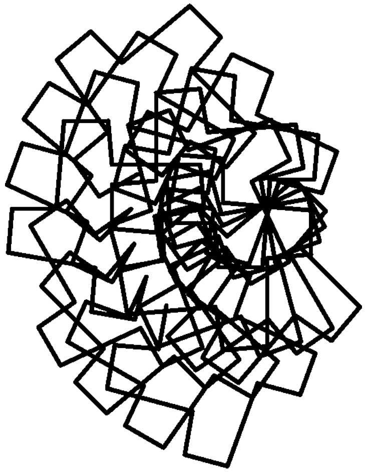
Pattern Classification with GP

Alexandros Agapitos

Natural computing Research and Applications group

UCD Complex and Adaptive Systems Lab



UCD Natural Computing Research & Applications (NCRA)

Outline

- Brief recap of Genetic Programming.
- Pattern classification with Genetic Programming.
    - Feature construction.
    - Classifier induction.
    - Ensemble induction.
- Classifier generalisation
    - Model selection based on a validation dataset.

Components of an evolutionary problem solver

1. Fitness evaluation.

   Fitness stands for solution quality.

Components of an evolutionary problem solver

1. Fitness evaluation.

   Fitness stands for solution quality.

2. Reproduction.
   - Crossover.
   - Mutation.

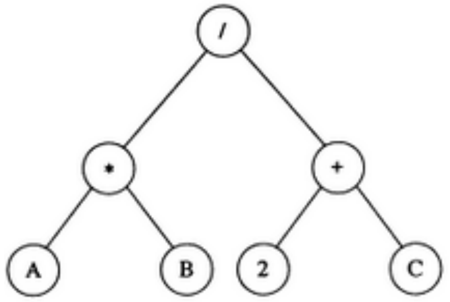
Components of an evolutionary problem solver

1. Fitness evaluation.
   Fitness stands for solution quality.
2. Reproduction.
   - Crossover.
   - Mutation.
3. Selection.
   - Parent selection.
   - Survival selection.

Program initialisation

Full **method:**

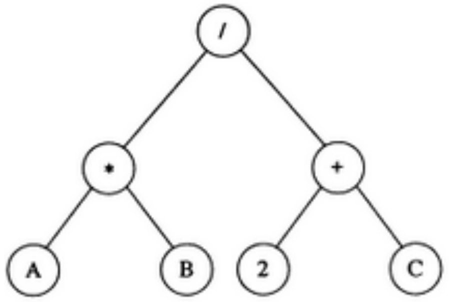
1. Choose a function as the root.
2. Recursively assign functions to the child nodes, until max. depth is reached at which point we chose a terminal.



Program initialisation

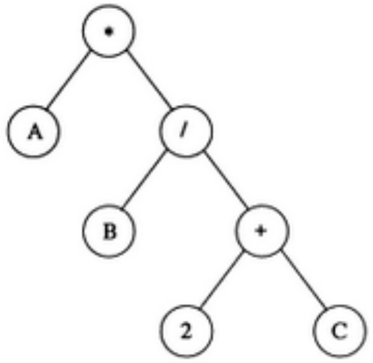
***Full* method:**

1. Choose a function as the root.
2. Recursively assign functions to the child nodes, until max. depth is reached at which point we chose a terminal.



***Grow* method:**

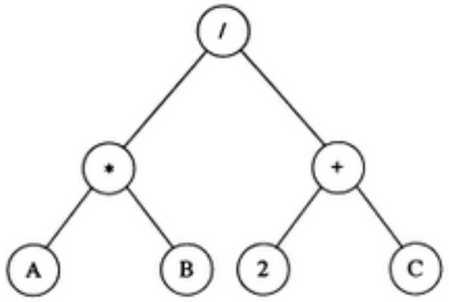
1. Choose a function as the root.
2. A function ***or*** a terminal can be assigned to a child node at any depth.



Program initialisation

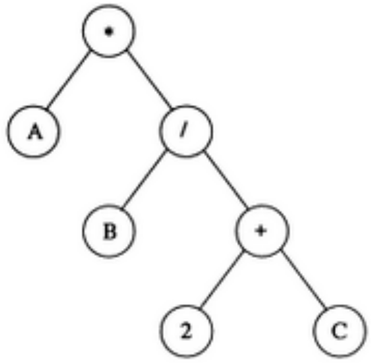
Full **method:**

1. Choose a function as the root.
2. Recursively assign functions to the child nodes, until max. depth is reached at which point we chose a terminal.



