Evolutionary feature selection and feature synthesis for medical diagnosing

MIBISOC Second Technical Course in Medical Imaging using Bio-Inspired and Soft Computing

Krzysztof Krawiec

Laboratory of Intelligent Decision Support Systems & Computational Intelligence Group
Institute of Computing Science, Poznan University of Technology, Poznań, Poland
http://www.cs.put.poznan.pl/kkrawiec/
krawiec@cs.put.poznan.pl

Parma, Feb 25, 2012

The outline

The topic: Evolutionary computation for selection, weighting, and synthesis of features, in particular features extracted from medical data, mostly form medical imaging.

The outline:

1. The machine learning context: Transformation of feature space.
2. Evolutionary feature selection, weighing, and synthesis, including:
   - Basic concepts of evolutionary computation, and selected variants of it, in particular to genetic algorithms and genetic programming.
1. Case study 1: Diagnosing of CNS tumors based on microscopic images of histological sections.
2. Case study 2: Diagnosing of ADHD based on clinical and MRI data.
3. Case study 3: Feature synthesis for object detection.

The pros and cons of evolutionary techniques compared to other approaches will be discussed, followed by practical recommendations.

Note:

- Some examples will come from outside of medical imaging.
- Slides available at `http://www.cs.put.poznan.pl/kkrawiec/wiki/?n=Site.MIBISOC`

The outline

Introduction

The broad perspective

Two alternative approaches to design of intelligent systems:

- *knowledge-driven*: the system is being built based on explicit knowledge, typically elicited from a human expert,
- *data-driven*: the system is being built by discovering and modeling patterns observed in data,

Types of intelligent systems in terms of transparency:

- *whitebox*: system's knowledge is human-readable (explicit) and can be interpreted,
  - or, at least, the line reasoning for particular case can be traced,
- *blackbox*: only the final result of reasoning (output) is available,

Note that:

- Both knowledge- and data-driven approaches can produce whiteboxes and blackboxes.
- Frequent practice: *greybox*.
- Low (albeit growing) tolerance for blackboxes in medicine.

The narrower perspective

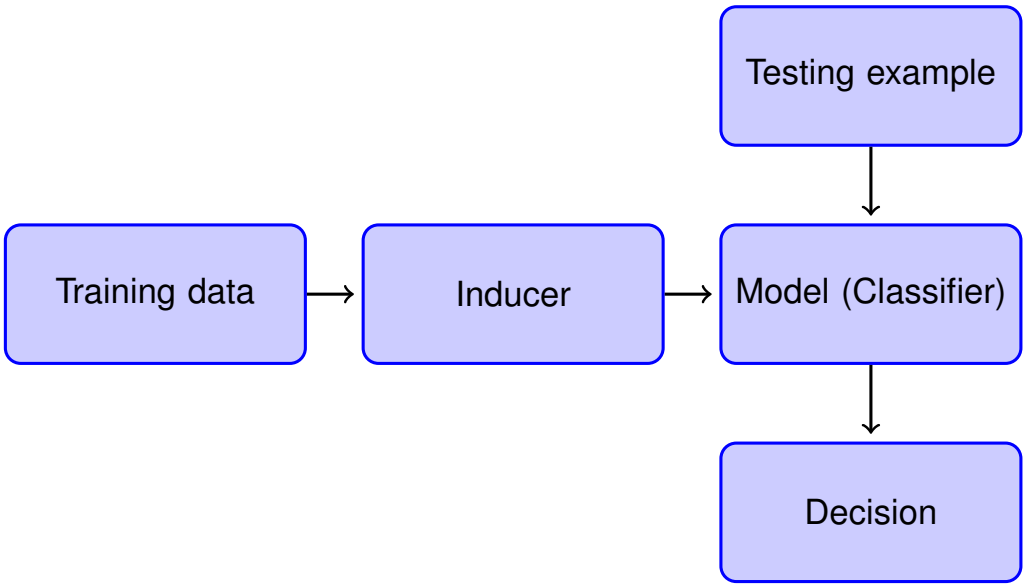
This lecture is about using evolutionary techniques (EC) together with machine learning (ML) to build grey- and black-boxes in a data-driven way.

From medical viewpoint: using EC+ML techniques to design:

- Clinical Decision Support Systems (CDSS)
- Computer Assisted Diagnosing (CAD)

Machine learning perspective

- The data come in the form of *examples* (*instances*, e.g., medical cases).
- An *inductive algorithm* (learning algorithm, learner, inducer) is available that can produce a *model*, which can be a *classifier* or *regression machine*
  - Learning algorithm $\neq$ classifier
  - We are primarily interested in supervised learning (examples are labelled) and classification.
- The model should generalize well, so we verify it on a separate subset of *testing examples* (unlabeled instances).
- Examples are described in terms of *attributes* (*variables*, *features*)



The concept of feature

Feature definition

What is a feature?

- A descriptor derivable from data (examples),
- Similar to: *attribute* (in ML), *independent variable* (in statistics), but:
  - Attribute is given,
  - Feature can be given or calculated from attributes,
  - Thus, attribute is a special case of feature; SetOfAttributes $\subseteq$ SetOfFeatures
- Can be defined on different *scales*: nominal, ordinal, metric (interval or rational).
- Typically, one needs more than one feature to build a well-performing system.

Note:

- Feature is a broad concept. Extreme examples:
  - The brightness of an image pixel of specific coordinates (very low information content),
  - System's decision (very high information content),
- Typically, we want features of intermediate information content.
- Attributes (variables) $\longrightarrow$ Features $\longrightarrow$ Decision

Transformation of representation

- Attributes form the [original] **representation** of data.
- There is often a need to **transform** the original representation.
- Why?

Motivation 1

There can be too many original attributes.
Talking in terms of feature selection:

- Typically, only some of the original features are relevant for the considered task.
- The remaining features are usually:
  - *redundant* = their values depend on the values of other features,
  - *irrelevant* = noise from the viewpoint of the diagnostic problem in question.

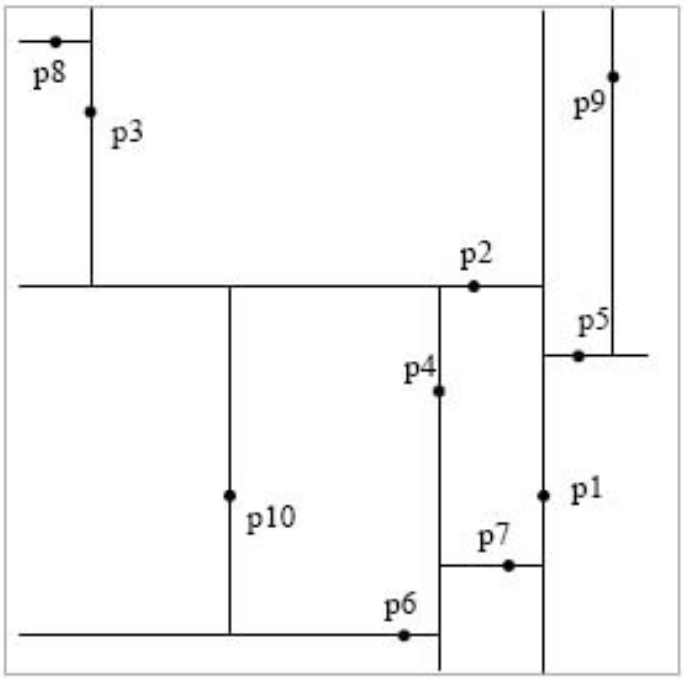
Why are such features inconvenient?

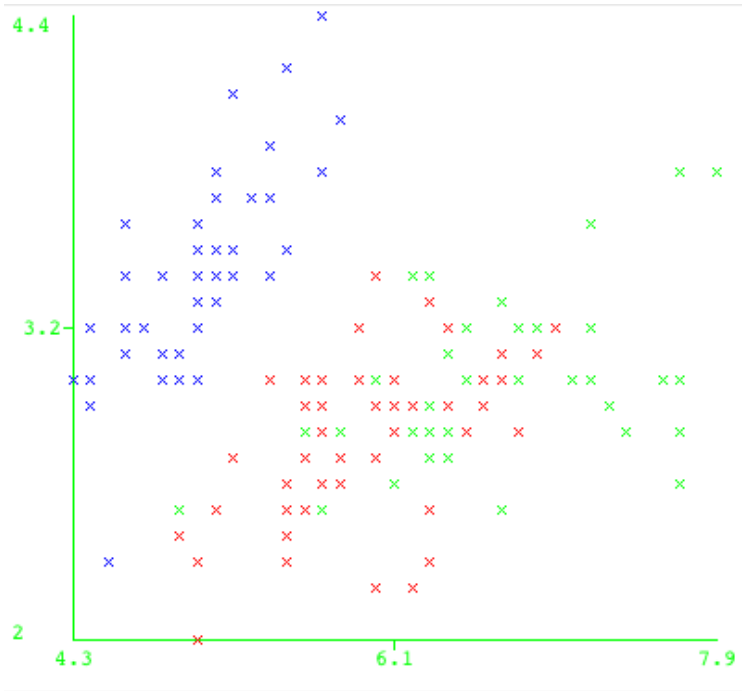
- Overly high computational cost of training and querying.
- Irrelevant features make it difficult for a classifier to discover the useful features.
  - Some inducers have to use all features (e.g., kNN), and they consider all features equally important.
  - Some inducers (e.g., decision trees) can overly focus on attributes that appear to be good when considered in isolation, but turn out to be inferior to *subsets* of other features.
- Redundant features are mutually highly correlated – some classifiers do not like that (e.g., the naive Bayesian classifier tends to overestimate the importance of correlated attributes).

Motivation 2

- Inducers tend to ‘prefer’ some data representations to others (*representation bias*)
- Some patterns in the data may be hard or impossible to capture using the original representation.

- Example: [most] tree induction algorithms analyze each attribute independently, and introduce orthogonal partitioning of attribute space.



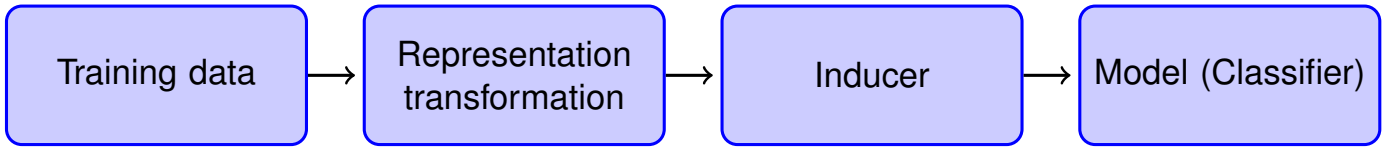


The techniques

Representation transformations to be considered here:

- Feature *selection* (FS),
- Feature *weighing* (FW),
- Feature *construction* (FC), or feature *synthesis*.

In all cases, feature transformation becomes a part of learning process.



