Uncorrelated mutation with n σ's

- Chromosomes: $\langle x_1,\ldots,x_n, \sigma_1,\ldots, \sigma_n \rangle$
- $\sigma'_i = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i (0,1))$
- $x'_i = x_i + \sigma'_i \cdot N_i (0,1)$
- Two learning rate parameters:
  - τ' overall learning rate (guarantee preserving all degrees of freedom)
  - τ coordinate wise learning rate
- $\tau' \propto 1/(2 n)^{½}$ and $\tau \propto 1/(2 n^{½})^{½}$
- And $\sigma_i' < \varepsilon_0 \Rightarrow \sigma_i' = \varepsilon_0$

15

Mutants with uncorrelated local mutation 2D example



Ellipse: bigger prob to move along x- than y-axis

16

Mutation case 3: Correlated mutations

- Chromosomes: $\langle x_1,\ldots,x_n, \sigma_1,\ldots, \sigma_n ,\alpha_1,\ldots, \alpha_k \rangle$
- where $k = n \cdot (n-1)/2$
- and the covariance matrix C is defined as:
  - $c_{ii} = \sigma_i^2$
  - $c_{ij} = 0$ if i and j are not correlated
  - $c_{ij} = ½ \cdot (\sigma_i^2 - \sigma_j^2) \cdot \tan(2 \alpha_{ij})$ if i and j are correlated
- Note the numbering / indices of the α's

17

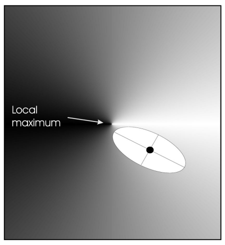
Correlated mutations cont'd

The mutation mechanism is then:

- $\sigma'_i = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i (0,1))$
- $\alpha'_j = \alpha_j + \beta \cdot N (0,1)$
- $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{N}(\boldsymbol{0},\boldsymbol{C'})$
  - ***x*** stands for the vector $\langle x_1,\ldots,x_n \rangle$
  - ***C'*** is the covariance matrix ***C*** after mutation of the α values
- $\tau' \propto 1/(2 n)^{½}$ and $\tau \propto 1/(2 n^{½})^{½}$ and $\beta = \frac{5}{180}\pi$
- $\sigma_i' < \varepsilon_0 \Rightarrow \sigma_i' = \varepsilon_0$ and
- $| \alpha'_j | > \pi \Rightarrow \alpha'_j = \alpha'_j - 2 \pi \operatorname{sign}(\alpha'_j)$

18

Mutants with correlated mutation



Ellipse with any orientation: other directions than along x- or y-axis may also be favoured

Recombination

- Creates one child
- Acts per variable / position by either
  - Averaging parental values, or
  - Selecting one of the parental values
- From two or more parents by either:
  - Using two selected parents to make a child
  - Selecting two parents for each position anew

Names of recombinations

	Two fixed parents	Two parents selected for each i
$z_i = (x_i + y_i)/2$	Local intermediary	Global intermediary
z_i is x_i or y_i chosen randomly	Local discrete	Global discrete

Parent selection

- Parents are selected by uniform random distribution whenever an operator needs one/some
- Thus: ES parent selection is unbiased - every individual has the same probability to be selected
- Note that in ES “parent” means a population member (in GA’s: a population member selected to undergo variation)

Survivor selection

- Applied after creating λ children from the μ parents by mutation and recombination
- Deterministically chops off the “bad stuff”
- Basis of selection is either:
  - The set of children only: (μ,λ)-selection
  - The set of parents and children: $(\mu+\lambda)$-selection

Survivor selection cont’d

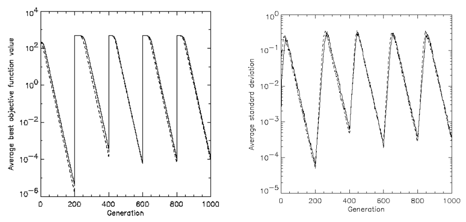
- $(\mu+\lambda)$-selection is an elitist strategy
- (μ,λ)-selection can “forget”
- Often (μ,λ)-selection is preferred for:
  - Better in leaving local optima
  - Better in following moving optima
  - Using the + strategy bad σ values can survive in $\langle x,\sigma \rangle$ too long if their host x is very fit
- Selective pressure in ES is very high ($\lambda \approx 7 \cdot \mu$ is the common setting)

Self-adaptation illustrated

- Given a dynamically changing fitness landscape (optimum location shifted every 200 generations)
- Self-adaptive ES is able to
  - follow the optimum and
  - adjust the mutation step size after every shift !