Grow **method:**

1. Choose a function as the root.
2. A function ***or*** a terminal can be assigned to a child node at any depth.

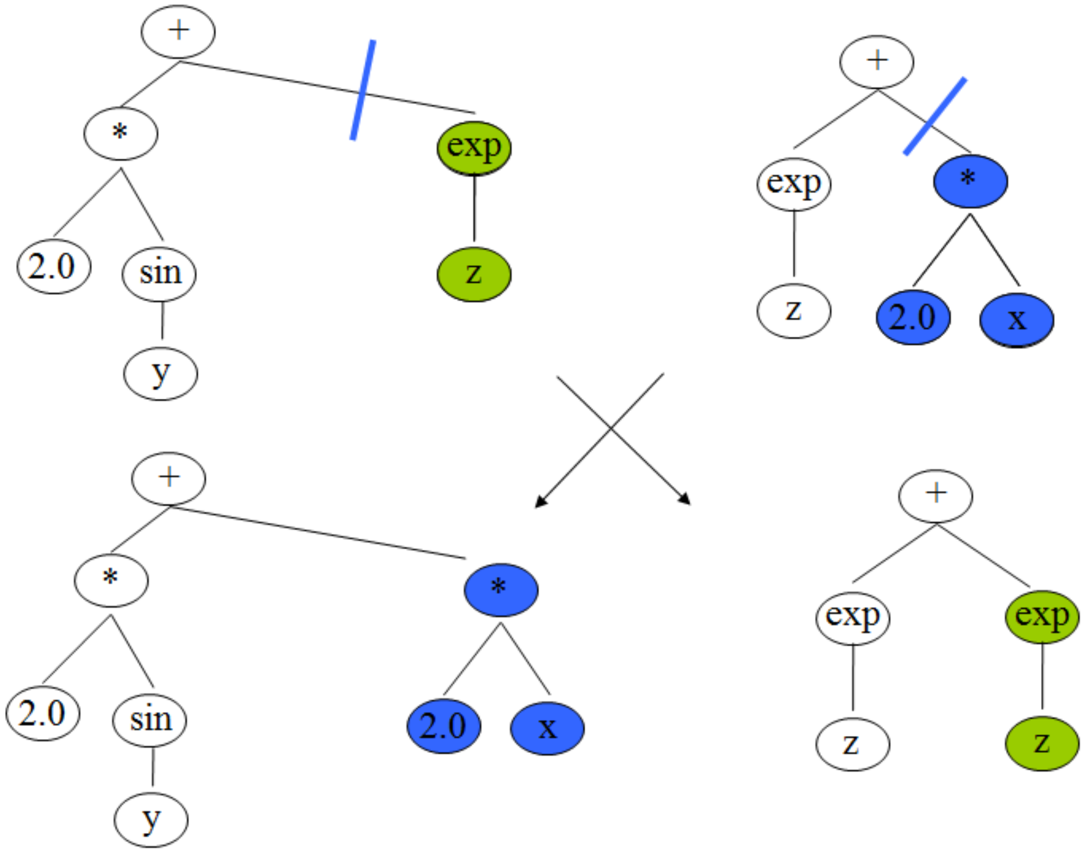


Ramped-half-and-half **method:**

- 50% full, 50% grow.
- Max. depth is ramped, so individuals are created in a range of depths.

Crossover

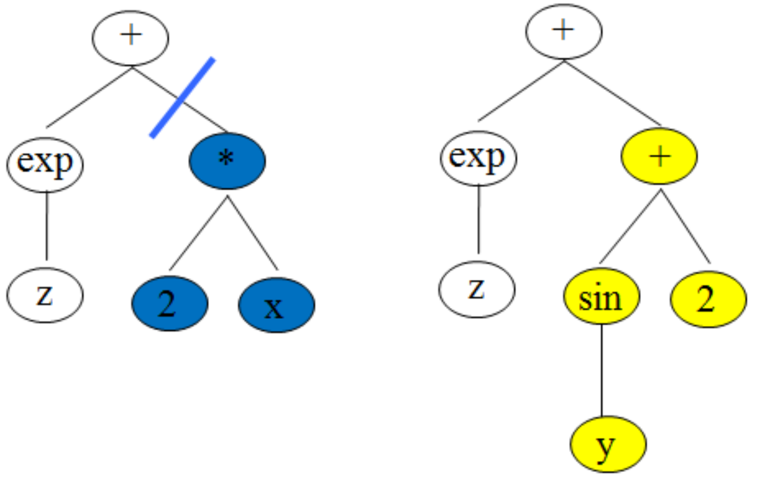
1. Select two parents.
2. Independently pick random nodes.
3. Swap sub-trees.



Mutation

Sub-tree mutation:

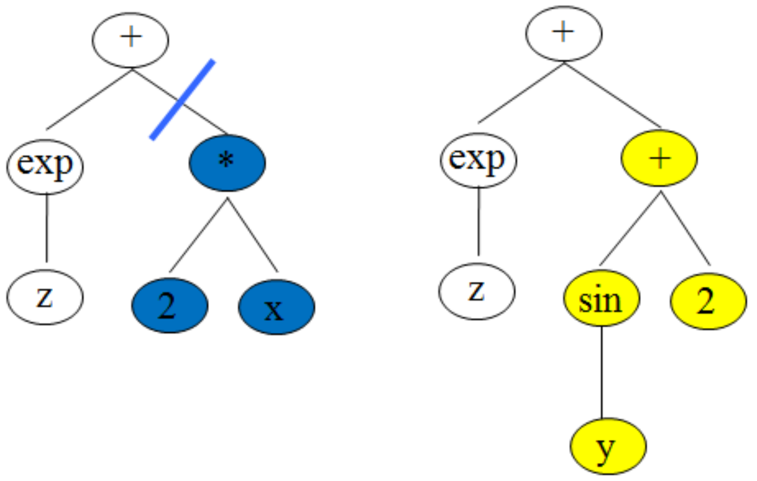
1. Randomly pick a node.
2. Delete sub-tree at selected node.
3. Generate random sub-tree (as per initialisation) in its place.



Mutation

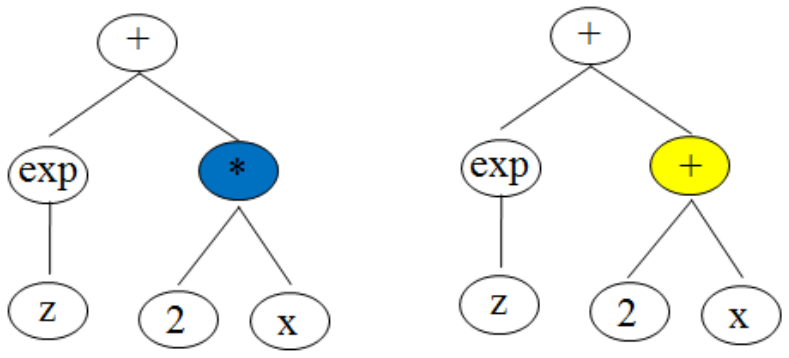
Sub-tree mutation:

1. Randomly pick a node.
2. Delete sub-tree at selected node.
3. Generate random sub-tree (as per initialisation) in its place.



Point mutation:

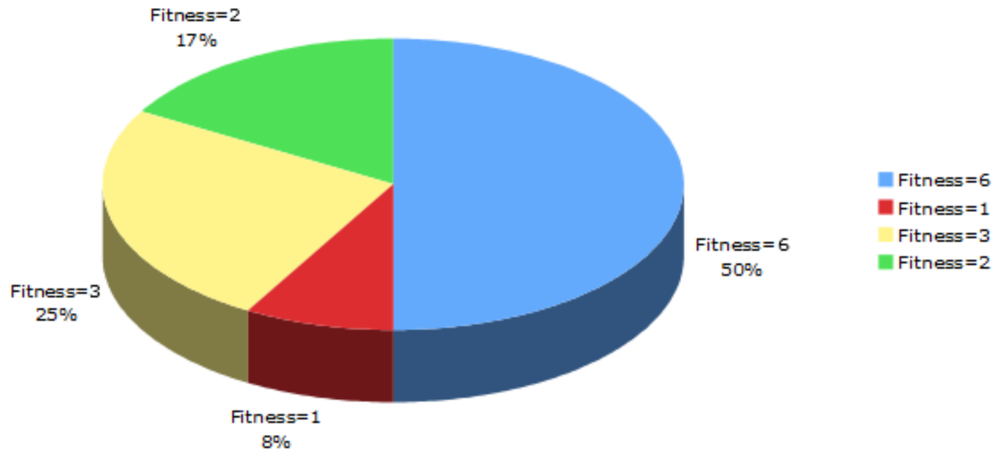
1. Randomly pick a node.
2. A terminal is replaced by another terminal, or a function is replaced by another function of the same arity.