Feature selection

Feature selection - problem statement

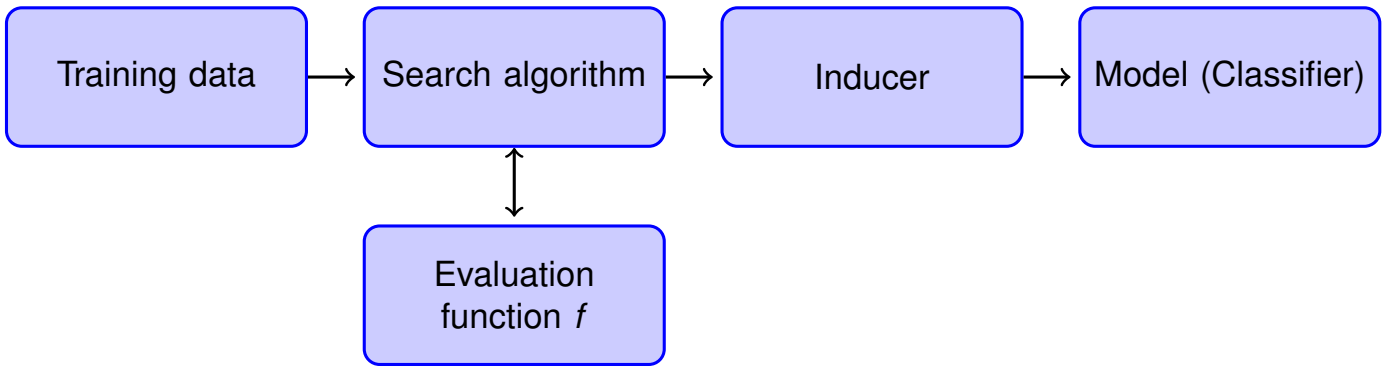
- The goal of feature selection (FS) is to get rid of redundant and irrelevant features.
- Formulation:
  Given a training set described by a set of features F, find a subset of features $F' \subseteq F$ that results in the highest classification accuracy on the testing set.
  - Ill-posed: we have no access to the testing set.
  - In practice: 'gives the highest value of some evaluation function'.
- FS is a **search problem**, where each state in the search space represents a subset of features.[1].

[1] D.W. Aha and R.L. Bankert, Feature Selection for Case-based Classification of Cloud Types: An Empirical Comparison, in: Proc. AAAI-94 Workshop Case-Based Reasoning, 1994, 106-112.

Feature selection - problem statement

The basic components of a general FS method:

- A search algorithm - looks through the space of feature subsets;
- An evaluation function f - used to evaluate the subsets of features.



Why is FS task nontrivial?

- Computational complexity. There are $2^{|F|}$ subsets of the feature set F,
  - An exact algorithm searching through all subsets has an exponential complexity.
  - Heuristics are necessary to reduce the number of considered subsets.
- High computational cost of some evaluation functions
- Open world assumption.
  - The FS algorithm has to take into account not only the descriptive properties of a particular feature set (related to the training set), but also its inductive (predictive) ability, which is usually verified on the testing set.

Desired properties of evaluation functions

- Evaluation function should ‘correlate’ with the predictive accuracy of the system.
  - (Not necessarily correlate in the strict sense)
- Given that:
  - the complete set of features F embraces all available information, and
  - the empty set $\emptyset$ embraces no information,

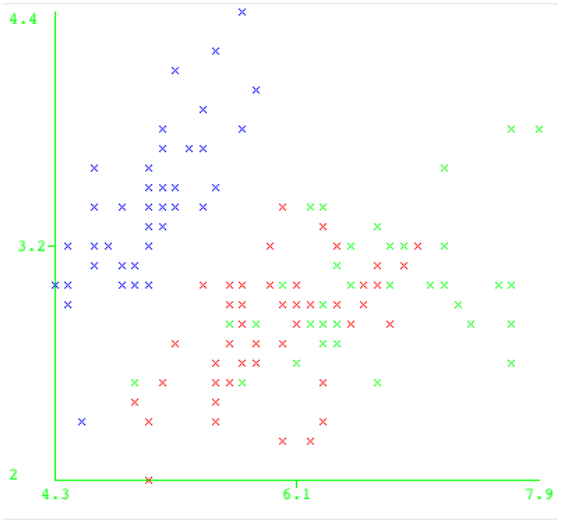
  evaluation should be [weakly] monotonous:

$$F_1 \subseteq F_2 \implies f(F_1) \leq f(F_2)$$

- This and other properties of evaluation measures are exploited by some FS methods.

The main line of division:

- Performance measures for *single attributes*
  - Examples: correlation, entropy.
  - Very limited use. Do not take into account the synergy between attributes.



- Performance measures for *attribute subsets*. Examples: correlation, [2] consistency.[3]
- The above are examples of *filter approach* to feature selection.

[2] Hall, M. A. (1998). Correlation-based Feature Subset Selection for Machine Learning. PhD Thesis, University of Waikato.

[3] Liu, H., and Setiono, R., (1996). A probabilistic approach to feature selection - A filter solution. In 13th International Conference on Machine Learning (ICML'96), July 1996, pp. 319-327. Bari, Italy.

Filter and wrapper model

Two canonic approaches to feature selection.[4]

- The **filter model**:
  - f is calculated using exclusively training data.
  - Features are selected (i.e. filtered) depending on properties of the data itself, independently from the chosen learning algorithm.
- The **wrapper model**,
  - f uses a classifier in the evaluation function.
  - Evaluation is usually done by estimating predictive accuracy in a cross-validation (CV) test.

[4] G. John, R. Kohavi, and K. Pfleger, Irrelevant features and the subset selection problem, in: Proceedings 11th International Machine Learning Conference (1994) 121-129.

The wrapper model – cross validation

For a given subset F':

1. Split the training data into n possibly equal subsets (folds).
2. Repeat n times, using only the features from F':
   - Train a classifier on $n - 1$ folds.
   - Test it on the remaining fold.

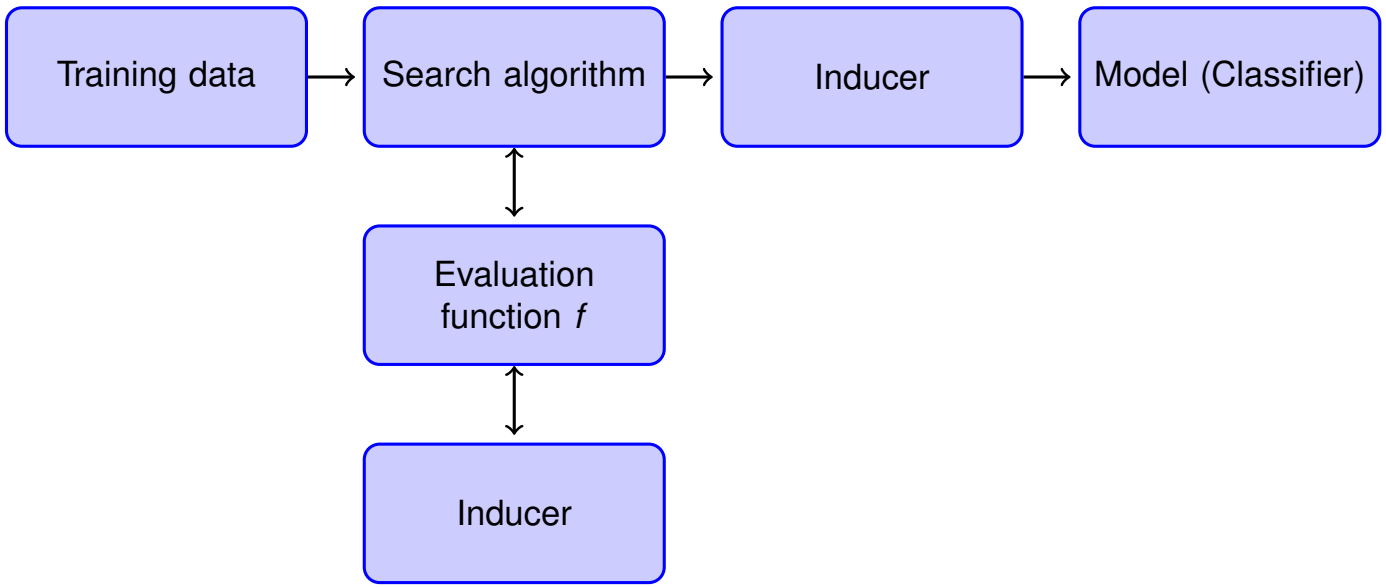
The resulting accuracy of classification becomes the evaluation $f(F')$

Which is better?

Wrapper approach is often reported to outperform other approaches.[5]
The reasons:

- It incorporates the inductive bias of the classifier into the evaluation function.
- The search process optimizes for the same measure that it is later used to evaluate its performance on the testing data.
- The inducer used in *f* can use additional 'bells and whistles', e.g., means to handle cost-sensitive classification, imbalanced decision classes, etc.

The price: high computational complexity.



[5] R. Kohavi and G.H. John, Wrappers for feature subset selection. Artificial Intelligence. Artificial Intelligence (1997) Volume: 97, Issue: 1-2, Publisher: Elsevier, Pages: 273-324

Search algorithms

Popular:

- *Forward selection*: start from an empty feature set and select features one by one, until stopping condition is met,
- *Backward elimination*: start the original set of features, and remove features one by one, until stopping condition is met.

Note:

- Both are *local* search algorithms.
- When the number of features is large, forward selection is often the only option (computational cost of f typically grows with the number of features).
- Hybrid and more sophisticated algorithms do exist (see, e.g., WEKA).