25

Self-adaptation illustrated cont'd



Changes in the fitness values (left) and the mutation step sizes (right)

26

Prerequisites for self-adaptation

- $\mu > 1$ to carry different strategies
- $\lambda > \mu$ to generate offspring surplus
- Not "too" strong selection, e.g., $\lambda \approx 7 \cdot \mu$
- (μ,λ)-selection to get rid of misadapted σ's
- Mixing strategy parameters by (intermediary) recombination on them

27

Example application: the cherry brandy experiment

- Task to create a colour mix yielding a target colour (that of a well known cherry brandy)
- Ingredients: water + red, yellow, blue dye
- Representation: $\langle$ w, r, y ,b $\rangle$ no self-adaptation!
- Values scaled to give a predefined total volume (30 ml)
- Mutation: lo / med / hi σ values used with equal chance
- Selection: (1,8) strategy

28

Example application: cherry brandy experiment cont'd

- Fitness: students effectively making the mix and comparing it with target colour
- Termination criterion: student satisfied with mixed colour
- Solution is found mostly within 20 generations
- Accuracy is very good

29

Example application: the Ackley function (Bäck et al '93)

- The Ackley function (here used with n =30):

$$f(x) = -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$$

- Evolution strategy:
  - Representation:
    - $-30 < x_i < 30$ (coincidence of 30's!)
    - 30 step sizes
  - (30,200) selection
  - Termination : after 200000 fitness evaluations
  - Results: average best solution is $7.48 \cdot 10^{-8}$ (very good)

30

ES technical summary tableau

Representation	Real-valued vectors
Recombination	Discrete or intermediary
Mutation	Gaussian perturbation
Parent selection	Uniform random
Survivor selection	(μ,λ) or $(\mu+\lambda)$
Specialty	Self-adaptation of mutation step sizes

31

Genetic Programming

Slides adapted from Chapter 6

Book by Eiben, Smith

32

GP quick overview

- Developed: USA in the 1990's
- Early names: J. Koza
- Typically applied to:
  - machine learning tasks (symbolic regression, classification…)
- Attributed features:
  - competes with neural nets and alike
  - needs huge populations (thousands)
  - slow
- Special:
  - Structured representation: trees, graphs
  - mutation possible but not necessary (disputed!)

33

Introductory example: credit scoring

- Bank wants to distinguish good from bad loan applicants
- Model needed generated from historical data

ID	No of children	Salary	Marital status	OK?
ID-1	2	45000	Married	0
ID-2	0	30000	Single	1
ID-3	1	40000	Divorced	1
…				

34

Introductory example: credit scoring

- A possible model:

  IF (NOC = 2) AND (S > 80000) THEN good ELSE bad
- In general:

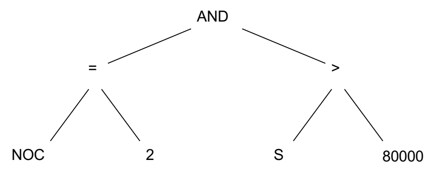
  IF formula THEN good ELSE bad
- Only unknown is the right formula, hence
- Our search space (phenotypes) is the set of formulas
- Natural fitness of a formula: percentage of well classified cases of the model it stands for
- Natural representation of formulas (genotypes) is: parse trees

35

Introductory example: credit scoring

IF (NOC = 2) AND (S > 80000) THEN good ELSE bad

can be represented by the following tree



36

Tree based representation

- Trees are a general form, e.g. consider
- Arithmetic formula

$$2\cdot\pi+\left((x+3)-\frac{y}{5+1}\right)$$

- Logical formula

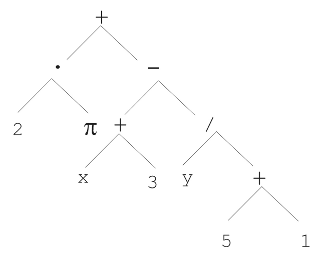
(x ∧ true) → ((x ∨ y) ∨ (z ↔ (x ∧ y)))

- Program

```
i =1;
while (i < 20)
{
      i = i +1
}
```

37

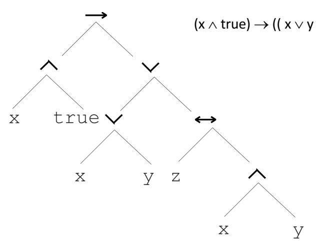
Tree based representation



$$2\cdot\pi+\left((x+3)-\frac{y}{5+1}\right)$$

38

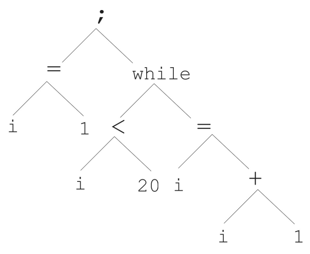
Tree based representation



(x ∧ true) → ((x ∨ y) ∨ (z ↔ (x ∧ y)))