Selection

Fitness proportionate selection:

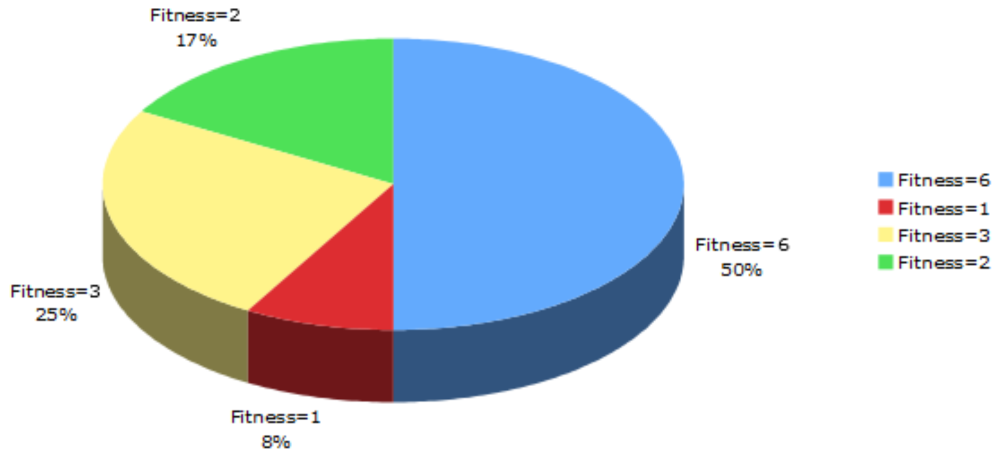
- High selection pressure earlier on.
- Low selection pressure later.



Selection

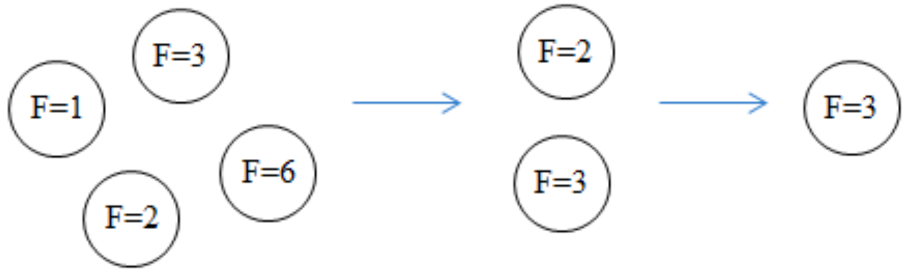
Fitness proportionate selection:

- High selection pressure earlier on.
- Low selection pressure later.



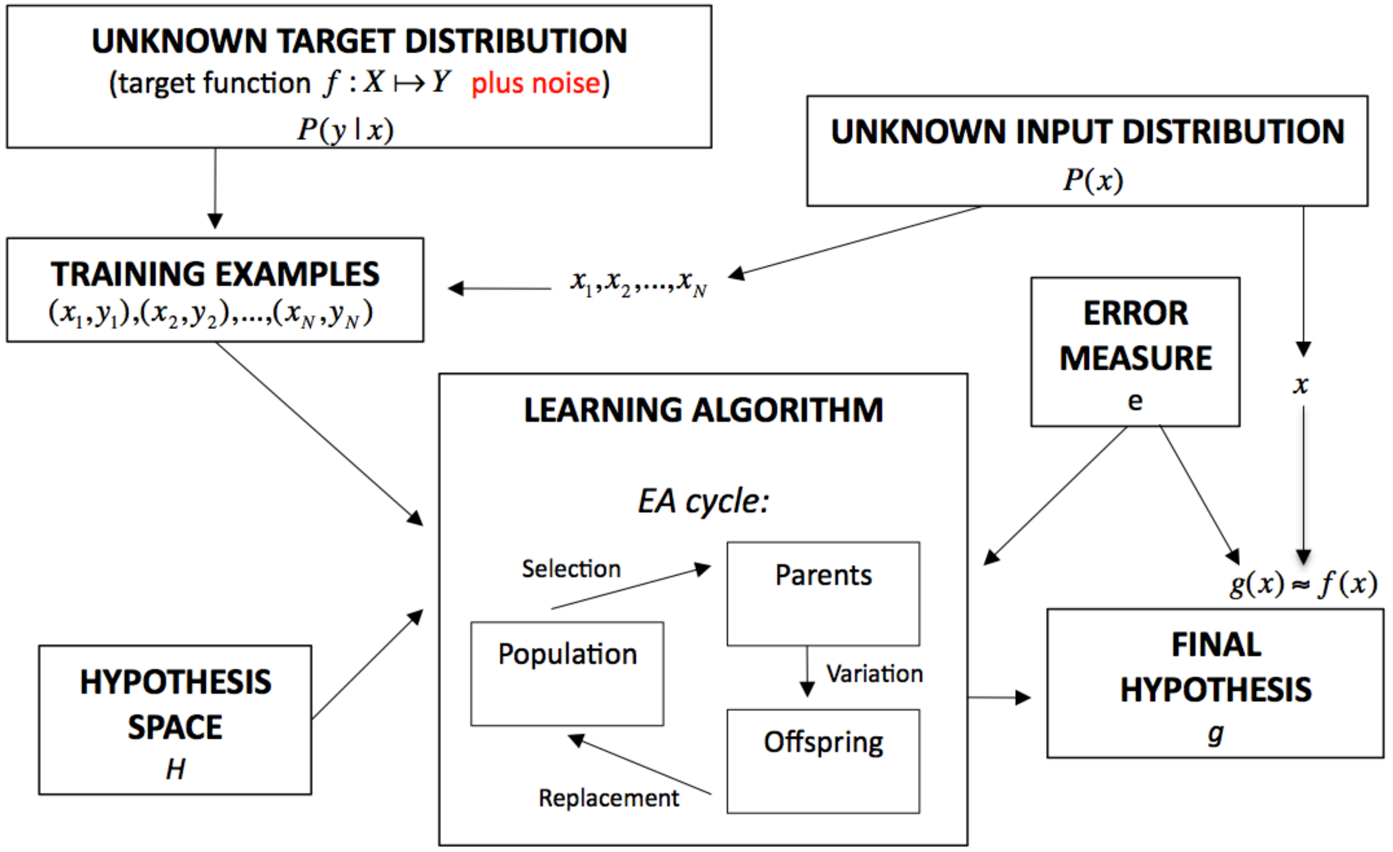
Tournament selection:

- Select *k* individuals at random.
- Best of *k* individuals becomes parent.
- Selection pressure easily adjustable.

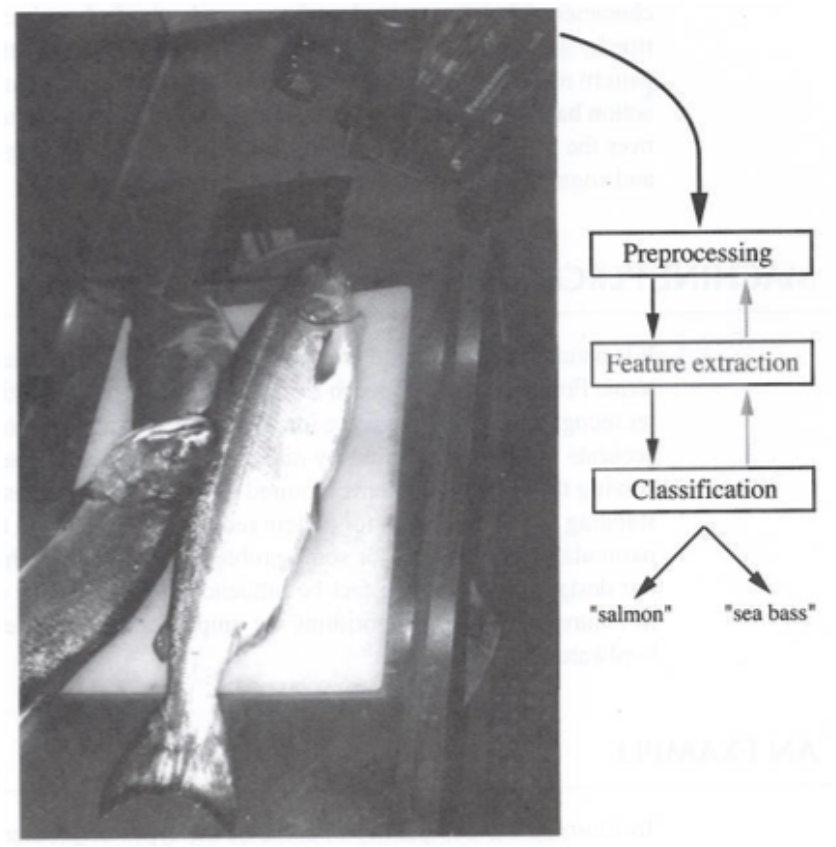


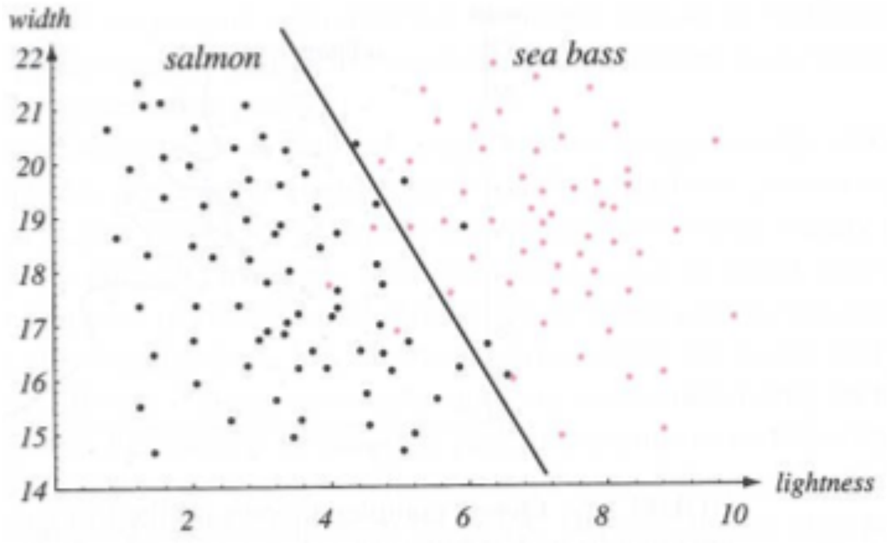
Pattern classification

- The goal of classification is to take an input vector x and to assign it to one of K discrete classes C_k, k=1, …K.
- The input vector is called *feature* vector.
- Supervised learning induces a model based on a training set of labelled examples.



An example: Fish classification





$$y(x) = wx + w_0 = \sum_{i=1}^{D} w_i x_i + w_0$$

GP in classification

- Pre-processing
    - Feature selection.
    - Feature construction.

GP in classification

- Pre-processing
    - Feature selection.
    - Feature construction.

- Model induction
    - Decision trees (discrete-valued functions)
    - Discriminant functions (real-valued functions)

GP in classification

- Pre-processing
  - Feature selection.
  - Feature construction.
- Model induction
  - Decision trees (discrete-valued functions)
  - Discriminant functions (real-valued functions)
- Ensemble classifiers

Pre-processing: Data transformations

The goal is to perform a transformation of representation, which, given the original vectors of features F_0, creates a new representation F derived from F_0 that maximises some criterion.

1. **Feature selection:** The resulting representation is a subset of the original one.
2. **Feature weighting:** The transformation method assigns weights to particular attributes. The weight reflects the relative importance of an attribute.
3. **Feature construction:** New features are created as non-linear transformations of existing features.