Software

Off-shelf ML frameworks that implement feature selection:

Waikato Environment for Knowledge Analysis (WEKA) by Eibe & Frank

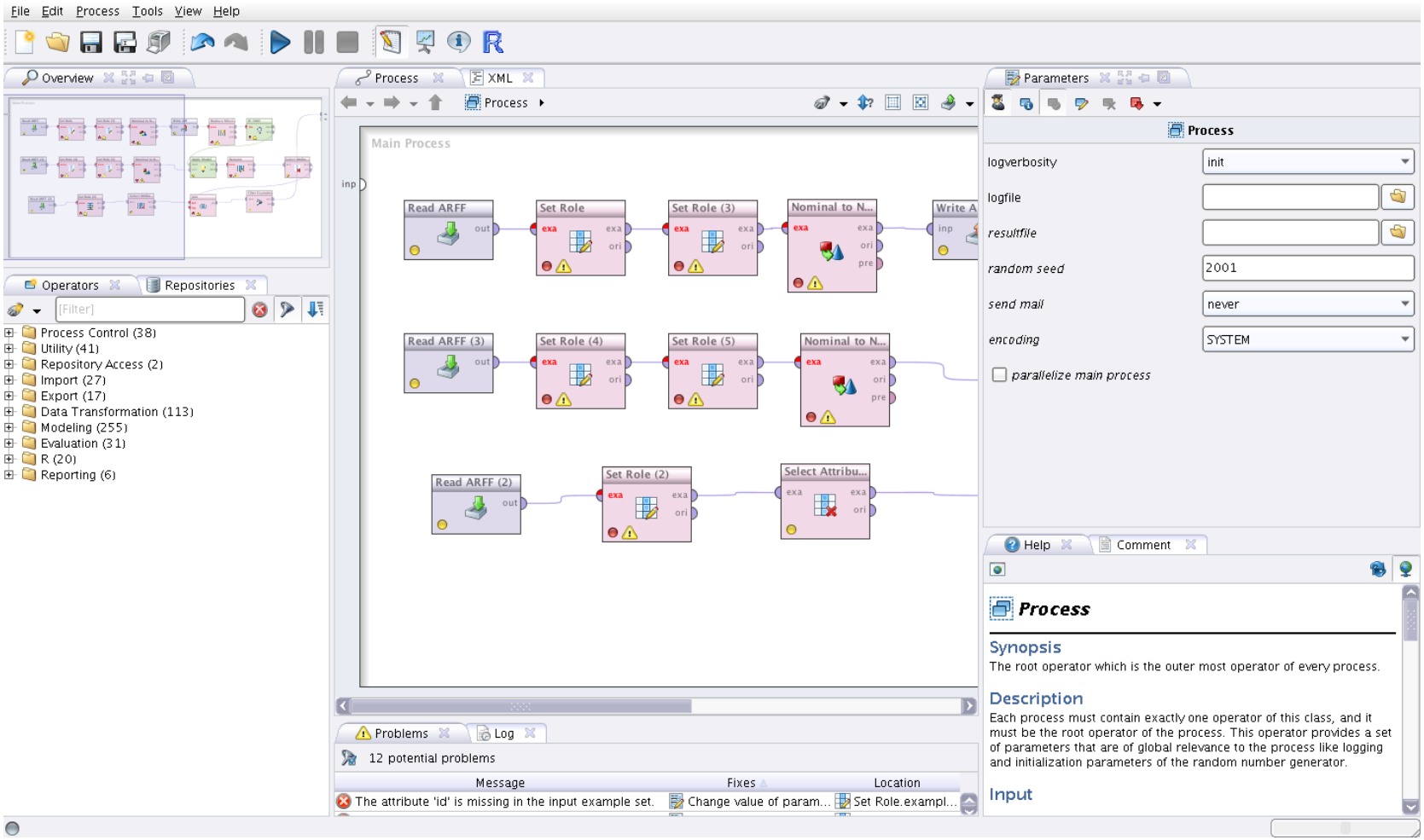
`http://www.cs.waikato.ac.nz/ml/weka/`

Rapid Miner by Rapid-I

`http://www.rapid-i.com/`

- Represents dataflows as XML files
- Cross-platform, nice GUI
- Versatile visualization
- Can integrate with WEKA and R

Demo 1: Simple feature selection



Remarks

- Feature selection is implicitly performed by many inducers.
  Examples:
  - Decision tree learning algorithms (e.g., CART, ID3, C4.5) do not have to use all features.
  - A process of neural net training can bring down the weights outgoing from an input neuron very close to zero, making it effectively absent.
- Feature selection is a variant of **constructive induction** – an approach to supporting automatic, problem oriented transformation of representation space to facilitate learning [6] [7] [8] [9]

[6] R.S. Michalski, Pattern recognition as knowledge-guided computer induction, Departament of Computer Science Reports, No. 927, University of Illinois, 1978.

[7] C.J. Matheus, The need for constructive induction, in: Proceedings 8th International Workshop on Machine Learning.

[8] C.J. Matheus and L. Rendell, Constructive Induction on Decision Trees, Proceedings of IJCAI-89 (Detroit, 1989) 645-650.

[9] J. Wnek and R.S. Michalski, Hypothesis-driven constructive induction in AQ17: a method and experiments, in: Proceedings of IJCAI-91, Workshop on Evaluating and Changing Representation in Machine Learning, Sydney, 1991.

Evolutionary feature selection

Evolutionary feature selection

The idea: use evolutionary algorithm to search the space of feature subsets.

Evolutionary computation in a nutshell:

- A family of global, stochastic, bio-inspired search algorithms designed according to neodarwinian theories of natural selection.
- Two necessary elements:
  - Variation
  - Selection
- Other (optional) elements:
  - Recombination operator (crossover)
- A multitude of variants: genetic algorithms, evolutionary algorithms, evolutionary strategies, genetic programming, ...

Generic Evolutionary Algorithm

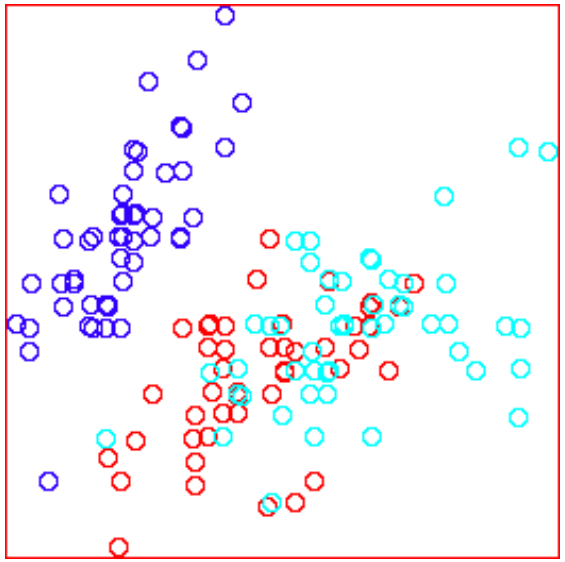
```
1: function EvolutionaryAlgorithm(f)                    ▷ f - fitness function
2:     P ← {p ← RandomSolution()}                       ▷ Initialize population
3:     repeat                                           ▷ Main loop over generations
4:         for p ∈ P do                                 ▷ Evaluation
5:             p.f ← f(p)                               ▷ p.f stores the fitness of p
6:         end for
7:         P' ← ∅                                       ▷ Next population
8:         repeat                                       ▷ Breeding loop
9:             p1 ← Selection(P)                        ▷ First parent
10:            p2 ← Selection(P)                        ▷ Second parent
11:            (o1, o2) ← Crossover(p1, p2)
12:            o1 ← Mutation(o1)
13:            o2 ← Mutation(o2)
14:            P' ← P' ∪ {o1, o2}
15:        until |P'| = |P|
16:        P ← P'
17:    until StoppingCondition(P)
18:    return arg max_{p∈P} p.f
19: end function
```

- f - the fitness function that defines the objective of optimization process,
- RandomSolution() produces a random solution.

Feature transformation using EC

Why?

- Feature selection is an NP-complete task, so heuristic algorithms (local or global) have to be used.
- Local search algorithms explore only tiny portion of the search space.
- Greedy search easily ends up with a sub-optimal set of features.
- Natural 'compatibility' (see next slides).
- More likely to discover synergy between features.
  - Features are not selected on the one by-one basis, but get selected 'in parallel', respecting mutual dependencies and regarding relationships between attributes' weights.



Evolutionary feature selection

- Simplest variant: use genetic algorithm (GA) with binary encoding of solutions.
- Each individual represents a subset of features.
- Genotype is a binary vector of length $|F|$:
  - 1 – inclusion of a feature,
  - 0 – omission of a feature.
- Fitness = evaluation function f (filter or wrapper, typically wrapper).

Genetic operators for feature selection

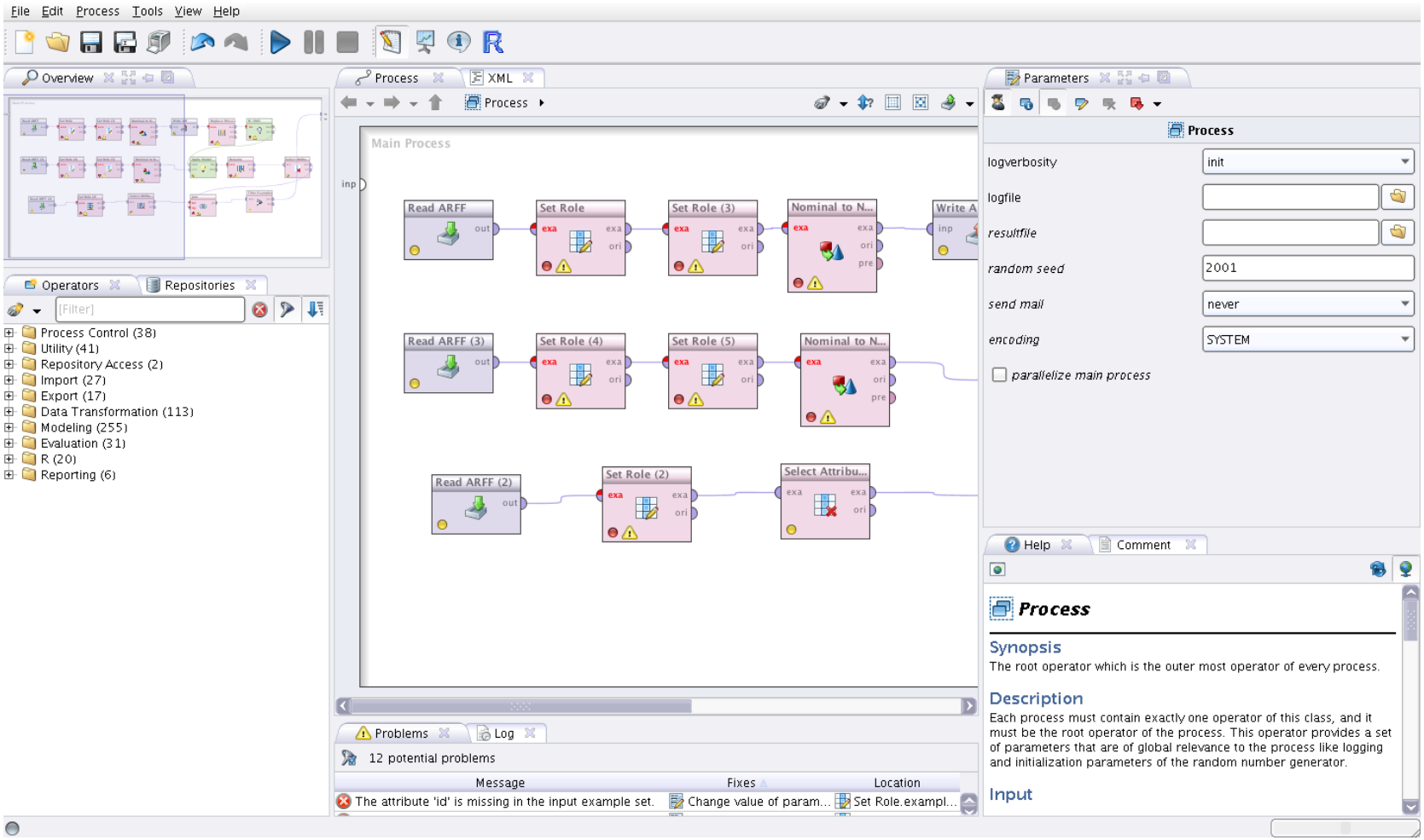
Crossover ‘mixes’ subsets of features. For instance, uniform crossover:

```
1: procedure CROSSOVER(p_1, p_2)
2:     for i = 1 ... n_features do
3:         if RND(p_swap) then
4:             o_1[i] ← p_1[i]
5:             o_2[i] ← p_2[i]
6:         else
7:             o_1[i] ← p_2[i]
8:             o_2[i] ← p_1[i]
9:         end if
10:    end for
11:    return (o_1, o_2)
12: end procedure
```

Mutation removes or adds a single feature:

```
1: procedure MUTATION(p)
2:     for i = 1 ... n_features do
3:         if RND(p_mutation) then
4:             p[i] ← ¬p[i]
5:         end if
6:     end for
7:     return p
8: end procedure
```

Demo 2: Evolutionary feature selection



Discussion

Some important settings:

- Initial probability of a gene being switched on; determines the expected size of initial subset (individual).
- Some learning algorithms require the features to be normalized.
- The internal cross-validation division should be fixed within a generation to make individuals' fitnesses comparable,
- ... but can vary between generations.