39

Tree based representation



```
i =1;
while (i < 20)
{
      i = i +1
}
```

40

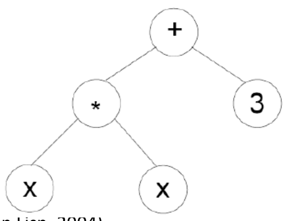
Koza's evolution of LISP programs

- Lisp is a functional language: f (x; y) is written as (f x y)
- 10 - (3+4) is written as (-10 (+ 3 4))
- Lisp programs can be represented as trees

f (x) = x^2 + 3   is written as   (+ (* x x) 3)

Here, + and * are function symbols (non-terminals), x and 3 are terminals.

Given a bag of terminals and non-terminals, we can directly generate programs!

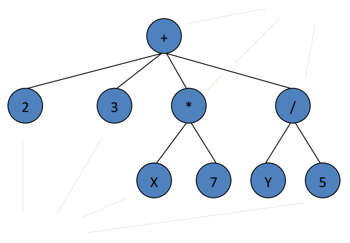


(Peter Seibel: Practical Common Lisp, 2004)

41

Tree Representation

LISP S-expression   (+ 2 3 (* X 7) (/ Y 5))



42

Tree based representation

- In GA and ES chromosomes are linear structures (bit strings, integer string, real-valued vectors, permutations)
- Tree shaped chromosomes are non-linear structures
- In GA and ES the size of the chromosomes is fixed
- Trees in GP may vary in depth and width

43

Tree based representation

- Symbolic expressions can be defined by
  - Terminal set T
  - Function set F (with the arities of function symbols)
- Adopting the following general recursive definition:
  1. Every $t \in T$ is a correct expression
  2. $f(e_1, \ldots, e_n)$ is a correct expression if $f \in F$, arity(f)=n and $e_1, \ldots, e_n$ are correct expressions
  3. There are no other forms of correct expressions
- In general, expressions in GP are not typed (closure property: any $f \in F$ can take any $g \in F$ as argument)

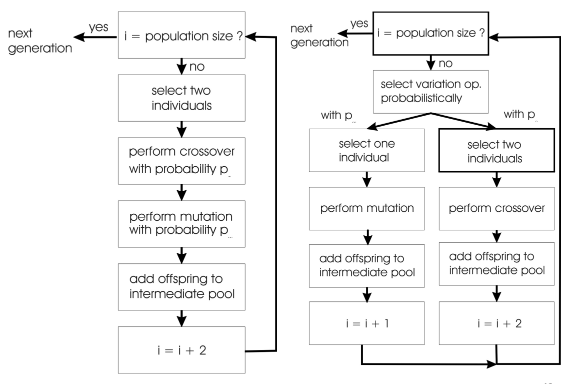
44

Offspring creation scheme

Compare

- GA scheme using crossover AND mutation sequentially (be it probabilistically)
- GP scheme using crossover OR mutation (chosen probabilistically)

45

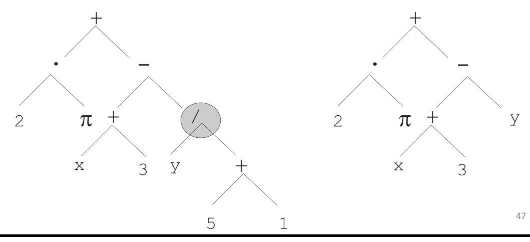


GA flowchart GP flowchart

46

Mutation

- Most common mutation: replace randomly chosen subtree by randomly generated tree



47

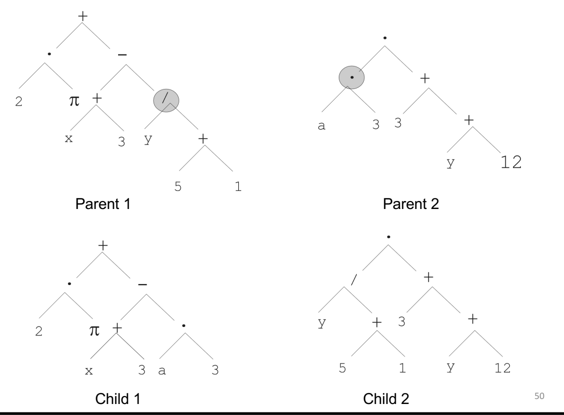
Mutation cont'd

- Mutation has two parameters:
  - Probability p_m to choose mutation vs. recombination
  - Probability to chose an internal point as the root of the subtree to be replaced
- Remarkably p_m is advised to be 0 (Koza'92) or very small, like 0.05 (Banzhaf et al. '98)
- The size of the child can exceed the size of the parent