Pre-processing: Data transformations

The goal is to perform a transformation of representation, which, given the original vectors of features F_0, creates a new representation F derived from F_0 that maximises some criterion.

1. **Feature selection:** The resulting representation is a subset of the original one.
2. **Feature weighting:** The transformation method assigns weights to particular attributes. The weight reflects the relative importance of an attribute.
3. **Feature construction:** New features are created as non-linear transformations of existing features.

An important consideration:

Both feature selection and feature construction are somewhat inherent in the GP learning process.

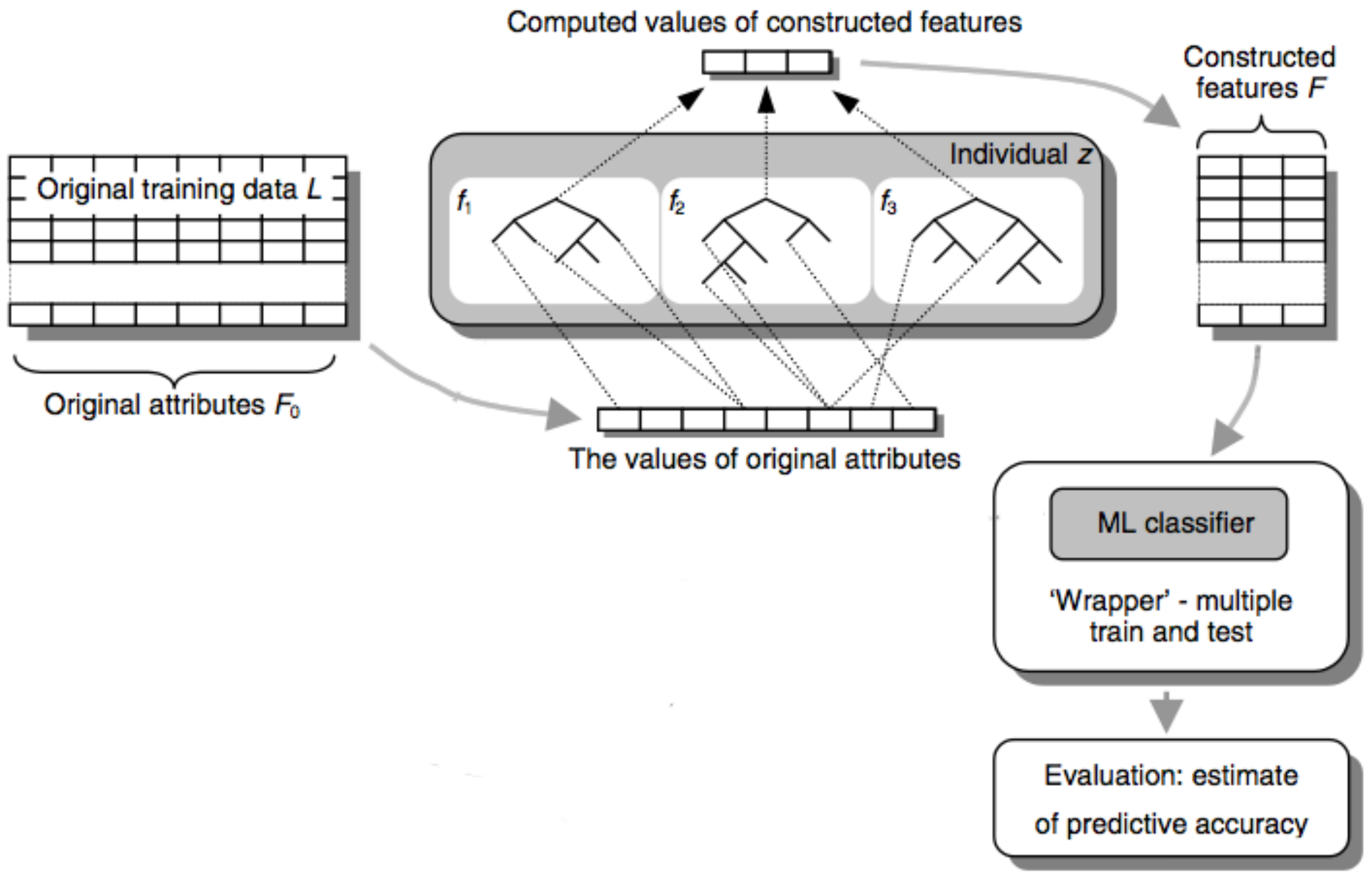
GP-based Feature Construction

1. **Filter approach:** Here the pre-processing is performed before processing with the model induction. Usually, some kind of statistical, logical or information theoretic criterion is used as the basis of pre-processing.

GP-based Feature Construction

1. **Filter approach:** Here the pre-processing is performed before processing with the model induction. Usually, some kind of statistical, logical or information theoretic criterion is used as the basis of pre-processing.
2. **Wrapper approach:** The executions of pre-processing and model induction are interleaved.
   - Each individual of the population encodes a fixed-length vector of evolved feature definitions.
   - Each feature definition is an expression-tree composed of various arithmetic and logic operators, numerical constants, and the values of the original features.
   - During evolution, each individual undergoes standard search operations.
   - The values of evolved features are computed for all training cases – a transformed ML dataset is obtained.
   - Cross-validation is carried-out using an inductive ML algorithm (i.e. Decision tree) that is trained on the transformed features.
   - The evaluation reflects the utility of the new transformation.