Discussion

When the only evaluation criterion is the predictive accuracy of individuals, there is no pressure to reduce the number of irrelevant features.

- Only features that *decrease* the accuracy by 'disturbing' the training process will be excluded.
- The features which are irrelevant and do not affect the accuracy will be selected or not on a random basis.

Conclusion:

- Use not-too clever classifier (e.g., kNN), or
- Introduce additional mechanism to keep the number of features low.

Discussion

Variants and extensions:

- Penalization for too large feature sets. For instance:
  - Explicitly, in the selection process: resolve ties using length of feature definition (*lexicographic parsimony pressure*)
  - Implicitly: Penalty term in fitness function.
    - E.g., Minimum Description Length (MDL)
- Improved CV schemes, e.g., stratified CV (evens distribution of decision classes).
- Asymmetric mutation: makes it less likely to add a feature than to remove it $\Longrightarrow$ bias towards smaller feature sets.
- Other fitness definitions for problems with strongly imbalanced decision classes (e.g., F-measure, G-measure, area under ROC curve (AUC)).

Discussion

Advantages:

- Decent results in a reasonable computation time.
- 'Decent' meaning: often hard to beat by local search methods.
- No special requirements concerning evaluation function (fitness).

Disadvantage: computational complexity.

- Each individual requires internal train-and-test experiment (e.g. cross-validation).
- However, the number of CV folds can be low.

Evolutionary feature weighing

Feature weighing

- Motivation: feature inclusion/exclusion is a black-or-white decision.
- Some features can be expected to be more *important* than others.
- Idea: weigh the features, i.e., assign a scalar value to each feature.
- Genetic implementation: Individual = vector of *weights*.

How to interpret the weights?

A classifier has to be capable to handle the extra information (feature weights).

- Example: k-Nearest Neighbor (k-NN) classifier:
  - It explicitly measures the distance between pairs of examples.
  - Weights stretch or squeeze the attribute space along particular dimensions.
  - When a difference in a given attribute has a high weight (i.e. is multiplied by a high value), this attribute becomes important in the process of discriminating examples.

A few other algorithms exist that accept attribute weights (e.g., some rule induction algorithms).

Genetic operators

Crossover:

- uniform crossover (as in FS),
- another possibility average operator (geometric crossover).

Mutation:

- operator can be a simple “random” operator, or
- a “creep” one, or
- modified random (changing 0 to a random value or a non-zero value to zero, as used in the experiments described in this paper).

- The weight-adjusting task is more general than feature selection.
- In fact, feature selection is a special case of the weighting task, where weights can be zero or one only [41].
- Cardinality of the search space for feature selection is $2^{|F|}$, and for weight adjusting it is $w^{|F|}$, where w is the number of possible weight values, and F denotes the set of features.

Case study 1: Evolutionary feature selection and weighting

Case study: Diagnosing CNS tumours

Based on:
Maciej Komosiński and Krzysztof Krawiec, Evolutionary weighting of image features for diagnosing of CNS tumors, *Artificial Intelligence in Medicine*, (19)1, 2000, pp. 25-38.

Note:

- Lots of other studies available; see, e.g., the review in [10].
- This case study is slightly more than 'canonic' FS/FW.

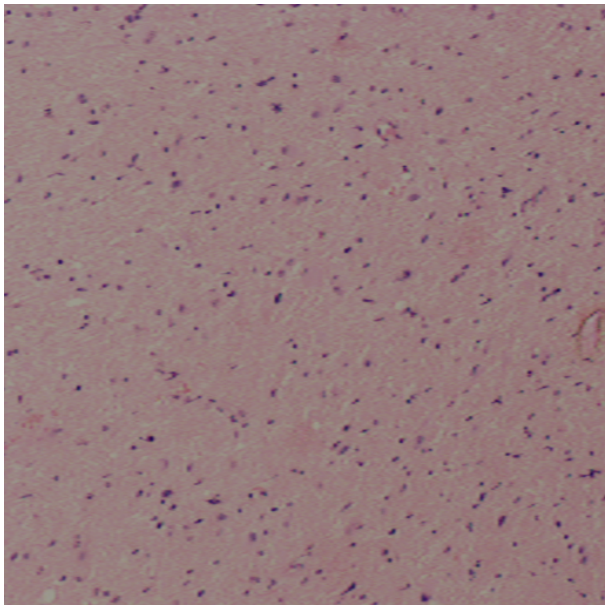
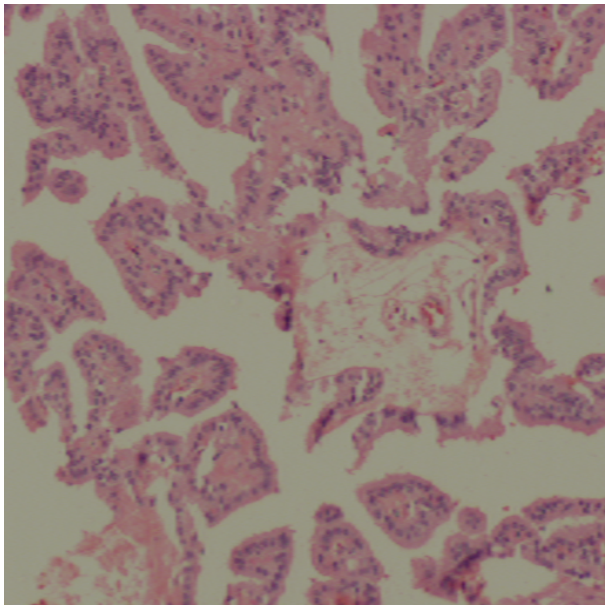
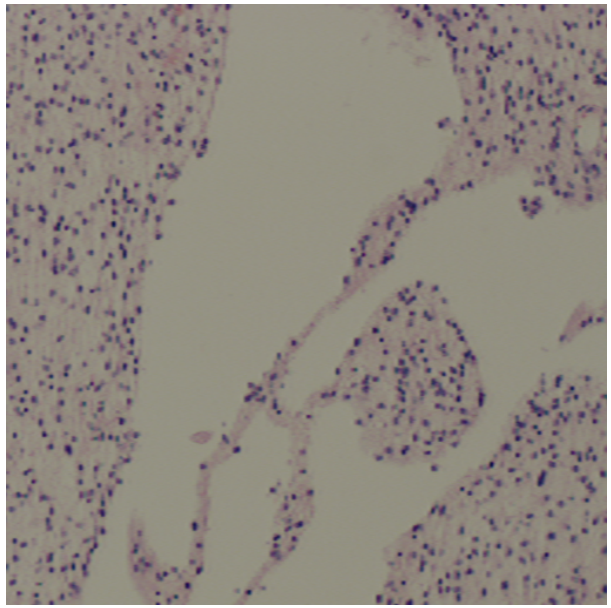
[10] Krzysztof Krawiec, Daniel Howard and Mengjie Zhang (2007) Overview of Object Detection and Image Analysis by Means of Genetic Programming Techniques. In Proceedings of Frontiers in the Convergence of Bioscience and Information Technologies 2007 (FBIT2007), Jeju, Korea, October 11-13, 2007.. IEEE CS Press, pages 779-784.

The data

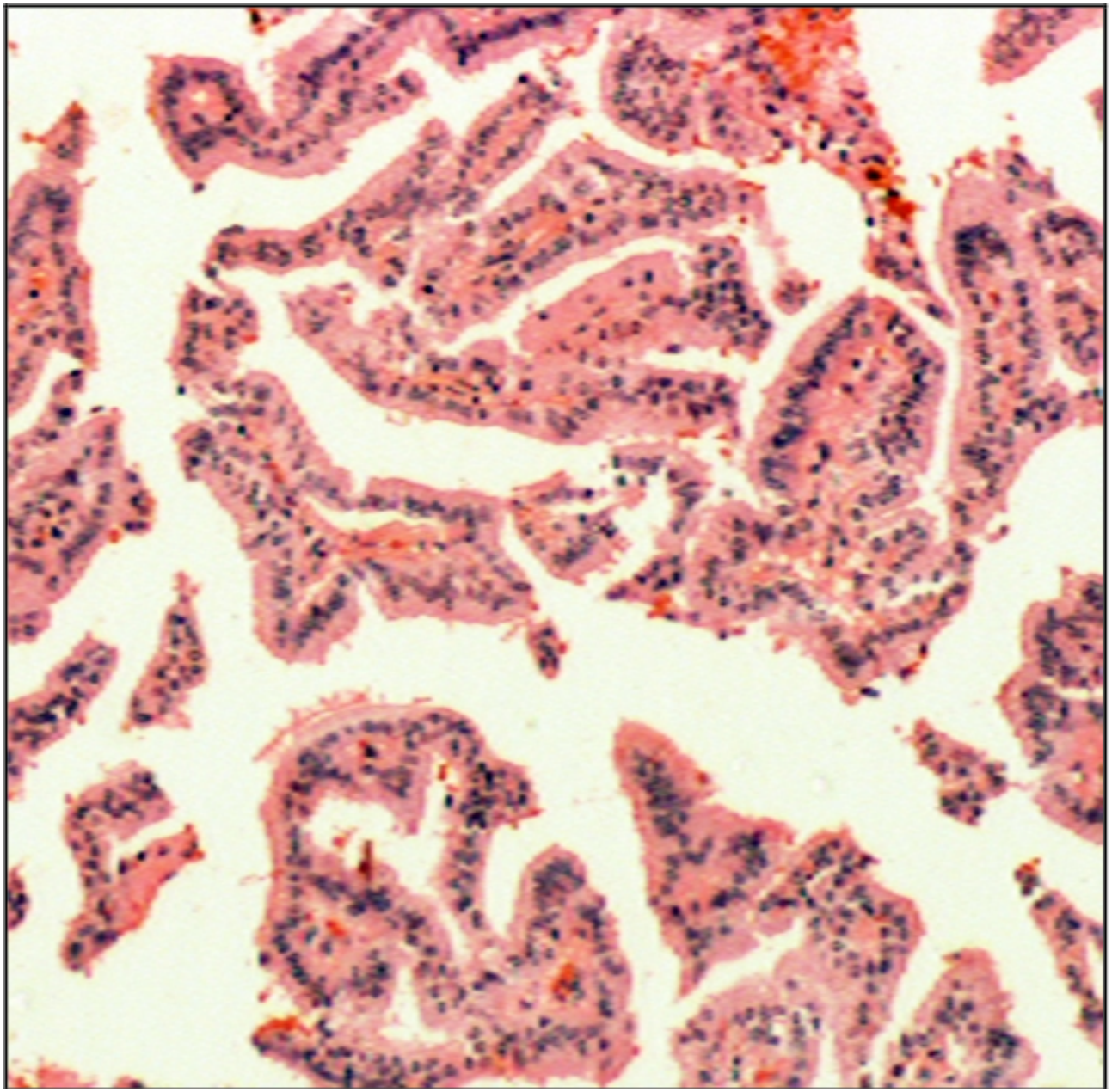
- Microscopic images of CNS tumors (neoplasms)
- 512×512 pixels
- 50 images per class (tumor type) $\Longrightarrow$ 600 images in total.
- Two subtasks:

A (Astrocytic tumours)	*B (Glial tumours)*
Astrocytoma anaplasticum	Astrocytoma fibrillare
Astrocytoma fibrillare	Oligodendroglioma isomorphum
Astrocytoma gemistocyticum	Ependymoma
Astrocytoma pilocyticum	Choroid plexus papilloma
Astrocytoma protoplasmaticum	Glioblastoma multiforme
Glioblastoma multiforme	Medulloblastoma

Example images

Example images



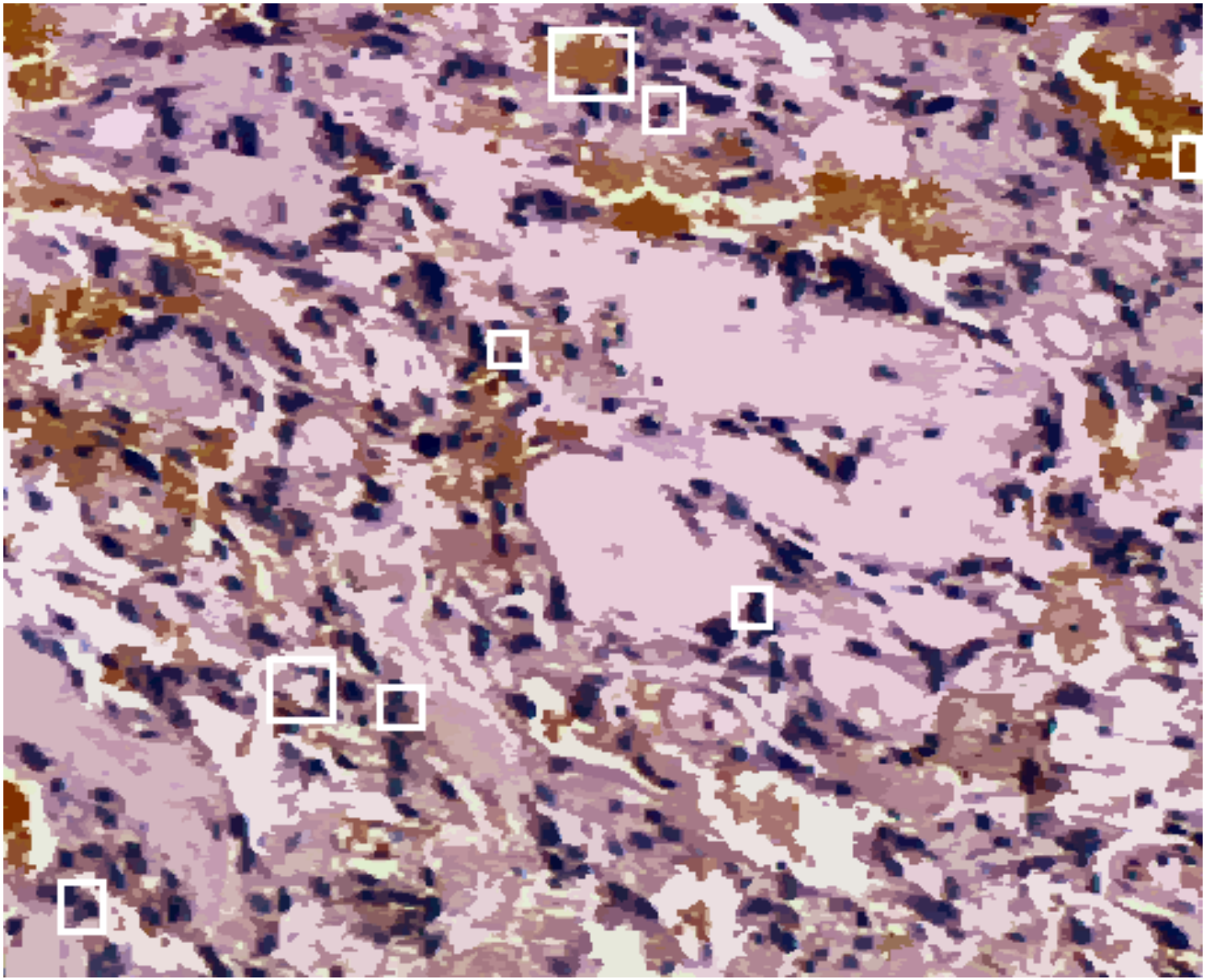
The approach

- The objective: classify the image to neoplasm class.
- Domain knowledge (expert's hint): frequency and spatial arrangement of different structures (e.g., cell nuclei) is important for diagnosing.
- In this study, we focus on frequency.

Image processing:

1. Preliminary filtering.
2. Image segmentation: region growing based on HSI color components),
   1. Results on average in ~13,000 regions per image.
   2. Number of regions varies from image to image.
3. The idea: Automatically define a set of types of regions, according to their mutual similarity.

Example image

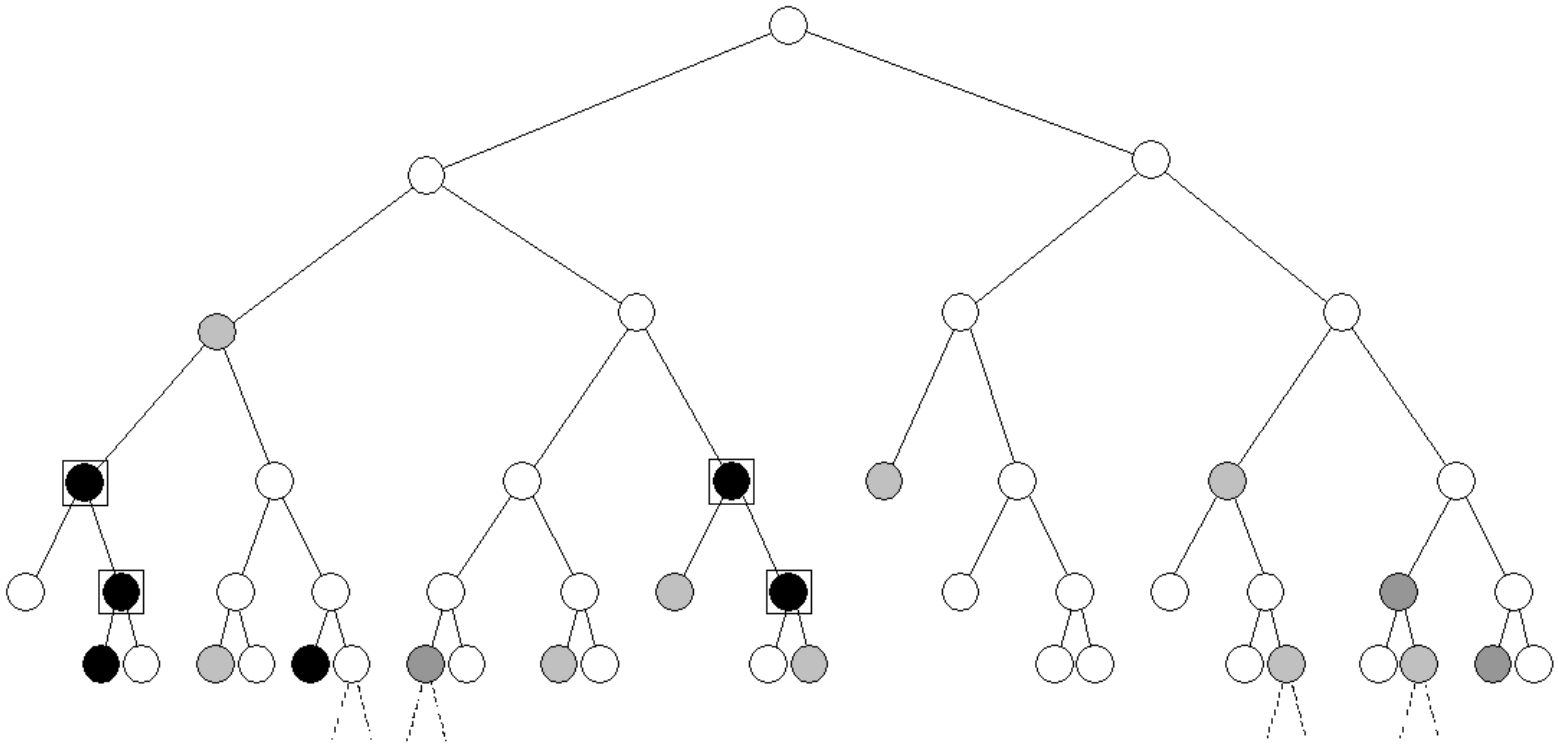


Learning region types

1. The base: Five images randomly selected from each decision class.
2. The result of image processing: a database of over 340,000 region records, each described by four features:
    1. region area and
    2. mean values of hue, saturation, and intensity.
3. Cluster analysis in the space of region features to find the centroids of the most characteristic regions: top-down, hierarchical clustering:
    1. Start from one cluster containing all considered objects.
    2. Recursively divide each cluster into two sub-clusters using the Principal Component Analysis, building in such a way a binary hierarchy (tree) of clusters.
        1. Use the best-first strategy: split the node with the highest intra-node variance at the moment.
    3. The process of partitioning continues until no node with variance greater than a predefined threshold can be found.

Feature definition

- The result: The hierarchy (tree) of cluster types composed of 69 nodes, 35 of which are leaves.