48

Recombination

- Most common recombination: exchange two randomly chosen subtrees among the parents
- Recombination has two parameters:
  - Probability p_c to choose recombination vs. mutation
  - Probability to chose an internal point within each parent as crossover point
- The size of offspring can exceed that of the parents



Selection

- Parent selection typically fitness proportionate or tournament
- Survivor selection:
  - Typical: generational scheme (thus none)
  - Recently steady-state also popular for its elitism

Initialisation

- Maximum initial depth of trees D_{max} is set
- Full method (each branch has depth = D_{max}):
  - nodes at depth d < D_{max} randomly chosen from function set F
  - nodes at depth d = D_{max} randomly chosen from terminal set T
- Grow method (each branch has depth ≤ D_{max}):
  - nodes at depth d < D_{max} randomly chosen from $F \cup T$
  - nodes at depth d = D_{max} randomly chosen from T
- Common GP initialisation: ramped half-and-half, where grow & full method each deliver half of initial population

Bloat

- Bloat = "survival of the fattest", i.e., the tree sizes in the population are increasing over time
- Ongoing research and debate about the reasons
- Needs countermeasures, e.g.
  - Prohibiting variation operators that would deliver "too big" children
  - Parsimony pressure: penalty for being oversized

Literature Riccardo Poli, William B Langdon, Nicholas F. McPhee (2008) A Field Guide to Genetic Programming.

Problems involving "physical" environments

- Trees for data fitting vs. trees (programs) that are "really" executable
- Execution can change the environment → the calculation of fitness
- Example: robot controller
- Fitness calculations mostly by simulation, ranging from expensive to extremely expensive (in time)

Example application: symbolic regression

- Given some points in $\mathbf{R}^2$, $(x_1, y_1), \ldots, (x_n, y_n)$
- Find function f(x) s.t. $\forall i = 1, \ldots, n : f(x_i) = y_i$
- Possible GP algorithm:
  - Representation by F = {+, -, /, sin, cos}, T = $\mathbf{R} \cup \{x\}$
  - Fitness is the error $err(f) = \sum_{i=1}^{n} (f(x_i) - y_i)^2$
  - All operators standard
  - pop.size = 1000, ramped half-half initialisation
  - Termination: n "hits" or 50000 fitness evaluations reached (where "hit" is if $| f(x_i) - y_i | < 0.0001$)

55

GP: issue

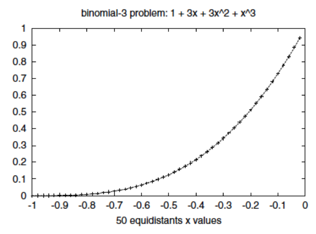
GP difficulty may increase even if the combinatorial search space does not change and the fitness function is smooth.

Daida, Jason M., et al. "What makes a problem GP-hard? Analysis of a tunably difficult problem in genetic programming." Genetic Programming and Evolvable Machines 2.2 (2001): 165-191.

56

Example: symbolic regression

Consider as fitness cases 50 equidistant points over the interval [-1, 0) generated from the target function.



The (unknown) target function is $f(x) = 1 + 3x + 3x^2 + x^3$.

57

GP experimental setting

- Functional set: $\{+, -, \times, \div\}$
- Terminal set:
  - X the symbolic variable
  - A set ERC (ephemeral random constants) of r random constants in the interval $[-a_R, a_R]$.
- ERC constants randomly sampled in the interval at population initialization
- Fitness is 1/(1 + (sum of absolute error values))

58

Experimental setting

- population size = 500;
- crossover rate = 0.9;
- replication rate = 0.1;
- population initialization with ramped half-and-half;
- initialization depth of 2 to 6 levels;
- fitness proportional selection;
- maximum number of generations = 200;
- maximum tree depth = 26.