GP-based Feature Construction



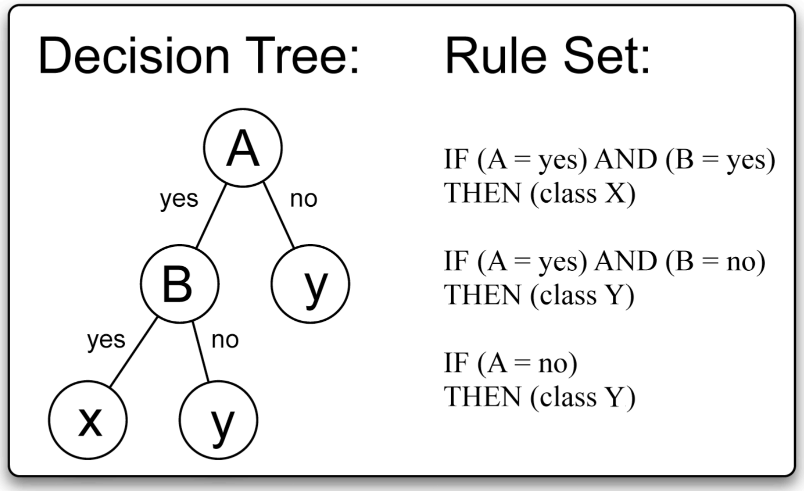
Computed values of constructed features
Constructed features F
Individual z
f_1
f_2
f_3
Original training data L
Original attributes F_0
The values of original attributes
ML classifier
'Wrapper' - multiple train and test
Evaluation: estimate of predictive accuracy

Classifier induction with GP

1. Discrete-valued classifiers
2. Real-valued classifiers

Discrete-valued classifiers

- Decision trees are formed by a conjunction of constraints on the values of features.



Rule Set:

IF (A = yes) AND (B = yes)
THEN (class X)

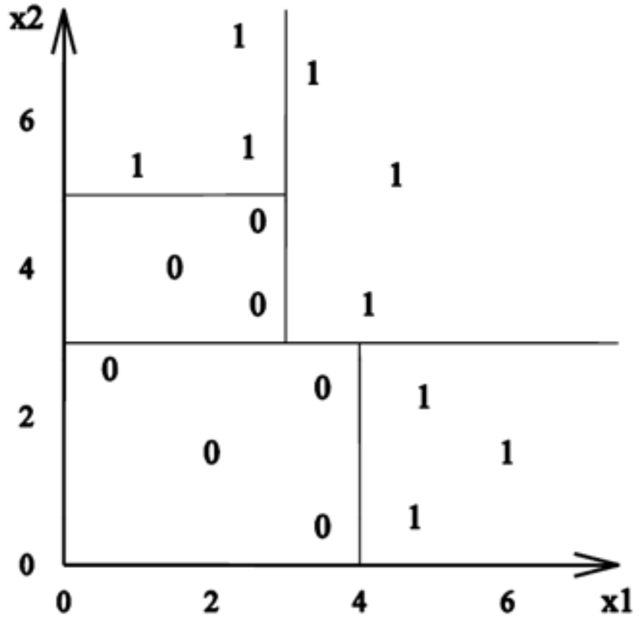
IF (A = yes) AND (B = no)
THEN (class Y)

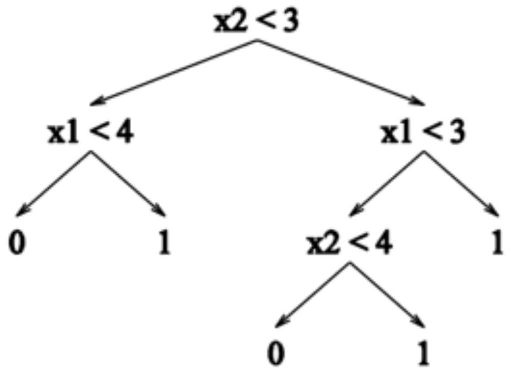
IF (A = no)
THEN (class Y)

- Types of DTs:
  1. **Univariate** (a.k.a. axis-parallel) decision trees: test a single variable at each predicate.
  2. **Linear multivariate** (a.k.a. oblique) decision trees: a linear combination of features is tested at each predicate.
- Inherent ability to represent both *2*-class and *n*-class classification models.
- Fitness function: classification accuracy (CA), i.e. given L training examples, CA = CorrectClassifications / L

Evolving univariate decision trees

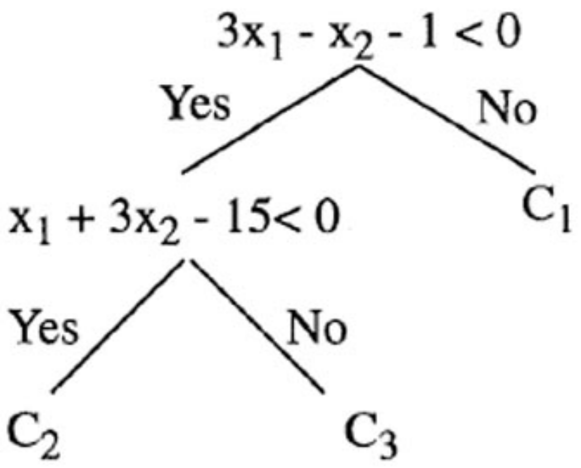
- We need a strongly-typed (tree-root type: int) system or a grammar.
- Function set:
  1. **If-then-else**. arg types: (boolean, int, int). return type: boolean
  2. **logic operators** <, >, =. arg types: (double, double). return type: boolean
- Terminal set:
  1. classification output (type int)
  2. features (type double)
  3. constants (type double)