Feature definition

- Feature vector = the segmentation result for a particular image expressed in terms of the 69 categories of regions.
- For a given segmented input image, each tree node yields two attributes:
  - The absolute number of regions
  - The relative number of regions (fraction).
- The final feature vector consists of 69×2=138 attributes.

a_1	a_2	a_3	a_4	a_5	a_6	...	a_{70}	a_{71}	a_{72}	...	a_{138}
71	12	7	9	0	34	...	0.03	0.005	0.003	...	0.009

- We use evolutionary FS and FW to reduce these data.

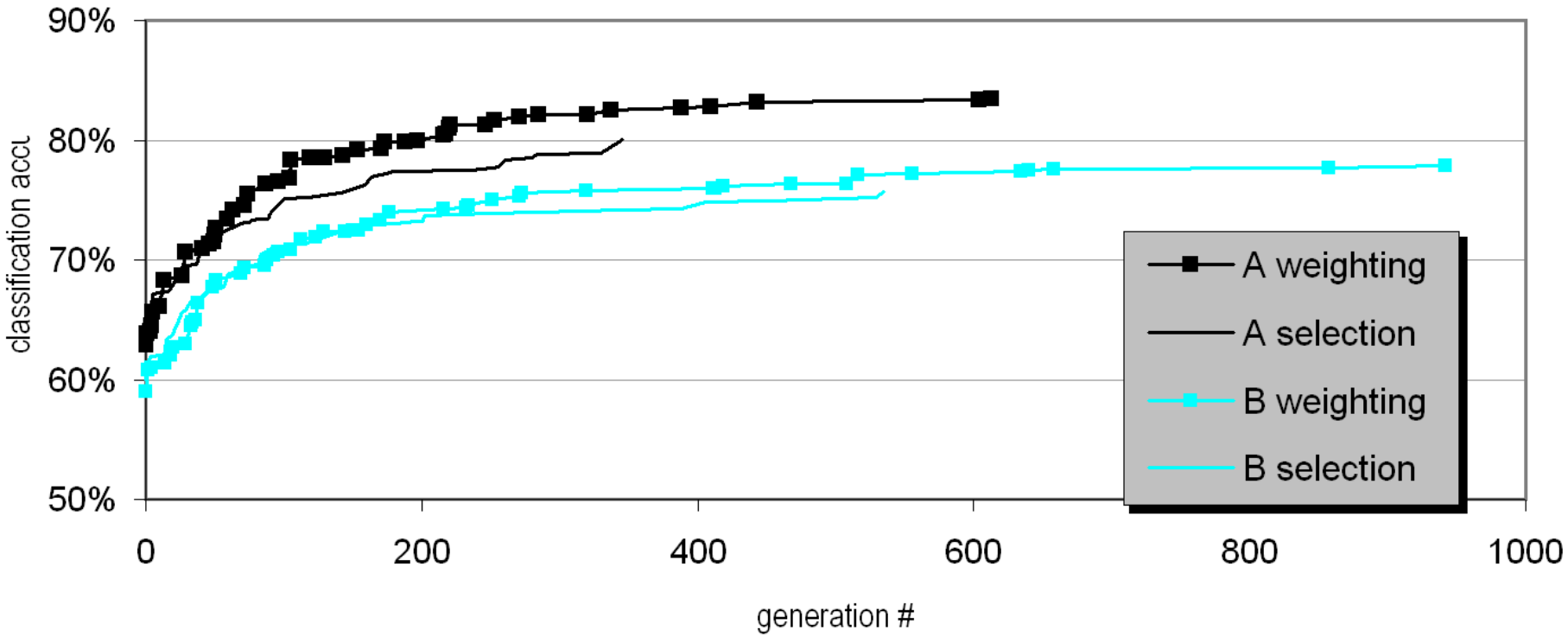
Setup: common settings

- Population size: 100 individuals
- Run length: max. 1000 generations, stopped after 200 non-improving generations.
- Tournament: Reminder with repetition method
- Fitness definition: wrapper based on 10-fold cross-validation using 1NN classifier.
- All features normalized prior to classification.
- Crossover: Uniform crossover, probability 0.8.
- Mutation probability: 1/(population size) = 0.01.
- Keeps track of the best individual found so far during the evolution. This is a standard off-line process.

The results

Data set	Classification accuracy [%]			# of features
	All features	Selection	Weighting	All / FS / FW
A: **astrocytic tumors**	57.50 ± 1.15	80.00 ± 1.22	83.43 ± 1.93	138 / 45 / 53
B: **glial tumors**	54.73 ± 2.32	75.70 ± 1.64	77.83 ± 2.03	138 / 49 / 62

Progress of evolution



Remarks: Representation of solutions

Individual encoding: vectors composed of:

- FS: 0s and 1s
- FW: integers $\in [0, 10]$
  - The advantage: neutral (non-effective) mutations less likely.

Mutation:

- FS: bit-flipping ($0 \rightarrow 1$ or $1 \rightarrow 0$).
- FW:
  - Set a gene to 0 when it was greater than 0,
  - Set it randomly to a $1 \ldots 9$ when it was 0.

The result: slight bias towards feature *selection*.

Remarks: Interpretation of weights

We allowed the weights to be positive or negative.

- Positive weight: the larger the difference in the attribute, the larger the dissimilarity of two considered examples.
- Negative weight: the larger the difference, the smaller the dissimilarity.
- Zero weight means that the corresponding attribute may be ignored (treated as irrelevant) during the computation of total similarity of examples.

Case study 2: Diagnosing of ADHD

The task

The ADHD-200 Global Competition.

- Timing: March – September 2011
- Objective: Based on **clinical** and **imaging data** made available by contest organizers, design a clinical decision support system for diagnosing three subtypes of ADHD.

The data

The data: Derived from dataset released by Functional Connectome consortium within subproject ADHD200:

- Patient population consisted of **285** children and adolescents (median age 11, range 7-21 years; 224 males) diagnosed with attention-deficit hyperactivity disorder (ADHD).
- Control population consist of **491** healthy individuals (median age 11.7, range 7-22; 231 females) not diagnosed previously with ADHD.
- Total: **776** cases.

Imaging protocol:

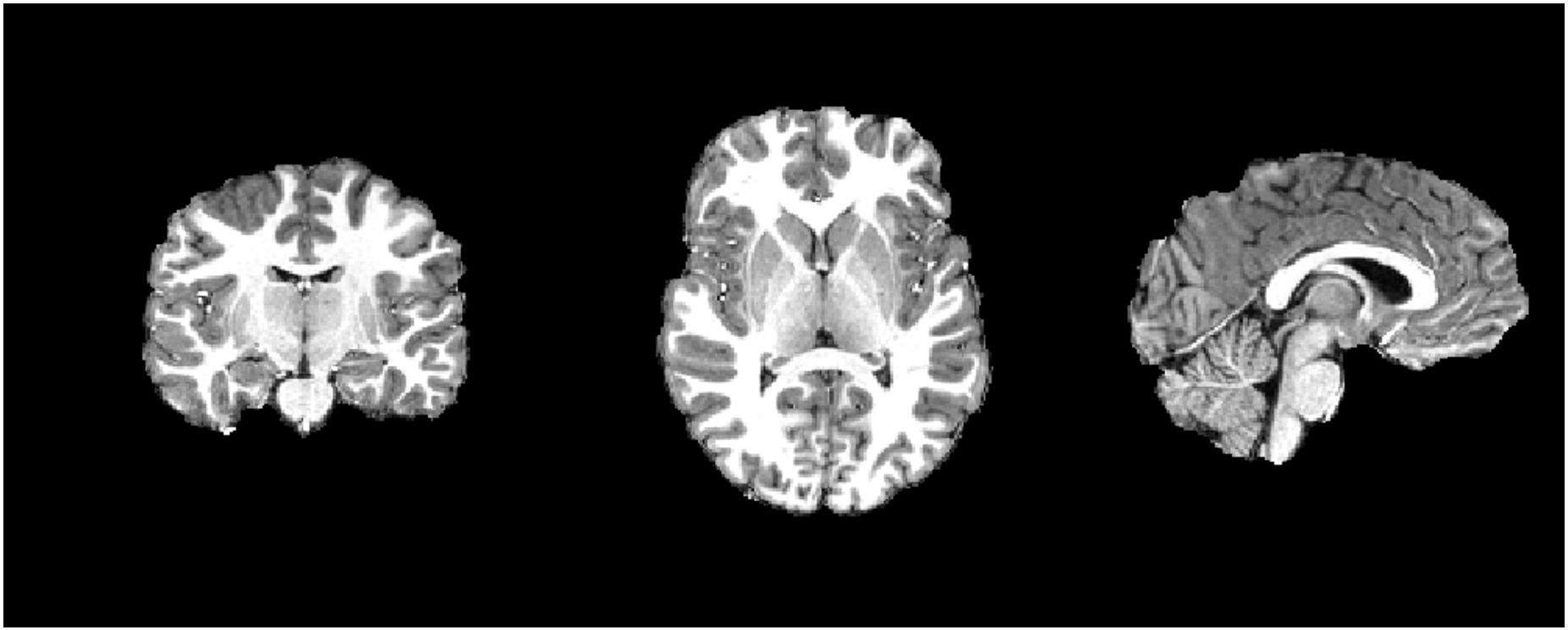
- structural: MPRAGE 3D volumetric T1-weighted scan,
- functional: resting state BOLD (blood oxygen level-dependent) acquisition, approx. 5-6 minutes.