59

Results

Let us see how the performance of this GP varies wrt range of ERC interval

Table 1. Main Part Summary. This table shows the total number of trials (out of 600 trials) that scored perfectly, in the upper decile, and in the upper quartile.

	a_R	Perfect	1 Decile	1 Quartile
No ERC $\rightarrow$	none	502	515	546
	0.1	14	42	130
	1	219	329	463
	2	144	285	433
	3	105	239	390
	10	57	145	312
	100	9	32	104
	1000	3	4	5

60

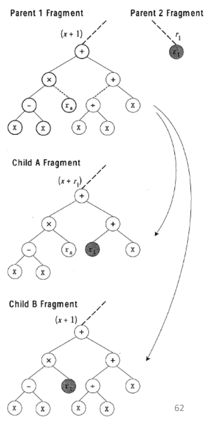
Observations

- The combinatorial search space does not change (r constants, only their value changes)
- The fitness function is not rugged

So why such difference in performance when changing the value of the constants?

Values in ERC matter

- ra not expressed in parent 1 (is multiplied by 0)
- r1 part of parent 2, can occur in the next generation as part of offspring A or B:
  - in A equivalent to (x + r1).
  - in B equivalent to (x + 1).
- Suppose (x + 1) is a desired fragment towards the solution of the problem. The magnitude of the difference between A and B increases as the value of r1 increases.
- The probability of this event increases as the range of r1 increases: 0.9 in [-1, 1] appears alternately as (x + 1) or (x + 0.9); 999.9 in [-1000, 1000] appears alternately as (x + 1) or (x + 999.9).



Practical guidelines

Pay attention to composition and interaction of function and terminal sets.

Apply a two-stage process:

1. Use GP to discover a selected subset of terminals and functions from a more general set.
2. Use GP to determine an appropriate data model.

Overall practical guidelines

- Study your populations: analyze mean and variance of fitness, trees depth, size, code used, run time, ... and correlations among these.
- Runs can be very long: consider checkpoint results.
- Control bloat.
- Encourage diversity and save good candidates.

When to use GP

- For machine learning tasks like protein structure prediction, symbolic regression, feature selection/importance especially in bioinformatics applications, classification, etc.
- For “black art" problems involving synthesis of topology and sizing of systems, like analog circuits.

Example

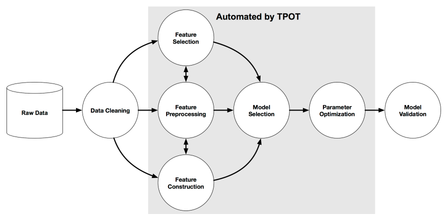
Automate pipeline creation and algorithm selection for machine learning.

TPOT is a Python Automated Machine Learning tool built on top of scikit-learn, a general-purpose Python machine learning library.
https://github.com/EpistasisLab/tpot

Randal S. Olson, Ryan J. Urbanowicz, Peter C. Andrews, Nicole A. Lavender, La Creis Kidd, and Jason H. Moore (2016). Automating biomedical data science through tree-based pipeline optimization. *Applications of Evolutionary Computation*, pages 123-137.

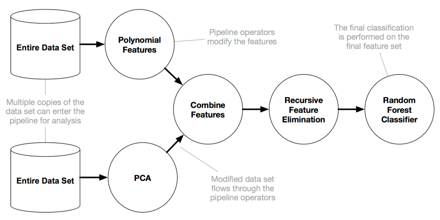
TPOT

An example Machine Learning pipeline:



67

TPOT


Entire Data Set
Polynomial Features
Pipeline operators modify the features
The final classification is performed on the final feature set
Multiple copies of the data set can enter the pipeline for analysis
Combine Features
Recursive Feature Elimination
Random Forest Classifier
Entire Data Set
PCA
Modified data set flows through the pipeline operators

- In TPOT a GP builds trees of pipeline operators to maximize the final classification accuracy of the pipeline.
- GP evolves the sequence of pipeline operators that act on the data set as well as the parameters of these operators, e.g., the number of trees in a random forest or the number of features to select during feature selection.
- It uses the Python GP package DEAP.

68

Other related systems

Autostacker: evolves multiple ensemble models within a stacked architecture

DarwinML: graph-based

69

GP technical summary tableau

Representation	Tree structures
Recombination	Exchange of subtrees
Mutation	Random change in trees
Parent selection	Fitness proportional
Survivor selection	Generational replacement

70

Ideas for project

- GP for automatic machine learning: see http://automl.chalearn.org/
- GP for evolving deep neural networks (see e.g. papers at GP 2018 conference)
- For new trends in EC see recent papers at the GECCO and PPSN conferences.

71

Deadline for the first assignment is 17 February

Upload your report with answers and the source code in Brightspace.

Next week: Swarm Intelligence

72