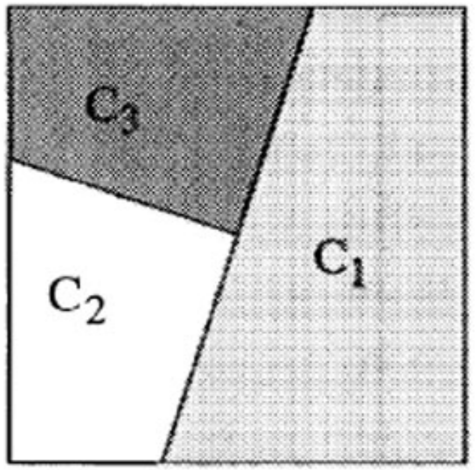




Evolving linear multivariate decision trees

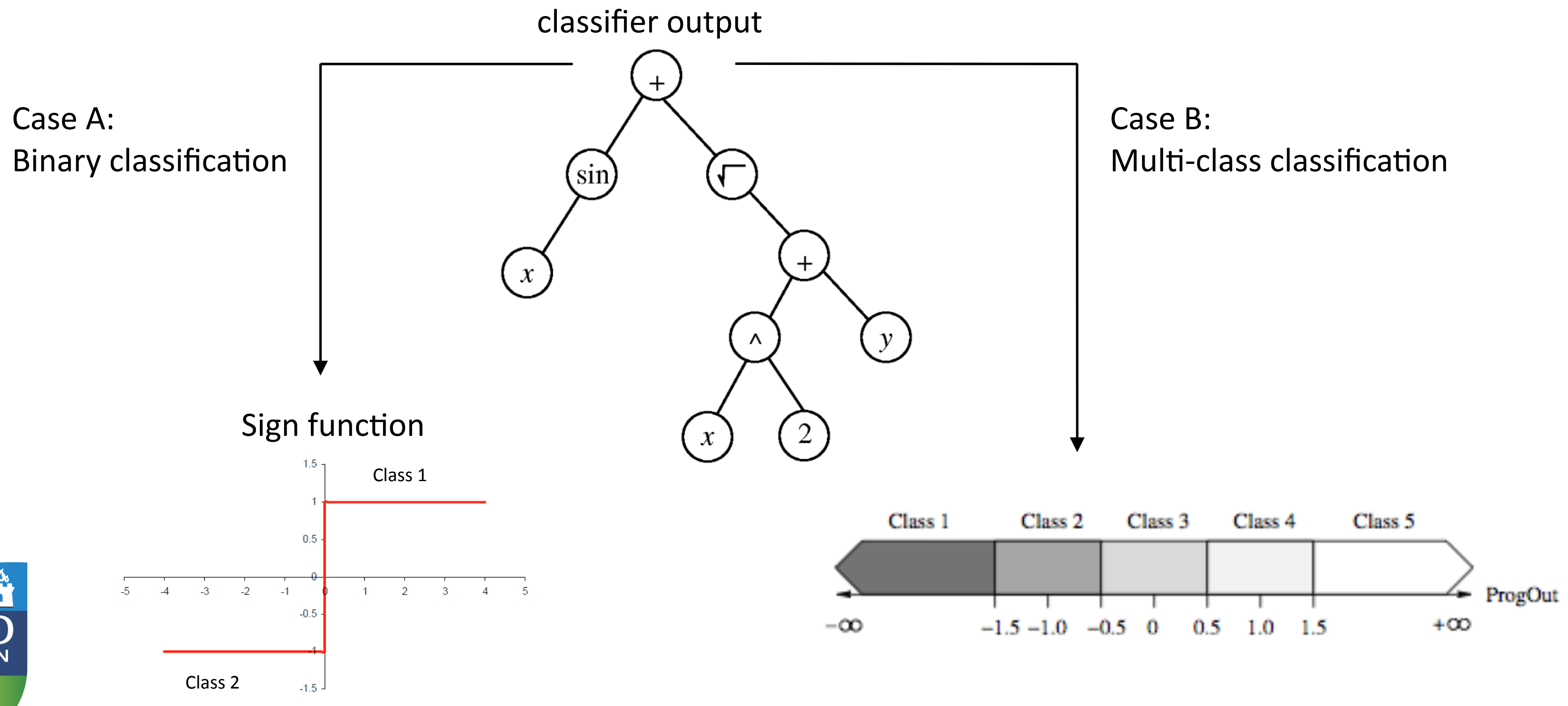
- We need a strongly-typed (tree-root type: int) system or a grammar.
- Function set:
  1. **If-then-else**. arg types: (boolean, int, int). return type: boolean
  2. **logic operators** <, >, =. arg types: (double, double). return type: boolean
  3. **arithmetic operators**: +, -, *. arg types: (double, double). return type: double
- Terminal set:
  1. classification output (type int),
  2. features (type double)
  3. constants (type double)





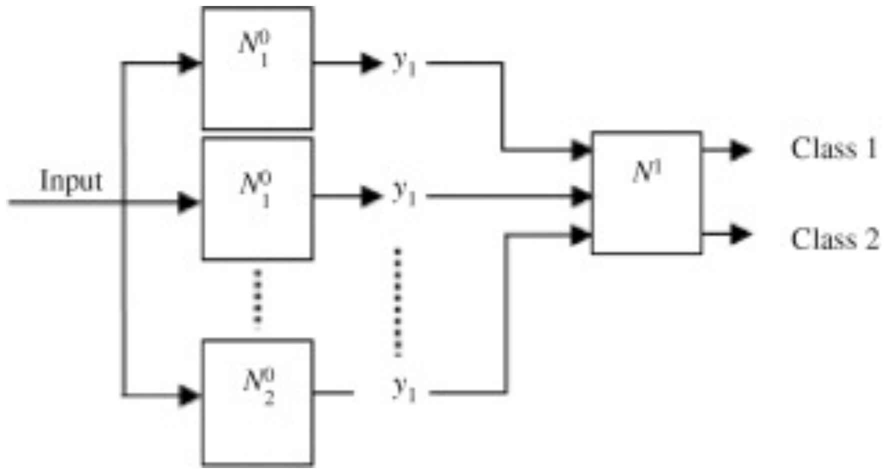
Real-valued classifiers

- Classifier is represented by a real-valued function.
- Function set: arithmetic operators (i.e. +, -, *, sin, etc.)
- Terminal set: features, constants
- Fitness function: classification accuracy



Constructing ensembles of real-valued classifiers

- The goal is to train a GP expression-tree to combine the predictions of several other learning algorithms.
  1. Several partitions of the original training data are formed and base classifiers are obtained from these datasets by means of different learning algorithms, i.e., ANNs, SVMs, RBFNs, etc.
  2. A GP expression-tree is evolved using the classifier-outputs as features.
  3. Function set: arithmetic operations, logic operations.
  4. Terminal set: classifier-outputs, constants