The features

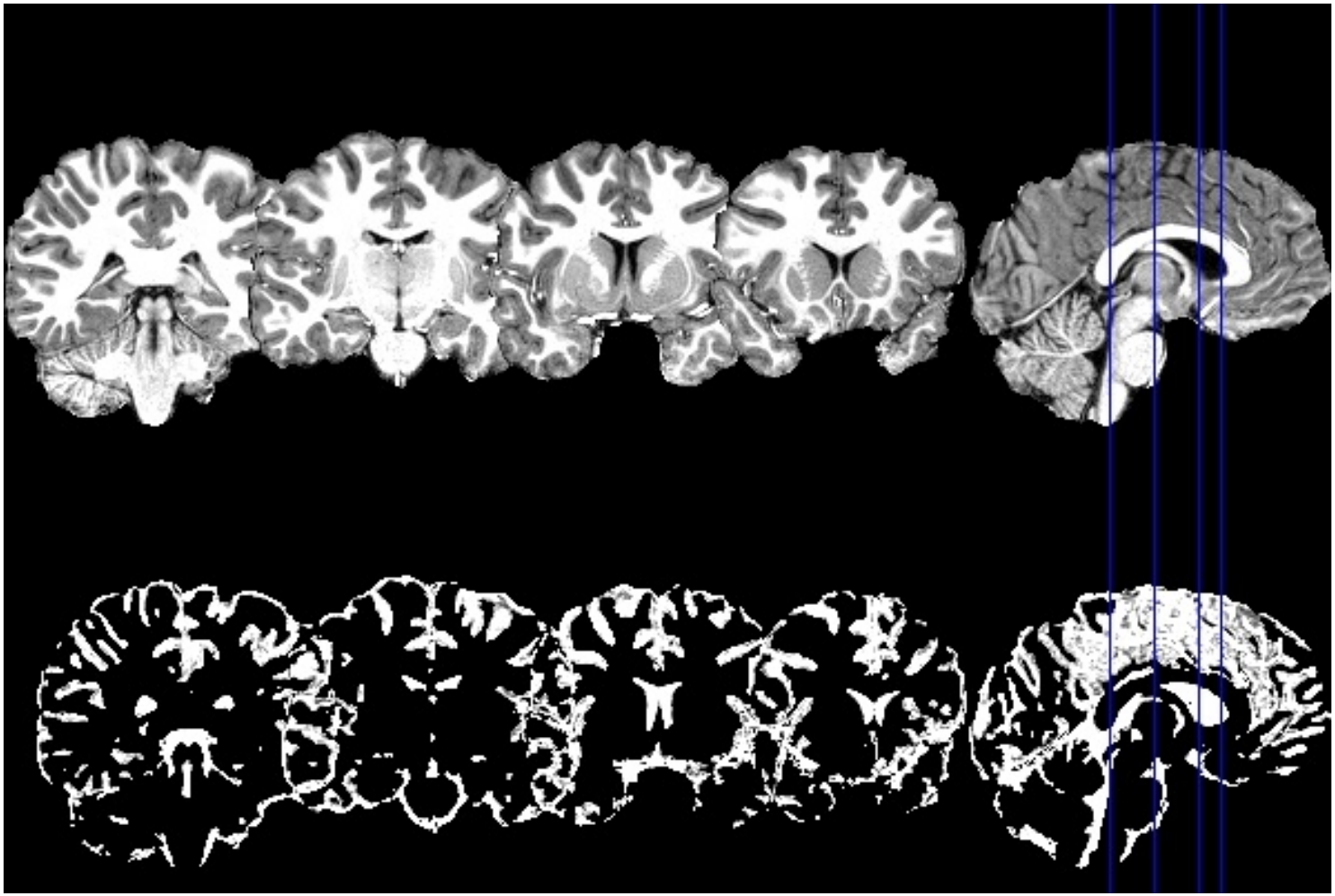
Three groups of features:

- clinical features (C),
- structural features (S), extracted from T1-weighted volumetric data,
- functional features (F), extracted from resting-state BOLD acquisition,

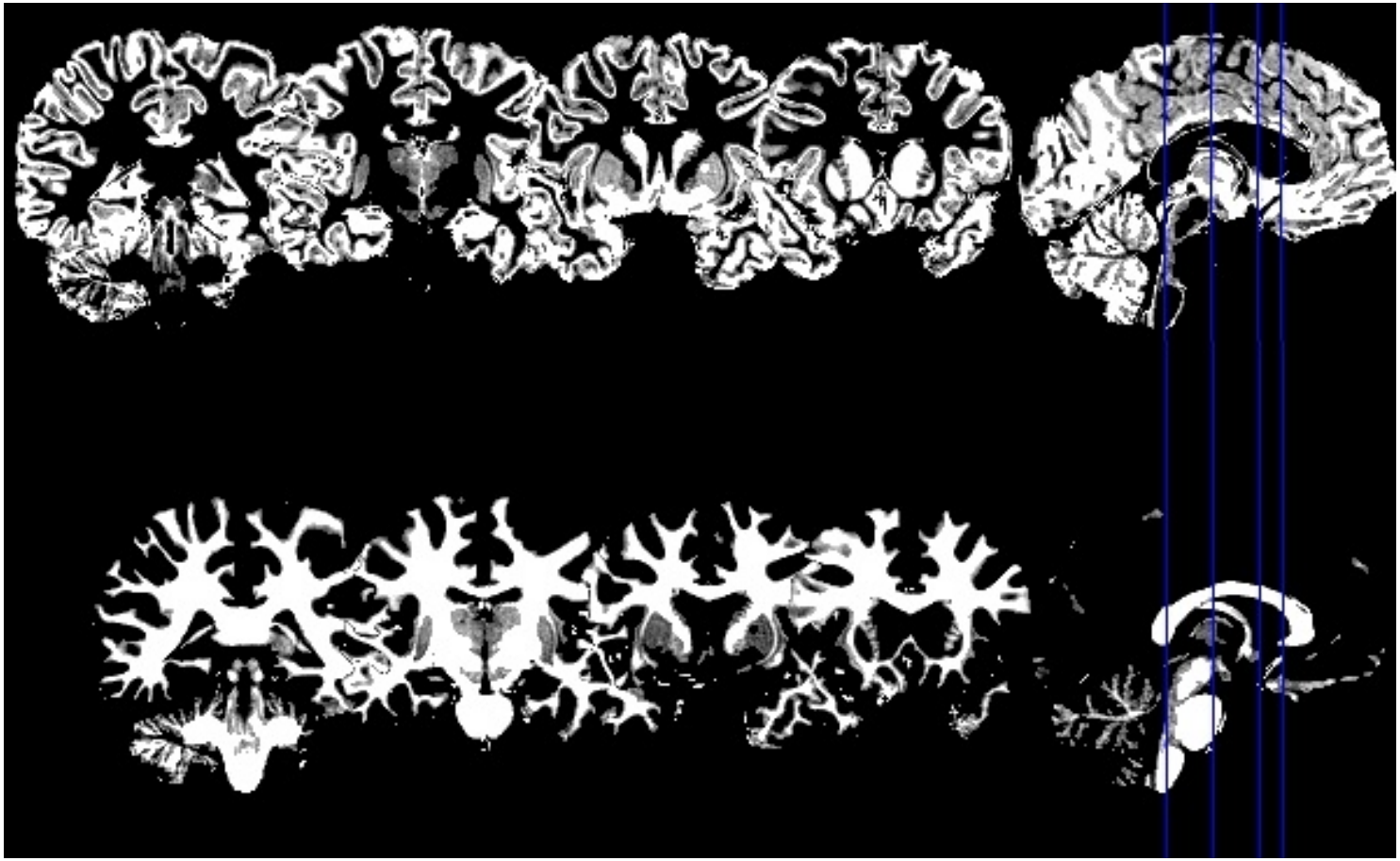
Preprocessing: BET (Brain Extraction Tool)



Preprocessing: Segmentation



Preprocessing: Segmentation



The complete vector of *selected features* (1)

The resulting complete feature vector for the **detection stage** includes the following 100 attributes:

- C: 3 attributes: gender, performance IQ, and full4IQ
- S: 41 features:
  - 38 PCs of eigenbrains,
  - 3 volumes of anatomical structures,
- F: 56 features:
  - 26 PCs calculated from pairwise linear correlations between the mean-signals,
  - 27 PCs calculated from pairwise linear correlations between the variance-signals,
  - 2 SAX features,
  - 1 spectral feature, calculated from the CSF (Cerebrospinal fluid) signal in the low frequency band [0.00 Hz ; 0.05 Hz),

The complete vector of *selected features* (2)

For the **subtype classification stage**, the best feature vector included 72 features:

- C: 1 attribute: gender,
- S: 33 structural features:
  - 31 PCs of eigenbrains,
  - 2 volumes of anatomical structures,
- F: 38 features:
  - 23 PCs calculated from pairwise linear correlations between the mean-signals,
  - 14 PCs calculated from pairwise linear correlations between the variance-signals
  - 1 SAX feature,

Conclusions

- Evolutionary feature selection seemed to be less effective than other feature selection methods.
- The feature selection method based on SVM weights seemed to work better.
- However, it turned out to overfit.
- Conclusion: overfitting is the major problem for FS/FW/FC.

Evolutionary feature synthesis

The idea

- Rather than select or weigh features, *invent* them, i.e.: define new features as quantities derived from the existing attributes.
- The new features either:
  - Augment the original attributes, or
  - Replace the original attributes.
- The well-suited EC tool: genetic programming.

Genetic programming

- A variant of EC where the genotypes represent *programs*, i.e., entities capable of reading in input data and producing some output data in response to that input.
  - Thus, in GP the solutions evolving under the selection pressure of the *fitness function* are themselves *functions.*
  - GP is by nature a ML approach.
- Fitness function f measures the similarity of the output produced by the program to the desired output, given as a part of task statement.
- Additional input data required:
  - Set of instructions (programming language of consideration).
- Canonical example: symbolic regression.
- Standard representation: expression trees.

- GP produced a number of solutions that are human-competitive, i.e., a GP algorithm automatically solved a problem for which a patent exists[11].
- A recent award-winning work has demonstrated the ability of a GP system to automatically find and correct bugs in commercially-released software when provided with test data[12].
- GP is one of leading methodologies that can be used to 'automate' science, helping the researchers to find the hidden complex patterns in the observed phenomena[13].

[11] Koza, J. R., Keane, M. A., Streeter, M. J., Mydlowec, W., Yu, J., Lanza, G., 2003. Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers.

[12] Arcuri, A., Yao, X., A novel co-evolutionary approach to automatic software bug fixing. In: Wang, J. (Ed.), 2008 IEEE World Congress on Computational Intelligence. IEEE Computational Intelligence Society, IEEE Press, Hong Kong.

[13] Schmidt, M., Lipson, H., 3 Apr. 2009. Distilling free-form natural laws from experimental data. Science 324 (5923), 81–85.

Crossover: exchange of randomly selected subexpressions (*subtree swapping crossover*).

1: **function** CROSSOVER(p_1, p_2)
2: **repeat**
3: $s_1 \leftarrow$ Random node in p_1
4: $s_2 \leftarrow$ Random node in p_2
5: $(p'_1, p'_2) \leftarrow$ Swap subtrees rooted in s_1 and s_2
6: **until** DEPTH(p'_1) $< d_{max} \wedge$ DEPTH(p'_2) $< d_{max}$ ▷ d_{max} is the tree depth limit
7: **return** (p'_1, p'_2)
8: **end function**

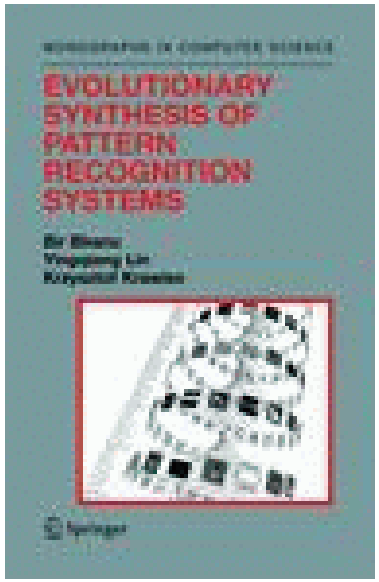
Mutation: replace a randomly selected subexpression with a new randomly generated subexpression.