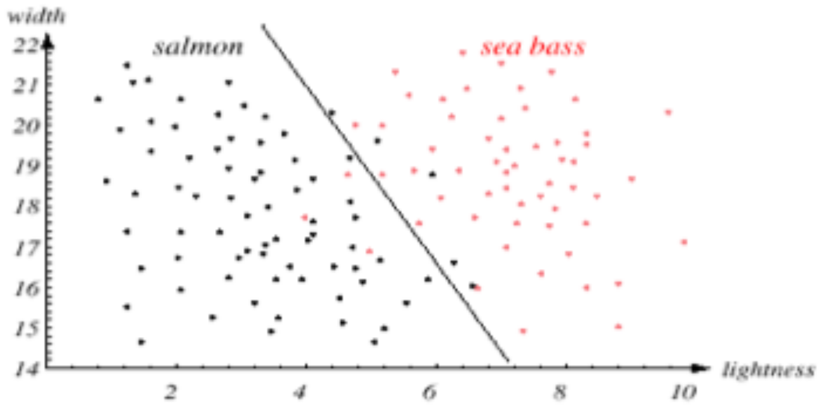
Classifier generalisation

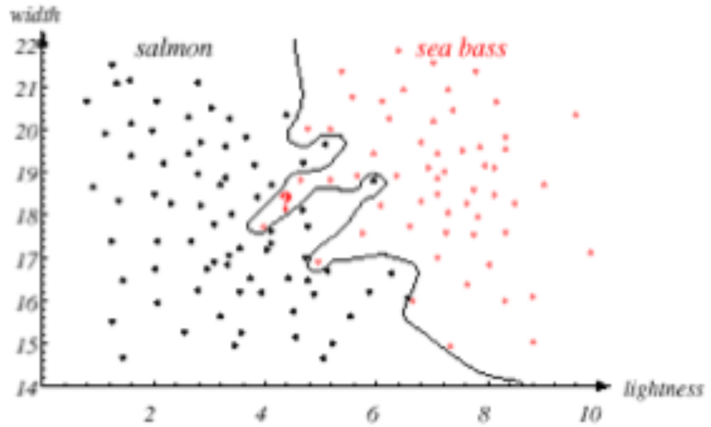
"I'm not superstitious either, but those were the three days Harris wore his lucky socks."

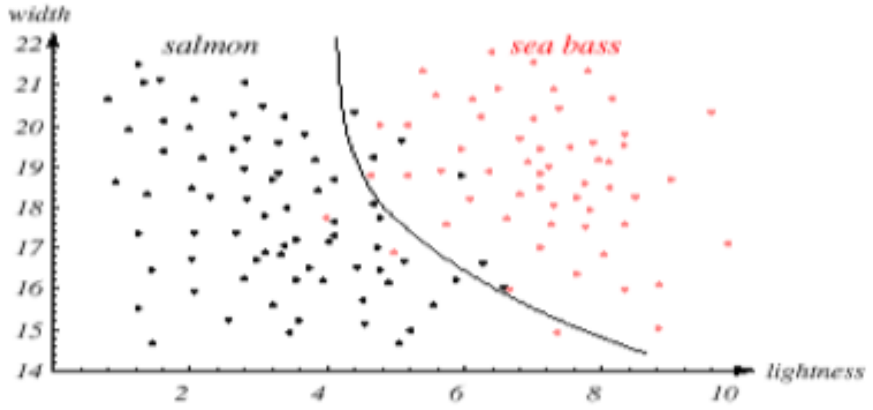
Generalisation is the ability of a classifier to perform accurately on unseen data (i.e. data that has not been presented during training).

Model selection

- Given *data* $x_n = x_1, x_2, \ldots, x_n$
  and *models* $M_1, M_2, M_3, \ldots, M_k$
  which model best explains the data?
- Need to take into account :
  1. Goodness of fit
  2. Complexity of models
- Methods:
  1. Statistical hypothesis testing
  2. Bayesian Model Comparison
  3. Minimum Description Length principle
  4. Validation dataset

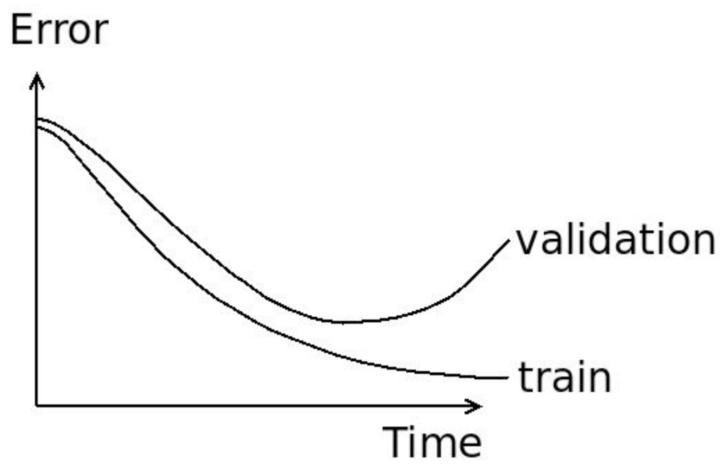






Use of a validation dataset for model selection

- Divide the data set into three sets:
  - The **training** set is used to fit the classifier.
  - The **validation** set is used for classifier selection.
  - The **test** set is used to assess the generalisation error of the chosen classifier.
  - Use proportions of 50%-25%-25%.



Summary

- GP has successfully tackled a wide range of pattern classification tasks.
- Dominant program representations for evolving classifiers are decision-trees for approximating discrete-valued functions, and standard arithmetic-based expression tress for approximating real-valued functions.
- Fitness functions are generally based on classification accuracy.
- GP implicitly performs feature extraction/selection to optimise the discrimination capability of the classifier.
- Classifier generalisation is addressed via the use of a validation dataset.

Further reading

- Duda R., Hart P., Stork D., ***Pattern classification***, John Wiley and Sons, Second edition, 2001.
- Sherrah J., Bogner R. Bouzerdoum B., ***A survey on the application of Genetic Programming to classification***, IEEE Transactions of Systems, Man, and Cybernetics, Part C: Applications and Reviews, 40(2), pages 121-144, March 2010.
- Agapitos A., Dyson M, Lucas S., Sepulveda F., ***Learning to recognise mental activities: Genetic programming of stateful classifiers for brain-computer interfacing***. In proceedings of GECCO 2008, Atlanta, USA.
- Theodoridis T., Agapitos A., Hu H., Lucas S., ***A QA-TSK fuzzy model Vs evolutionary decision trees towards nonlinear action recognition***, In proceedings of IEEE International Conference on Information and Automation, June 20-23, 2010, Harbin, China.
- Jabeen H., Baig A., ***Review of classification using genetic programming***, International journal of engineering science and technology, 2(2), pages 94-103, February 2010.
- Loveard T., Ciesie V., ***Representing classification problems in genetic programming***, In proceedings of IEEE Congress on Evolutionary Computation, Vol. 2, pages 1070-1077, May 2001.
- Ludmila Kuncheva, **Combining pattern classifiers: methods and algorithms**, Wiley-Blackwell, 2004.