1: **function** MUTATION(p, $\mathcal{I}$)
2: **repeat**
3: $s \leftarrow$ Random node in p
4: $s' \leftarrow$ RANDOMPROGRAM($\mathcal{I}$)
5: $p' \leftarrow$ Replace the subtree rooted in s with s'
6: **until** DEPTH(p') $< d_{max}$ ▷ d_{max} is the tree depth limit
7: **return** p'
8: **end function**

Why feature synthesis for image data?

What makes the visual data particularly suitable for feature synthesis?

- Answer: The structure of the data (internal organization), including:
  - Neighbourhood relation between pixels.
  - Temporal succession of frames in video sequences.
- This enables us to:
  - Bias the search for features towards the meaningful ones,
  - Synthesize features from raw image data.[14][15].
- Compare this to standard ML tasks, where typically there is no additional knowledge on attributes that could be exploited.



[14] Bir Bhanu, Yingqiang Lin, Krzysztof Krawiec, Evolutionary Synthesis of Pattern Recognition Systems, Springer Verlag, New York, 2005, ISBN: 0-387-21295-7.

[15] K. Krawiec, Evolutionary Feature Programming: Cooperative learning for knowledge discovery and computer vision. Wydawnictwo Politechniki Poznanskiej, 2004.

Case study 3: Vehicle detection

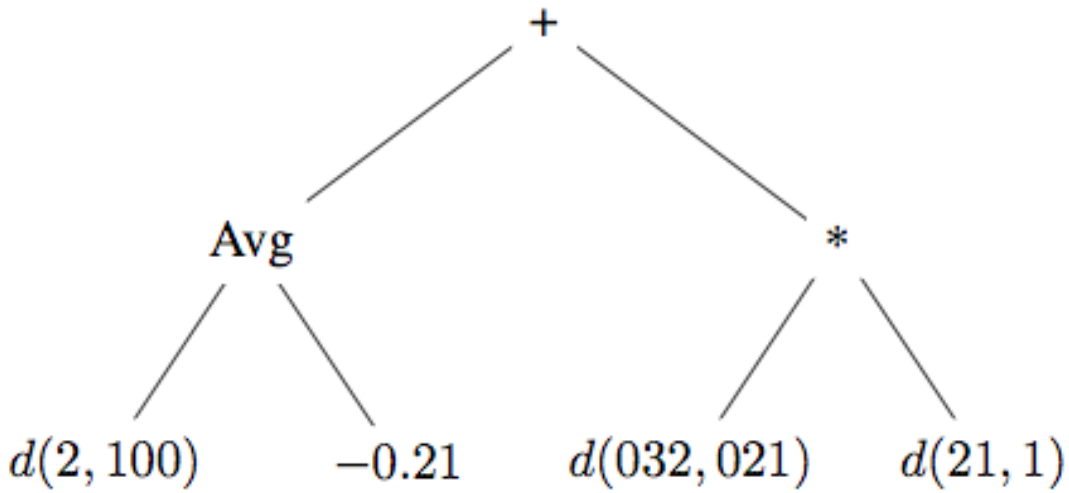
- Based on:
  Krzysztof Krawiec, Bartosz Kukawka and Tomasz Maciejewski (2010) Evolving cascades of voting feature detectors for vehicle detection in satellite imagery. In IEEE Congress on Evolutionary Computation (CEC 2010), IEEE Press, pp. 2392-2399.
- Employ GP to synthesize vehicle detectors.
- Vehicle detector inspects a 32×32 window in the input image.

Feature definition

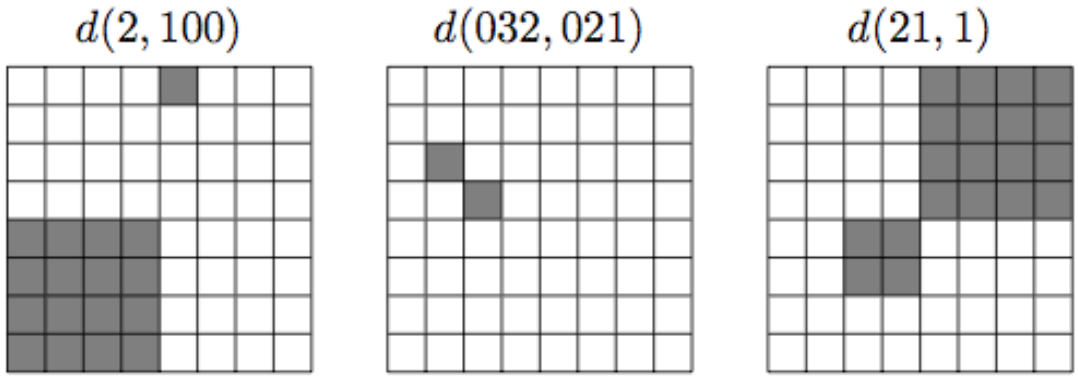
- A quad tree stacked over 32×32 input window
  - Tree nodes correspond one-to-one to rectangular image regions (*tiles*).
  - The nodes at consecutive depths correspond to 16×16, 8×8, 4×4, and 2×2 tiles; there are, 4, 16, 64, and 256 of them $\Longrightarrow$ total of 340 tiles.
  - Each tile uniquely identified by *quad key* – a variable-length sequence of quaternary digits.
- Feature $d(m, n)$ = difference between mean brightness values in *two* tiles identified by m and n.
- Total number of features: $340 \times 340 = 115,600$
- A clever trick (integral image) makes extraction of such features very effective (4x memory access + 3 subtractions).

Exemplary tree and accessed features

An exemplary GP tree (base classifier):



The tiles accessed by particular features (16x16 grid not shown):



Fitness function

- F-measure = the harmonic mean of *precision* p and *recall* r (sensitivity)

$$fitness = F_{measure} = \frac{2pr}{p+r},$$

$$p = \frac{TP}{TP+FP}, \quad r = \frac{TP}{TP+FN}$$

Detections

Detections

Detections

Remarks

- A single detector was not good enough to solve the task. Many different detectors had to be combined into a cascade to attain reasonable detection rate.
- In this application, the evolved feature *acts as a classifier* (detector).
  - No 'standard' ML classifier involved.
  - More sophisticated setups possible (feature synthesis + ML classifier)[16]
- Directly portable to other applications, including medical imaging.

[16] Krzysztof Krawiec, Bir Bhanu (2005) Visual Learning by Coevolutionary Feature Synthesis. *IEEE Transactions on System, Man, and Cybernetics – Part B*, 35(3): 409-425.

Overfitting

- Overfitting is the main headache for feature construction, because there are many more degrees of freedom here than for feature selection.
- Is quite common in all applications of GP (so-called *bloat*).
- Recommended countermeasures: Penalize too complex feature definitions in one of the following ways:
  - Explicitly, in the selection process (lexicographic parsimony pressure): resolve ties using length of feature definition.
  - By introducing penalty into fitness function (e.g., using MDL).

Other countermeasures against overfitting

1. Additional verification:
   - Before training, set aside a separate subset of examples, called *verification set*
   - After training (including feature selection) test the resulting model on the verification set.
   - Accept the model only if it performs sufficiently well on the verification set.

2. Make the evaluation function f more thorough
   - E.g., repeat cross-validation several times for different partition of training data into folds.

3. Embed the feature selection process into cross-validation.
   - The training process in each CV fold carries out an independent FS/FW/FC
   - Each CV fold produces its own (possibly different) features.
   - The problem: which of them to choose?

Concluding remarks

General conclusion

Evolutionary feature selection, weighing, and construction is an important alternative to more conventional methods, applicable in particular to medical imaging data.

- Its presence in software packages like Rapid Miner or Weka is an intermediate proof.

The aspects not discussed here:

- Suitable also for regression problems and unsupervised learning (requires different evaluation functions).

Also:

- Lot of other work done, not cited here.

Some recommendations

Don't use evolutionary feature selection if:

- The number of features is small (say, ≤ 20). Small numbers of features can be selected using exact search.
- The local search techniques (e.g., forward selection or backward elimination) give good results.

Use it when:

- The number of features is large.
- Single features have low information content (and, thus, only subsets of features provide satisfactory performance).

Other recommendations:

- For more advanced evolutionary approaches, consider Evolutionary Computation in Java library by Sean Luke
  `http://cs.gmu.edu/~eclab/projects/ecj/`

The alternatives

Do I need features selection at all?

Consider also other forms of representation transformation:

- Principal Component Analysis (PCA)
- Nonlinear PCA
- Singular Value Decompositon (SVD)
- Neural approaches (e.g., self-organizing map, SOM)
- For really large feature sets (1000's): Random selection

Possibly:

- Combine the above approaches with EC-based methods.
- Example: in the ADHD project, we used PCA to transform the raw data, and then FS.

Evolutionary FS/FC as a knowledge discovery tool

- The result of FS or FC algorithm can be inspected by humans.
- Example: A model of dependency of global temperature on on climate forcings discovered using GP: [17]

$$\hat{\tau} = v_1 \left(e^{N_2O+e^{v_2}}\right)^{-1}, \text{ where:}$$

$$v_1 = -e^{v_5} + AMO + CO_2\, e^{N_2O} + ENSO - 0.15502\, ln\,(1.10235)\, e^{N_2O} - v_6 - e^{-\left(e^{e^{N_2O}}\right)}$$

$$v_2 = -\left(e^{v_3 v_4}\right)$$

$$v_3 = -0.15502\, \frac{e^{-0.18342-0.15502\, e^{AMO}}}{-\left(CH_4 - e^{-\left(e^{N_2O}\right)}\right) + e^{-\left(e^{e^2\, N_2O}\right)}} + \left(e^{ln\left(VEI_{6,10}\right)}\right)$$

$$v_4 = -0.31004\, \frac{N_2O}{e^{N_2O} + ln\left(CH_4 - e^{-\left(e^{N_2O}\right)}\right)} + \left(e^{-\left(e^{AMO_{5,8}}\right)}\right)$$

$$v_5 = \frac{e^{-0.18342+NAO}}{-\left(e^{AMO+\left(e^{N_2O}\right)}\right) + e^{-\left(e^{e^2\, N_2O}\right)}\left(e^{e^{AMO+\left(e^{N_2O}\right)}} - N_2O\right)} + 0.13508$$

$$v_6 = ln\,(1.10235)\, ln\left(CH_4 - e^{-\left(e^{N_2O}\right)}\right) ln\left(-\left(e^{N_2O}\right) - 2\, N_2O + 2\, ENSO\right)$$

[17] K. Stanislawska, K. Krawiec, Z. Kundzewicz, Modeling Global Temperature Changes with Genetic Programming, Computer and Mathematics with Applications, 2012 (to appear).

Future/current directions

- Exploit the strong sides of EC as a tool of FS/FC:
  - Global search
  - Capability of combining feature subsets.
- Reformulate the task in a multi-objective manner (e.g., accuracy and number of features).
- Work on overfitting prevention.
- Decomposition of feature selection/synthesis task (e.g., using cooperative coevolution)

Thank you.

http://www.cs.put.poznan.pl/kkrawiec/
krawiec@cs.put.poznan.pl